COMPUTER COMMUNICATION REVIEW

A Publication of ACM SIGCOMM

Volume 33, Number 1 January, 2003
ISSN #: 0146-4833

Contents

Newsletter Sections

Information for Authors

By submitting your article for distribution in this Special Interest Group publication, you hereby grant to ACM the following non-exclusive, perpetual, worldwide rights:

- to publish in print on condition of acceptance by the editor
- to digitize and post your article in the electronic version of this publication
- to include the article in the ACM Digital Library
- to allow users to copy and distribute the article for noncommercial, educational or research purposes

However, as a contributing author, you retain copyright to your article and ACM will make every effort to refer requests for commercial use directly to you.

Additional information for authors is available at the CCR website: http://www.acm.org/sigcomm/ccr

Notice to Past Authors of ACM-Published Articles

ACM intends to create a complete electronic archive of all articles and/or other material previously published by ACM. If you have written a work that was previously published by ACM in any journal or conference proceedings prior to 1978, or any SIG Newsletter at any time, and you do NOT want this work to appear in the ACM Digital Library, please inform permissions@acm.org, stating the title of the work, the author(s), and where and when published.

Association for Computing Machinery

Special Interest Group Data Communication

acm sigcomm

January 2, 2003

Dear SIGCOMM Colleagues:

The past six months have been very busy for SIGCOMM. I'd like to give you a brief account of what has happened and what new things are coming in the next few months.

First, we held the annual SIGCOMM conference in Pittsburgh. Overall, the conference was outstanding. Peter Steenkiste and Matt Mathis were wonderful hosts at an old and architecturally distinguished hotel in Pittsburgh. The conference program, thanks to program co-chairs Vern Paxson and Hari Balakrishan, was filled with exciting papers. Our surveys of the conference attendees indicated that they found the conference intellectually powerful and productive.

The only bad news from the conference was that it lost money, despite strong sponsor support and near record attendance. The loss came from unprecedented student participation (50% of the conference attendees). As you know, SIGCOMM for several years has kept the student registration fees below cost to encourage student participation. Having now succeeded in attracting students (beyond our wildest expectations), we need to make the student fees more accurately reflect costs. Expect the student registration fees for SIGCOMM 2003 to be more in line with those of other conferences.

The new HOTNETS workshop (organized by Larry Peterson, David Wetherall, and Chris Edmonson) was held in Princeton in late October. The workshop is devoted to emerging ideas in networking and so we heard a lot of exciting, and not always fully-baked, ideas. The result was an intellectually vibrant two days. Some of that excitement is captured in this issue of CCR, which contains the proceedings and workshop report. We expect to hold another HOTNETS workshop in either late 2003 or early 2004.

The second Internet Measurement Workshop was held in Marseilles in early November. As with the first workshop, there was a lot of exciting work presented. One senior colleague tells me it was the first technical meeting in years where he felt he really needed to listen to each paper presentation. Plans are already afoot for an Internet Measurement Conference in 2003.

SIGCOMM is also involved in the creation of two new conferences. SIGCOMM will be the primary sponsor of the ACM Conference on Embedded Networked Sensor Systems (SenSys). SenSys 2003 will be held in early November at UCLA. And we're in discussions with ACM SIGOPS about creating a conference (tentatively entitled PONDS) about problems that sit on the boundaries of operating systems and networking.

SIGCOMM has also just completed negotiations with IFIP (thanks to Lyman Chapin for taking the lead on this agreement) to cosponsor future Latin American Workshops on Data Networking. We hope to collocate the next workshop with the CLEI conference in 2003.

Finally, a reminder to vote. The current SIGCOMM elected officers complete their terms this summer. So this spring you'll be receiving a ballot asking you to elect new SIGCOMM officers. I don't know who is going to be nominated for what offices, but given the outstanding set of people that the Nominations Committee has encouraged to call me for a briefing on what it is like to lead SIGCOMM, I expect we'll have a great slate. Also on the ballot will be (I hope) an opportunity to vote on new SIGCOMM bylaws. The current SIGCOMM bylaws are a decade old and no longer reflect how SIGCOMM really operates.

Sincerely,

Craig Partridge
Chair, ACM SIGCOMM

HotNets-I: Workshop Summary

David Wetherall and Larry Peterson
Program Co-Chairs

Introduction

The first HotNets workshop was held on October 28 and 29, 2002 at Princeton University. Twenty-five papers were presented over a day and a half in a single-track, discussion-oriented format. Attendance was kept to 65 people to encourage interaction among the participants. The participants were drawn mostly from presenters and authors, plus the organizing committee and 10 students generously supported by an NSF grant.

We organized HotNets with specific goals in mind. We wanted to foster "new idea" papers on early-stage, creative networking research, especially those discussing architectural or high-level design decisions. We also wanted to provide a forum for reflective discussions on the research needs and direction of the broad networking community. The need for such a venue had become steadily apparent to us. Top-tier venues such as SIGCOMM have become highly selective (1 in 12 in 2002!) with the side-effect of discouraging risky papers. Part of the difficulty is that "new idea" papers can be challenging to evaluate because judgment and wisdom are often more important than detailed simulation and measurement. Moreover, when papers cover new ground, there is often little agreement about the proper set of assumptions and importance of problems. Because few existing venues are suited to "new idea" papers, few papers of this kind are published. This is unfortunate, since good architecture and design-oriented papers are often important to the future of networking and have a different impact than purely technical results.

We are not the first to recognize this problem. The CSTB's *Looking Over the Fence at Networks* report[1] in 2001 highlights the "ossification" of the Internet and networking research, and calls for networking research to "more aggressively seek to develop new ideas and approaches." The SIGCOMM Technical Advisory Committee report[2] in 2001 puts forward proposals for re-invigorating the SIGCOMM conference, including complementary workshops. We hope that venues such as HotNets can play a role in stimulating innovative new networking research by providing the community feedback needed to help early-stage, creative work mature into solid conference papers.

The response to HotNets in this role has been positive from its inception. We loosely modeled HotNets on HotOS and the SIGOPS European Workshop, successful workshops in the systems community. We focused on short (6 page) position papers that argued a thoughtful point-of-view rather than full papers that included new results, gathered a small and tightly-knit Program Committee, and emphasized breadth and diversity of topics. The PC included a mix of experience and active researchers in the areas of sensor networks, wireless, and operating systems, as well as the more traditional SIGCOMM topics. Our Call For Papers was even broader.

We received 127 submissions, many more than we had anticipated, across a fairly broad range of topics. We accepted 25 papers, which is quite selective for a workshop. Interestingly, almost half of these papers were based on fresh SIGCOMM rejects! (We leave it to the reader to guess which ones.) This suggests to us that there is a strong demand for a position paper outlet in the networking community. At the workshop itself, we reserved fully half of the formal time for discussion, making do with 15-minute paper presentations and discussing groups of papers together where we could. The size of the workshop, around 60 people, was judged by most to have worked well at providing a welcoming atmosphere for discussion.

Summaries of the workshop discussion are contained in the body of this report. Several overall themes can be seen to emerge. A number of papers led to a discussion of the research process itself: our models and simulation practices, along with testbeds that seek to change the way we do research (and technology transfer). There seemed to be much scope for improvement in these areas, and we note that the NSF has recently held a workshop and announced the creation of a new program in Networking Research Testbeds.

Other clusters of papers were accepted in the areas of peer-to-peer, sensor networks and network and protocol architecture. Papers in these areas mostly explored approaches quite different than the norm. Some of the P2P papers advocated combining information retrieval techniques with P2P structures to enable search. Papers on sensor networks helped to lift us out of an Internet-centric view of networking. The set of papers on architectural issues ranged from the use of overlays to work toward QoS, to the technologies in the future Internet core. The remaining papers constituted a mix of material, little of which shows up in most networking conferences: economic models of networks, router design, network management, and the use of techniques like epidemics in network protocols. This broad set of topics and revolutionary rather than evolutionary papers were very much

what we sought to accept when we first planned HotNets.

There are many people we would like to thank for taking HotNets from an idea to a reality. The other members of the Program Committee (Deborah Estrin, Stefan Savage, Srini Seshan, Scott Shenker, Ion Stoica, Amin Vahdat, and John Wroclawski) contributed an enormous amount of time and energy (far more than they anticipated!) to select a lively program. We asked each PC member to read roughly half of the 127 submissions, including all the likely candidates for acceptance. We were fortunate that Chris Edmonson-Yurkanan took on the role of General Chair. She brought much experience to the task, and this ensured that all of our plans unfolded smoothly. Both the paper reviewing process and the workshop itself ran smoothly too, and we thank Tammo Spalink and Melissa Lawson for much behind-the-scenes "heavy lifting" that made this possible. ACM SIGCOMM, especially Craig Partridge, provided encouragement and lent its sponsorship to the workshop. The NSF, and in particular Ty Znati, helped us by providing a grant that included student travel stipends. We are also indebted to our other sponsor, Intel Research, for key support that enabled a small workshop to remain financially viable. And we wish to thank all of our scribes, listed at the end of this report, for the discussion summaries that form the body of this report.

Planning for HotNets-II is well underway, and we are pleased that Scott Shenker and Hari Balakrishnan have agreed to be PC co-chairs. We hope that HotNets will grow into a successful venue that works in synergy with leading networking conferences by providing feedback and exposure to early-stage work, helping it to grow into mature research, and we would be delighted if HotNets papers help to break new ground for conference publications. How well HotNets succeeds in these respects, however, depends on your support and participation.

Session 1: Architecture

David Wetherall chaired the first session, which included three talks describing architectural proposals. The first, by Ion Stoica, argued that we have failed to deploy QoS in today's Internet, at least partly due to the fact that ISPs have little incentive to change IP. However, with the success of many overlay networks, a natural question to ask is whether overlays can be used to provide QoS functionality. His answer is yes, at least in some cases. Stoica described the architecture of OverQoS, where overlay nodes are deployed in different Autonomous Systems (ASes) within the Internet, and virtual links between entry and exit nodes are used to carry a bundle of application data packets from multiple transport flows across different sources and destinations. The key idea is to use a controlled-loss virtual link (CLVL) abstraction that aggregates a set of flows along a virtual link into a bundle, uses FEC or ARQ mechanism to protect bundle traffic against losses, and redistributes bandwidth and loss across flows within a bundle at the entry node.

During the question period, Dan Rubenstein asked if OverQoS is TCP-Friendly? Stoica responed that the goal of OverQos is not about being TCP friendly, but rather to control losses. Stefan Savage asked if the CLVL abstraction is point-to-point and if there are any routing choices? Stoica said this issue is future work, and that there are no easy answers. Another participant asked if OverQoS supports fair sharing among flows, to which Stoica responded that CLVL is an abstraction; there's freedom in how one manages flows within a bundle.

Bob Braden presented the second talk, which advocated an alternative to the traditional layered architecture: a so called *role-based architecture* (RBA). The idea is motivated by problems with layered architecture; e.g., layer violations, sub-layer proliferation, feature interactions, and middlebox's breaking the end-to-end model. Many of the problems seem to be related to traditional layering. To provide better clarity, more generality and in-band signaling with data flow is need. Braden suggested that we could change the way we think about protocols by giving up layers and using RBA, in which functional building blocks are called roles, packet headers include an arbitrary collection of sub-headers or role-specific-headers (RSHs), and header processing needs new rules for ordering and access control.

Craig Partridge asked if RBA just creates a new name space, which is a known pain. How can we ensure that there are no fights over names? Braden responded by saying that roles are static, so a hierarchical name space should suffice. David Wetherall asked if everyone would need to be aware of roles, as opposed to today, where IP is the single common protocol. Braden replied that roles do not control routing; routing is done as today. Roles are purely local (i.e., are known only inside a node). Larry Peterson asked if roles are significantly different than IPv6 extension headers. John Wroclawski responded that in addition to supporting out-of-order processing, RBA also introduces a problem of role discovery, which is a significant difference from IPv6. Jay Lepreau asked if there are security consequences of out-of-order processing. The answer is yes. Finally, there was a discussion about whether role interaction is a problem, with no consensus reached.

Richard Gold presented the final talk of the session, which began with the observation that there are many forces working against the Internet as a black box, including CDN plumbers, RON do-it-yourselfers, DDoS firefighters, and P2P people. This is because end nodes may know more than BGP (e.g., RON), and ISPs may know more than BGP (e.g., CDN). His main argument is that IP is too direct. The Internet needs an additional level of indirection that allows one to allocate a remote network pointer, and then direct traffic to that pointer, which forwards the traffic to its ultimate destination. He advocated the idea of *network pointers* (NP), which live below IP, thereby making IP an overlay on top of network pointers. NPs can be dynamically allocated and also have processing associated with them. Gold spent the rest of his talk walking through several scenarios that might benefit from network pointers.

Craig Partridge observed that all the papers in this session are creating name spaces, there is fighting over name spaces already, and this is the problem. Braden responded that IANA would assign role IDs and that perhaps there would be a hierarchical name space. Gold replied that NP names are purely local—there is no global name space problem. Gold was also asked about dangling pointers, to which he

replied that they would be handled by higher levels, in an application-specific way. He remarked that NPs can be hidden or explicit (in terms of discovery).

Session 2: Models

Amin Vahdat chaired the second session, which discussed the models we use in networking research. Sally Floyd presented the first paper, which argued that incorrect models constitute a major problem for network research. She provided examples of unrealistic assumptions that have led researchers to wrong conclusions. To make Internet models as simple as possible but not simpler, Floyd proposed application-specific modeling, where only those aspects that are relevant to the examined problem are represented precisely.

The talked provoked a long discussion. Tom Anderson suggested that we need better protocols rather than better models. Floyd responded by saying we need both, the point being that wrong models lead to wrong conclusions about protocols. Stefan Savage then asked if it is the right approach to come up with a model first and to research and design protocols later? Floyd replied that meaningful research requires correct models. Craig Partridge asked how we can get the correct measurement data needed for model validation? Floyd suggested that we start by telling network providers what we need, although we do not know all the relevant parameters right now.

Timothy Roscoe argued that by using an application-specific model, we potentially miss the overall impact of the application on the network. Floyd replied that application-specific models do not narrow the focus. They still represent the complete system, although only relevant aspects are modeled precisely. Ion Stoica wondered if the focus on the right experimental setups creates a hidden danger of not accounting for future developments? Floyd noted that "designing for a benchmark" is bad, but that the analysis of designs should substantiate their merits. David Wetherall suggested that each design should be tested for its sensitivity to all relevant parameters? Floyd agreed, but noted that researchers rely on intuition and their own mental model of what is relevant. It is important to learn which parameters are relevant. Finally, Eddie Kohler and Jay Lepreau both observed that the infrastructure they are building (see next section) attempts to provide such support.

In the second talk of the session, Konstantinos Psounis presented a new methodology for performing large-scale simulations: (1) take a sample of the network flows, (2) feed the sample into a scaled-down version of the network, and (3) extrapolate from the performance of the scaled-down network to that of the original.

During the follow-on discussion, Psounis was asked if in addition to predicting average statistics, his approach could predict the distribution tail, to which he replied that they predict the whole distribution including the tail, but that this can take long time. Taieb Znati then asked if the method is applicable to weird distributions? Psounis responded that it was not, but one can sample on the session level that exhibits a suitable distribution. David Wetherall asked how much the approach can scale down simulations without degrading the prediction accuracy, for example, with errors less than 5%? Psounis pointed out that he had presented two cases. In the first case, scaling-down is unlimited. In the second case, scaling-down by a factor of 50 preserves the precision. Psounis explained that the authors have not looked in general at topology-related problems, which can impose a limit on scaling.

During the final talk, David Alderson suggested a new approach for generating a topology for network modeling and design. The approach formulates topology generation as an optimization problem that considers economic tradeoffs and technical constraints faced by a single ISP in its network design. Solving the problem provides realistic annotated topologies for different levels of ISP hierarchies.

During the discussion, Alderson was asked if his approach considers time as a factor. He replied that the evolutionary growth of the Internet has to be addressed, and that the formulation of his optimization problem can account for legacy issues and requirements of incremental deployment. In response to a question about how the authors know they are right, co-author Walter Willinger responded by saying that to validate the results they need to examine various Internet topologies. Finally, Timothy Roscoe asked a general question about the differences between the agendas of ISPs and the researchers at HotNets. The consensus was that ISPs think the researchers are too theoretical, even though we can tell them many useful things.

Session 3: Infrastructure

John Wroclawski chaired a panel on infrastructure for network research. Eddie Kohler began by describing XORP, an open extensible router platform. Kohler argued that network research is increasingly divorced from reality, with a growing gap between research and practice. Trying new ideas in commercial routers would help narrow this gap, but unfortunately, router vendors are reluctant to deploy new services if there is no immediate monetary return. Even when router APIs are exposed, the basic router functionality is still immutable. XORP makes it possible to experiment with new protocols on routers. It currently includes open APIs for WWW, RTP, SIP, and an open implementation of TCP SACK, and IGMPv3. The code is likely to be released with a BSD-like license.

Stefan Savage commented that XORP's robustness definition (it shouldn't crash) may not be correct. Kohler said that this was a fair point. In this case, robustness means that the forwarding path doesn't crash. Larry Peterson asked about performance robustness; what about a service that runs too long? Kohler replied that the robustness definition is different for user code vs. the forwarding path, and for or the forwarding path, robustness will likely be provided through code review.

The second panelist, Jay Lepreau, argued that research on wireless and mobile communications, as well as on sensor networks, can greatly benefit from emulation and testbeds. He then described how his group is extending Emulab to include a wireless space/time-shared testbed. The idea is to give PDAs and laptops to students, and either attach sensors to public buses or lay them out in large (empty) hangers.

Ratul Mahajan asked about the privacy of students. Lepreau responded that there must be appropriate guidelines to protect privacy. Deborah Estrin asked how hangars can be used for research on sensor networks. What is the input to the sensors? This brings up the issue of the line between reality and simulation when sharing. Craig Partridge noted that there has to be some input (e.g., a moving tank) and Brad Karp asked about interference of external factors, and how they might be estimated/controlled remotely. The consensus was that reproducibility is always a concern in mobile experiments.

Finally, Larry Peterson described the PlanetLab overlay testbed, which is designed to support emerging research into geographically distributed services and network measurement. PlanetLab's goal is to have 1000 viewpoints into the Internet, including both edge sites and network crossroads (e.g., co-location centers). Peterson argued that PlanetLab has the potential to evolve into a service-oriented network architecture in which a common set of base services provide the foundation for high-level services.

During a general discussion period, Peterson was asked what control sites have over the installed PlanetLab nodes. His response was that the main knob is the output bandwidth of the interface card — that is, the percentage of host infrastructure a PlanetLab node is allowed to utilize. Stefan Savage asked why the presenters for the session chose to specialize. Why don't you play together? Kohler responded by saying that his group doesn't have code yet and are focused on routers. Peterson responded by saying that performance is at the bottom of their list of concerns right now, but that eventually PlanetLab will need to move to the physical path (rather than the proxy path) and so the extensible router infrastructure will become useful. He also expects a growing synergy between emulation and PlanetLab's real-world Testbed. In response to a question about the diversity of node placement in PlanetLab, Peterson said the distribution was currently a bit Internet2-centric but that nodes will be going into co-lo centers soon. He also said that PlanetLab would like to see individuals at member sites take machines home and connect them to PlanetLab by cable modems and DSL. Brad Karp directed a question to Peterson and Lepreau, noting that if these projects are successful, researchers will have access to the largest measurement projects in history. Are there tools to make sense of all the data? Peterson responded by saying that building an instrumentation service for PlanetLab is one of their top priorities. Werner Vogels commented that Cornell gave laptops to students, but the real problem was to track them, as it's expensive to put access points everywhere.

Session 4: Routing

Stefan Savage chaired a session on routing. Timothy Roscoe gave the first talk, and began with the observation that routing and firewalls are separate processes in the Internet. He then proposed a new perspective, called *controlled networking*, in which routing and filtering are unified operations. With controlled networking, the presence of every packet is precisely assured and every packet flow is explicitly "white listed". Specifically, a predicate describes what can be seen and what is allowed. He claimed that there is no need to change routers, end-systems, or IP itself to achieve this.

During the discussion that followed, he was asked if there could be high-level languages to specify policy? Roscoe responded that predicates are fairly low level, and probably suitable for being implemented in hardware. Several people then questioned the expressiveness of the system: for example, if it would allow one to stripe packets over multiple paths. Roscoe said such things might not be captured in this model, although the functionality they provide is pretty basic.

Dapeng Zhu gave the second talk, pointing out the weakness of the current interdomain routing protocol (BGP), specifically that it is vulnerable to a single misconfigured router. This is because meaningful filtering/verification of routing updates is not done due to scalability concerns. BGP takes 3 minutes to fail over on average (15-30 min in reality), which prevents mission critical applications from using the Internet (or at least the current IP-level recovery mechanism). He stated that the fundamental problem of BGP routing is that every BGP router needs to have a consistent view of the world, and then proposed that we use a new interdomain routing protocol called Feedback Based Routing (FBR), which basically separates static information (topology) from dynamic information (quality of routes).

During the discussion, Stefan Savage noted that since ISPs try to get traffic off their own network as soon as possible, they have no incentive to adopt the proposed scheme. Zhu responded that it is up to the ISPs, and how they want to tradeoff between cheaper with less traffic on their own networks and more expensive but more robust alternatives.

The next talk, by Venkata Padmanabhan, started with the observation that networks are vulnerable to malfunctioning routers that may compromise routing or misroute packets. Similar problems are also present in other scenarios, such as wireless and p2p networks. He then argued that existing techniques to deal with the problem include flooding link state routing information (unscalable), authenticated routing advertisements (doesn't guard against compromised routers), and central repository (ISPs don't share policy information). Moreover, all these techniques try to protect routing; they don't verify forwarding. He proposed secure traceroute as a technique to detect faulty or malicious routers. The approach assumes single-path routing, and verifies the origin and contents of data. It uses keys for hop-by-hop verification, and identifies faulty links and routers.

Padmanabham was asked what can be done after discovering a bad link. He said that packets should be routed around it if possible. He was also asked if secure traceroute could prevent routers from faking links or routers behind it. Padmanbham said no, but one could apply secure traceroute persistently to discover the problem.

In the final talk of the session, Douglas De Couto reported his experiences with routing in ad hoc networks. He noted that the min-hop-count routing metric is well understood, and the alternatives are complex than simply assuming that hop-count is the most important metric, even in wireless networks where link quality is bimodal. He showed simulation results demonstrating that DSDV is not opti-

mal, and listed some reasons for this being the case, the bottom-line being that the intuition for wired networks is wrong for wireless networks. On the other hand, he argued that a reasonable alternative—maximizing bottleneck bandwidth—doesn't work in practice. Instead, the key goals should be to maximize spectrum usage by minimizing packet transmissions.

During the question period, it was noted that the design space is very large, and that other parameters such as delay should also be considered. De Couto said that these metrics optimize throughput. Ion Stoica asked what additional factors are specific to wireless? De Couto said that the variation in link quality is important, and things generally happen faster; e.g., there are external factors, such as doors opening, elevators moving, and so on.

Session 5: P2P/Overlay Networks

Ion Stoica chaired a session on peer-to-peer (P2P) and overlay networks. Chunqiang Tang gave the first talk, and began by making the following observations: (1) search engines index less than 16% of the web, (2) the Internet is growing by a factor of 3 each year, and (3) Google is growing by a factor of 2.5 each year. The upshot is that search engines only index a modest fraction of the Internet and the growth curve of search engines is not keeping up with the growth curve of the Internet as a whole. He then described pSearch, which has the goal of providing a scalable information retrieval (IR) infrastructure using P2P. He also argued that current approaches are deficient. For example, existing P2P systems are either not scalable, not accurate, or both. Distributed Hash Table (DHT) systems are scalable but do not directly support full-text search.

During the discussion, Tang was asked about an A/V searcher. He said that has been work at IBM to represent A/V files as vectors so this approach should still work.

The next speaker, Edith Cohen, argued that peer-to-peer overlays need a versatile and scalable search mechanism with two features: scope and support for partial-match queries. While centralized solutions provide both features, existing decentralized schemes offer at most one. Cohen proposed associative overlays, a new architecture that supports both partial matches and search for rare items. Associative overlays couple the overlay topology with the search strategy.

During the question period, Lakshminarayanan Subramanian asked why we can't use a web search engine instead? Cohen said that we are looking for a decentralized solution. Eugene Ng asked why we can't use structured overlays, to which Cohen replied that structured overlays rely on distributed hash tables that support only exact matches. We would like to support imprecise queries, like in Google. Chunqiang Tang added that unstructured overlays are also easer to maintain, and that data is growing faster than centralized architectures can cope with. Cohen also noted that centralized approaches also have legal, political, and economic problems. Craig Partridge asked if there is a fault-tolerance tradeoff between centralized and decentralized solutions? Marcel Waldvogel responded that it is easier to mount attacks on decentralized schemes, and Timothy Roscoe commented that P2P services can be easily disrupted and shut down. Cohen responded that P2P services can be made robust.

In the next talk, Marcel Waldvogel describes MITHOS, a scalable p2p overlay network infrastructure for data location that supports low latency routing and fast forwarding using minimum information. After reviewing possible structures for data location—e.g., a DHT, d-dimensional (ir)regular mesh, rectangular tile, distributed tries—he argued that one would like to have a geographic layout in which less routing information is necessary, a rough estimate of relative distances is possible, and even third-parties can figure the distance. His approach was to assign Cartesian coordinate as the id for each node, and do a quadrant-based routing. Since a node's id cannot be known a priori, the proposed approach determines the id during join phase, bootstrapping from some known member.

During the question period, Eugene Ng wanted to know what Waldvogel meant by the possibility that MITHOS could possibly replace IP routing. Waldvogel replied that he had ambitiously stated that as a possibility.

In the last talk of the session, Mayank Bawa observed that the metrics we normally use to evaluate Application Level Multicast (ALM) include stress (duplicate packets on same physical link), delays in routing along an overlay, and increased usage of network resources. Then he argued that existing metrics applied to ALMs are incomplete, and that one should also account for transience of peers and its impact on connectivity of multicast sessions. For example, peers show herd behavior in unstable P2P networks. In order to achieve good end-application performance on such an unstable infrastructure, one must mask peer transience by keeping the topology connected, and by continuing application interrupted. He proposed an *infrastructural peering* layer below applications, between the end-application and data-transfer layer, that is, between RTSP and TCP. This layer maintains state to decouple the two sessions and serves as place holder for primitives for resource discovery and policies for maintaining topology.

Craig Partridge asked why adding a layer helps? Bawa responded by saying that peering functions are difficult and put a heavy burden on application developers. Thus, the peering layer separates concerns: someone provides peering functionality, and all applications use this functionality. Bob Braden suggested that an alternative conclusion to draw was that that RTSP provides insufficient functionality. Venkata Padmanabhan asked, since peering in media streaming is application-specific, what would serve as common useful functionality? Bawa replied that it could be the specification of topological policies; e.g., vertex-disjoint or edge-disjoint paths.

During an extended discussion of the entire session David Wetherall said that the speakers made a strong case for doing IR using P2P, so why don't we throw out DHT and always do this. What is the downside? Someone wanted to know if unstructured overlays make hard-to-find items become *really* hard to find? Cohen commented that DHTs would only work with structured queries, and Tang said that unstructured overlays are easy to maintain and that he is

trying to find a middle ground between structured (DHT) and unstructured systems. Someone asked why not use a centralized service? Cohen pointed out that a centralized service like Google has legal issues to consider. Tang said that a decentralized approach may be better at keeping pace with the growth of the Internet and that it may have better fault tolerant attributes. There was then a long discussion about whether distributed architectures are really better than centralized ones. Stefan Savage concluded that the design space has not been fully explored. We have been restricted to napster, gnutella, and DHTs. In reality, machines have different capabilities, and we should leverage that. Timothy Roscoe objected to the argument that P2P systems have better fault tolerance, since "Civil Attacks" are a very inexpensive and efficient way to bring down a P2P network.

Session 6: Network/Protocol Design Issues

Deborah Estrin chaired a session on network and protocol design issues. Hui Zhang presented the first talk, which argued several points: (1) IP is yet to conquer voice and public data networks; (2) today we have SONET at the edge, and WDM in the middle of the network; and (3) SONET has had a faster growth spurt than the Internet. Zhang then question various IP myths—such as efficiency, simplicity, robustness and low cost—and conclude that IP is a good service layered network (e.g., enterprise voice) but that the core of the network will remain circuit switched.

During the question period, David Wetherall wanted to know what Zhang was advocating. Zhang responded that he is asking a question, more than providing an answer. Where do we want IP to be? Should it takeover the service layer or be the core. Zhang's position is that the former is more realistic. A general discussion about the economics of IP-based networks versus circuit-based networks ensued, with little agreement. Zhang then went back to his original argument, which is that IP has to be driven by enterprise level services; it will take over from the edge. John Wroclawski remarked that this is what already happens, and that Zhang must be advocating something else? Zhang replied that IP's success is in its service model, and a design that deals with heterogeneity, but it is not a complete success in deployment. Timothy Roscoe supported this position by noting that anyone in the business of making money is not using IP in the wide area right now.

The next speaker, Cyriel Minkenberg, shared his experiences working on a multi-terabit-per-second router (2-3 Tb/s) targeted at the OEM market, and comprising either 256 OC-192 or 64 OC-768 links. The design uses a single stage, electronic (not optical) switch. He pointed out that most of what he is presenting is well known to the switch/router design community. He then gave a practical perspective on how most well-known techniques don't work beyond a terabit. For example, he reported that power (and not gate count) is the limiting factor with respect to the number of chips and boards in a system. He also remarked that packaging options are constrained for a variety of reasons, including building codes, form factors, industry standards, power budget (cooling), and serviceability.

Stefan Savage asked that if power and packaging are the main concerns, then why is circuit switching simpler. What are the differences between the design and implementation of circuit and packet switched designs? Minkenberg said that optical switches are well suited to circuit switching. The drawback is that one needs core optics.

In the next talk, Tom Anderson motivated the need for robustness in protocol design. He started with a list of configuration and protocol related problems seen in the Internet, and argued how they did more damage than the Baltimore fire or the WTC attack. He claimed that better protocols are possible, and without too much cost. He backed up this position by giving several examples of major incidents in the Internet, including how BGP handles errors (which results in cascades), how the TCP connection establishment protocol leaves the door open for SYN flooding, and how TCP's fast recovery algorithm is open to attack. He concluded by arguing for a better design methodology—not just software engineering, but better designs—and presented a set of guidelines.

During the discussion period, a questioner pointed out that the guidelines were too general, and designers should have been following them all along. Anderson argued that this clearly has not been the case. Someone then asked if IP-based protocols are less robust? Anderson disagreed; others such as ATM have not been tested.

Werner Vogels presented the final talk of the session. He said the starting point for their work was their troubles with reliable multicast—bigger groups lead to bigger problems (chances of a member being unavailable increase), and the throughput went down under stress. At that point, the authors moved to designing robust distributed systems based on epidemics, which provide probabilistic guarantees, asynchronous communication patterns, and robustness to message loss. He said that even though the science is well understood, epidemic-based systems require significant engineering to work.

Deborah Estrin asked if the size of the system was a problem? Vogel said that communication is scalable, but you need to know complete membership. Local state is on the order of number of nodes, and this is a problem; i.e., how do you select a small subgroup an still provide global guarantees?

During an extended session-wide discussion, a questioner asked Hui Zhang what the utilization of circuit switched networks was. He said roughly 40%. At this point a question was raised as to how this is counted, specifically how is reserved bandwidth that is not used counted. Someone else commented that packet switched networks have to be over-provisioned because of highly dynamic traffic patterns. Tom Anderson blamed TCP for this. Badri Nath asked Zhang what his view of the world was? Zhang said that he is only pointing out facts that the IP community has been ignoring. Brad Karp noted that Cyriel Minkenberg questioned some assumptions, and argued for eliminating some features to bring down the cost of routers. He asked if there are any other assumptions that might be appropriate to discard, such as in-order delivery. Minkenberg responded positively to that suggestion. Someone asked Tom Anderson

about whether following his guidelines would make protocols more complex. Anderson said that there was a strong case for simplicity, but some of our notions of simplicity, such as "circuit switching is complex" are wrong. He also noted that most vulnerabilities he pointed out could be attributed to simplifying assumptions.

Session 7: Sensor/Ad Hoc Networks

Larry Peterson chaired the final session on sensor and ad hoc networks. Sylvia Ratnasamy gave the first talk, and began by explaining that the context of her work is very large networks (a million or more nodes), where each node has local memory, processing, and short-range communication. The goal is to find a scalable and energy efficient method of data retrieval. She pointed out that the main difference between her work and previous work is that her work focuses on data centric *storage* (DCS). The idea is to store data in the sensornet by name, similar to DHTs. Without systematically placing specific data on specific nodes, queries can only be made by flooding the network, with replies returned by the reverse path. With DCS, however, event information is shipped unicast to the specific node assigned to store that event. As a result, queries do not require flooding and can be made more efficient.

During the question period, someone asked what about catastrophic failures of a node that contains important event information? Ratnasamy said one can replicate the data. In response to a question about other fault tolerant issues, she said you can store data along the retrieval path.

Deepak Ganesan gave the second talk on a new data handling architecture for resource constrained sensor networks. The system, called DIMENSIONS, uses wavelet compression to provide better support for observing, analyzing and querying distributed sensor data at multiple resolutions, as well as exploiting spatio-temporal correlations. Typical examples include micro-climate monitoring. The goal of the system is to provide for flexible spatio-temporal querying and mining of sensor data with the ability to drill down on details, while preserving the capability for long-term data mining within the constraints of node storage. These goals can be achieved by exploiting redundancy of the data, the rarity of interesting features, the scale of sensor networks, and the low cost of approximate queries.

Ganesan was asked how he deals with bad sensors. He responded that wavelets are reasonably good for these things, and that you will lose some information anyway. In response to a question about what fraction of the sensors can fail, co-author Deborah Estrin said the work is too recent to say for sure. When asked about how good wavelets are at summarizing, Ganesan said they have two useful features: lossy compression and the fact that they handle general, rather than specific features. He also said they are good for many types of data, especially images, sequences, and time/frequency series.

Jeremy Elson began his talk by questioning if time synchronization really matters. For the Internet, the answer is "sometimes". However, for sensornets, time synchronization is usually a fundamental requirement because they often make time-dependent measurements of physical events. He next questioned whether the problem has already been solved by NTP, 802.11 sync, GPS, WWVB, or high-stability oscillators. He pointed out that the major difference between the Internet and sensornets is the assumption about energy. This means that sending and listening for packets, or operating the CPU is not free. Elson also pointed out that NTP relies on other infrastructure such as GPS to supply out-of-band time information, and that GPS doesn't work indoors or on Mars. GPS is also expensive and rather big. In response to this problem, his approach was to develop a palette of methods, which each application using the most appropriate method. For example, many times one need not worry about having a single global time reference. Instead, each node just needs to know how its clock is related to neighboring nodes. As another example, one could use "post-facto" synchronization: start the system unsynchronized, use an interesting event recorded by all the nodes as a reference, and later line up the timescales to this event. The benefit is that it avoids wasting energy performing the synchronization if it isn't necessary for the operation of the experiment.

Craig Partridge asked what the government uses for underwater time synchronization of their detection networks. Elson didn't know. Partridge then asked how one reconciles a multi-solution approach with the idea that extra solutions add complexity. Elson responded that sensornets can't afford the cost of general solutions.

Badri Nath presented the final talk, arguing that source-addressed routing won't scale for sensornets because the headers will get too big. By describing the routing direction as a mathematical curve and choosing next-hops based on whatever node is closest to the curve in the desired routing direction, one can keep the header small. Part of the reason this can work is that nodes in sensornets often follow some physical topology (e.g., along a river bank) so that "direction" has a well-defined physical meaning. An additional benefit to this approach is that because specific nodes are no longer named, this routing is better able to tolerate node failures. Nath then presented an example of "spoke flooding" where packets are sent along rays from a source node. With only 40% of the communication that standard approaches us, 80% of the nodes can be reached.

When asked if it is possible to reach a particular node, Nath responded by saying yes, if it is within one hop of the trajectory. If not, the node is not reached and it is considered a routing failure.

During a session-wide discussion, David Wetherall commented that these solutions all seem very application-specific. Nath said that his initial domain is a sensor net where the nodes are cars. Elson's domain is environmental monitoring. Traditionally this monitoring is very coarse grained. The scientists he works with are excited at the potential of getting very fine grained data. Craig Partridge observed that it is common to create lots of point solutions in a new space, and wondered if something will gel out, or will there be several solutions? Elson didn't know, but said that by their nature, sensornets are very application-specific. He allowed for the possibility that we can reuse techniques and generalize. In response to a question about the average battery life, Ganesan said months (days if operating at full throttle), and that the goal is one year. Elson pointed out that one could deploy a heterogeneous set of nodes. Some with "bigger" batteries, and that you then set up a hierarchy similar to a memory cache hierarchy.

Acknowledgments

This material was collected by the dedicated efforts of the HotNets I scribe team: Sergey Gorinsky, Scott C. Karlin, Ratul Mahajan, Akihiro Nakao, Dragos Niculescu, Tammo Spalink, and Limin Wang.

References

[1] Computer Science and Telecommunications Board, US National Research Council. *Looking Over the Fence at Networks: A Neighbor's View of Networking Research*. National Academy Press, 2001. `http://www.cstb.org/pub_lookingover.html`

[2] SIGCOMM Technical Advisory Committee. Improving Sigcomm: A few straw proposals, July 2001. `http://www.acm.org/sigcomm/admin/July2001RepFinal.pdf`

OverQoS: Offering Internet QoS Using Overlays

Lakshminarayanan Subramanian* Ion Stoica* Hari Balakrishnan† Randy H. Katz*

Abstract

This paper proposes *OverQoS*, an architecture for providing Internet QoS using overlay networks. OverQoS empowers third-party providers to offer enhanced network services to their customers using the notion of a *controlled loss virtual link* (CLVL). The CLVL abstraction bounds the loss-rate experienced by the overlay traffic; OverQoS uses it to provide differential rate allocations, statistical bandwidth and loss assurances, and enables explicit-rate congestion control algorithms.

1. Introduction

There has been a growing demand for Internet QoS over the past decade. Several research efforts have addressed the problem of enhancing the best-effort service model to provide QoS, resulting in the Intserv and Diffserv architectures [2, 3]. These, and other proposals for Internet QoS, have two key requirements: first, they require *all* routers along a path to implement QoS mechanisms for scheduling and buffer management, and second, they require the right incentives for Internet Service Providers (ISPs) to enable these functions. Unfortunately, these two requirements have often turned out to be difficult to meet, and despite much research, the Internet largely continues to provide only the basic best-effort service model.

Over the past few years, overlay networks have emerged as an alternative for introducing new functionality that is either too cumbersome to deploy in the underlying IP infrastructure, or that requires information that is hard to obtain at the IP level. Examples of overlay networks include application-layer multicast [5, 8], Web content distribution networks, and resilient overlay networks (RONs) [1]. Motivated in part by the positive results of these approaches for specific network services, we seek to investigate if an overlay network can do the same for Internet QoS.

To motivate why an overlay might lead to a promising QoS architecture, consider the following *third-party QoS provider* model. In this model, a provider buys network access from several traditional ISPs and places nodes in different routing domains. These nodes form an overlay network, which the third-party provider uses to offer enhanced network service to its customers. Another example would be an organization that uses overlays to provide enhanced services in its Virtual Private Network.

Of course, this idea isn't useful unless the third-party provider can demonstrate enhanced services. This paper describes a system called *OverQoS* that shows that an overlay network can indeed provide *certain forms* of QoS. An important aspect of OverQoS is that it does not mandate *any* changes to the data or control planes of the IP routers between OverQoS nodes, instead placing all the QoS machinery at the OverQoS nodes.

When QoS is provided at the IP layer, an IP router has total control over its packet buffers and the output link bandwidth, and can directly schedule these resources. In contrast, an OverQoS node controls neither the bandwidth nor the losses on the underlying IP path. However, *if* each OverQoS router handles a flow aggregate, transmitting the aggregate at some fair rate across the *virtual link* (the underlying IP path) connecting two OverQoS routers, then it can allocate the resources available to the flows *within* the aggregate by controlling the rates of each flow on the virtual link.

While this allows flows within an aggregate to achieve proportional sharing, it does not provide any assurances on achieved rates or observed loss rates. To address this, we develop the notion of a *controlled loss virtual link* (CLVL), which ensures that as long as the aggregate rate does not exceed a certain value, the loss rate observed by the aggregate is very small. Exposing the CLVL abstraction to a flow aggregate traversing a virtual link is a powerful one. We argue this by showing that it can be combined with traditional scheduling (e.g., weighted fair queuing) and buffer management schemes running at the OverQoS nodes to provide service differentiation. We also show that it can provide approximate statistical assurances when used in conjunction with adaptive admission control schemes (e.g., where flows can periodically renegotiate their transmission rates), and discuss how a CLVL abstraction enables new explicit-rate end-to-end congestion control algorithms.

2. OverQoS Architecture

This section describes the OverQoS network architecture (Figure 1). A *virtual link* is the underlying IP path connecting two overlay nodes. A virtual link is unidirectional and carries traffic from an *entry overlay node* to an *exit overlay node*. A *bundle* is the stream of application data packets carried across the virtual link; it typically includes packets from multiple transport-layer flows across different sources and destinations.

In general, a virtual link is characterized by a capacity b and a loss rate p. (We don't focus on delay assurances in this paper.) The capacity b represents the maximum rate at which the entry OverQoS node can send traffic over the virtual link, while the loss rate p represents the probability that a packet is dropped on the virtual link due to congestion. In practice, we expect b to be either determined based on some fairness criterion or obtained from a contract

*EECS Department, University of California at Berkeley, emails: {lakme,istoica,randy} @cs.berkeley.edu

†Laboratory of Computer Science, Massachusetts Institute of Technology, hari@lcs.mit.edu

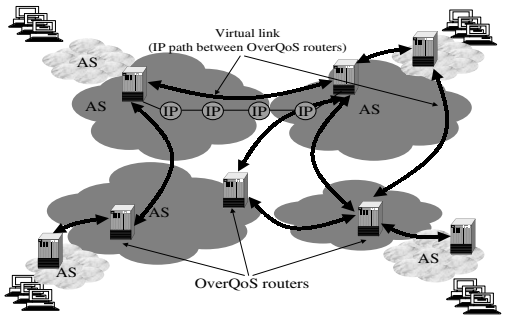


Figure 1: The OverQoS system architecture. OverQoS routers in different AS's communicate with each other over virtual links using the underlying IP paths.

agreement with the administrators (ISPs) of the underlying network. One way of providing fairness is to set b based on an *N-TCP pipe* abstraction. This abstraction provides a bandwidth which is N times the throughput of a single TCP connection on the virtual link. N may be negotiated between the OverQoS provider and the ISPs, or be picked to be the number of flows in the bundle (see Section 3).

Three constraints make the design of the mechanisms at the OverQoS nodes challenging:

1. OverQoS nodes will usually span different routing domains and AS's.
2. The OverQoS nodes will usually not be directly connected to the congested links.
3. In general, some (or most) of the traffic traversing the congested links of the IP path between two overlay nodes will not be part of the OverQoS bundle.

As a result, OverQoS needs to handle time-varying cross-traffic and network conditions that it has no control over, and yet enhance the service quality of the bundle.

OverQoS is based on two fundamental design principles: a) loss control; b) aggregate resource control. Loss control enables OverQoS to obtain a minimum service quality irrespective of the varying network conditions. By aggregating flows into a bundle, OverQoS can exercise complete control in distributing the available resources (bandwidth, loss) for the bundle amongst the individual flows.

2.1 Controlled-Loss Virtual Link (CLVL)

To enable OverQoS to provide better than best-effort services, we propose a new abstraction, *controlled-loss virtual link (CLVL)*, to characterize the service received by a bundle. Using mechanisms implemented at the entry and exit nodes, a CLVL provides a bound, q, on the loss rate seen by the bundle over a certain period of time regardless of how the underlying bandwidth b and loss rate p vary in time. The idea is that a CLVL *isolates* the losses experienced by the bundle from the loss-rate variations in the underlying IP network path.

One way to control the virtual link loss rate is to add redundancy packets to the bundle. Forward error correction (FEC) and automatic repeat request (ARQ) are two ways to do this. While ARQ has a lower bandwidth requirement than FEC, ARQ may need more time to recover depending on the RTT between the overlay nodes and the number of retransmissions. In Section 4.2, we present a hybrid FEC/ARQ solution.

The traffic between two overlay nodes consists of the flows in the bundle and redundancy traffic for loss recovery. If r represents the amount of redundancy required to achieve a target loss-rate q, the *available bandwidth* for the flows in the bundle is $c = b(1 - r)$. This leads to the definition of the CLVL service model: *As long as the arrival rate of the bundle at the entry node does not exceed c, the packet loss rate across the virtual link will not exceed q, with high probability.*

2.2 Aggregate Resource Control

The CLVL abstraction provides the service on a bundle aggregate, rather than on a per-flow basis. This has two benefits: First, the entry node has *control* over how the resources of the aggregate are distributed among the individual flows in the bundle. Second, applying FEC for loss-control on an aggregate is more efficient than on a per-flow basis. The larger the number of packets within any time window, the lower the FEC overhead [10].

The entry node exerts control on the traffic in the bundle at two levels of granularity: on the bundle as a whole, and on a per-flow basis within the bundle. At both these levels, the entry node can control either the sending rate or the loss rate. The entry node first determines the virtual link's underlying parameters, b and p. Next, it determines the level of redundancy r required to achieve a certain target loss-rate q and estimates the resulting available bandwidth c. The entry node then distributes the bundle's available bandwidth c among the individual flows. If the net input traffic is larger than c, the extra traffic is dropped at the entry node and the losses are distributed across the flows in the bundle according to their service specifications.

In the next section, we provide some of the potential benefits to an end-user for using an OverQoS architecture as opposed to just using the Internet. Section 4 discusses how a CLVL can be implemented. Section 5 gives examples of how the CLVL abstraction can be used to provide enhanced services.

3. Why Use OverQoS?

In this section, we try to answer the following question: *Can OverQoS provide enhanced service to all OverQoS flows without negatively affecting the background traffic?* If "yes", we would have a strong case for using OverQoS.

More precisely, we want to know whether there are OverQoS solutions that satisfy the following constraints:

1. Any OverQoS user should get a service no worse than using the best-effort Internet. Otherwise a user won't have any incentive to use OverQoS.

2. OverQoS should not penalize the background (best-effort) traffic. Ideally, we would like a best-effort flow to receive roughly the same throughput irrespective of how many other flows (that share the same congested link) use OverQoS. This way, an ISP won't have negative incentives not to support OverQoS traffic.

It is not immediately clear that it is possible to simultaneously satisfy both constraints. Consider n flows traversing a congested link. We consider two scenarios, (a) all n flows are best-effort, and (b) m flows belong to a CLVL, and the rest of

$n - m$ flows are best-effort. Assume that all best-effort flows are TCPs. We call the throughput achieved by a flow in scenario (a) the *TCP-equivalent throughput* of that flow. Then constraint (2) can be rewritten as: each background flows should achieve no less than its TCP-equivalent throughput no matter how many flows belong to the CLVL. From here it follows that the aggregate bandwidth of a CLVL, b, should not exceed the sum of the TCP-equivalent throughputs of the m flows that belong to the CLVL. However, this implies that if a CLVL flow achieves more than its TCP-equivalent throughput, there is at least another CLVL flow that achieves less than its TCP-equivalent throughput. But this apparently violates constraint (1) since the user whose flow achieves less than its TCP-equivalent throughput may get a better service by switching to the best-effort Internet!

Fortunately, this is not the case. The problem with the above argument is that we have implicitly equated the service received by a user with the throughput of his flows. However, a user is not always interested in optimizing the throughput of each of his flows. We illustrate this point with three examples in which OverQoS provides enhanced services to users while still satisfying the above constraints.

Trading throughput for loss-rate: For some users it is more important to achieve a low loss rate than maximizing the throughput of their flows. For instance, if the TCP-equivalent throughput of a flow were 100 Kbps, a user using a voice-over-IP application would be happy to get only 64 Kbps as long as the loss rate is less than say 0.1%. The reason for using OverQoS in this case – instead of simply using per-flow FEC – is that per-aggregate FEC has a lower overhead in terms of redundant traffic than per-flow FEC.

Spatial bandwidth redistribution: Given a choice, many users would like to have control on their aggregate traffic. For instance, a user that has multiple flows may want to improve the throughput of his "important" flows at the expense of those less "important". Consider a user that has two flows in the same CLVL where the TCP-equivalent throughput of a flow is 0.5 Mbps. In this case, the user should be able to redistribute the total of 1 Mbps among its two flows as he wishes. This functionality can be easily achieved by using hierarchical link sharing [14]. Note that in today's Internet, a user cannot achieve this; unless the congestion is on the outgoing link, reducing the throughput of one flow will not result in increased throughput for the other flows.

Temporal bandwidth redistribution: A user may want to reduce the completion times of short flows if this won't impact the completion times of long flows. Such a service could significantly improve the web browsing experience since the majority of web transfers consist only of a few data packets [7]. To illustrate the feasibility of such a service, consider the example in Figure 2 in which a CLVL with a bandwidth of 1 Mbps is shared by one long flow that transfers 200 Kb, and four short flows that transfer 50 Kb each. The long flow starts the transfer at time 0, while short flows start their transfers at times 0, 0.1, 0.2, and 0.3 sec respectively. Figure 2(a) shows the case when all flows receive an equal share of the CLVL's bandwidth. This accounts for the case when the entry node runs a simple FIFO scheduler, and all flows use the same congestion control scheme and the have the same RTT. As a result, the long flow finishes its transfer in 0.4 sec, while all short flows complete their transfer in 0.1 sec. In contrast, Figure 2(b) shows the case when the entry node runs a per-flow scheduling algorithm

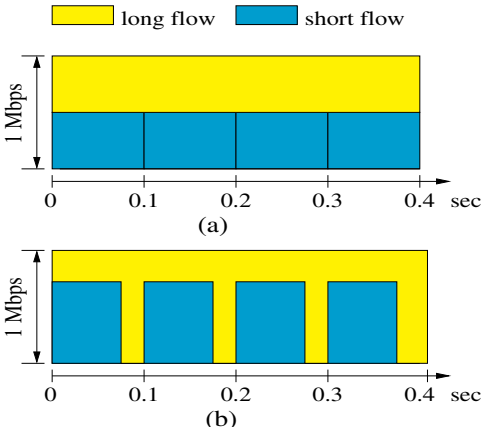


Figure 2: Improving short flow completion times. (a) Short and long flows split equally the available bandwidth. (b) Short flows get 3/4 of the available bandwidth. The completion time of short flows decreases to 0.066 sec; the completion time of the long flow remains unchanged.

that gives the short flows 3/4 of the available bandwidth. As a result, each short flow completes the transfer in only 0.066 sec. The important point to note is that this improvement does not affect the long flow; the long flow still completes its transfer in 0.4 sec. A scheduling algorithm that can implement this service without prior knowledge of the flow lengths is presented in [11].

In summary, OverQoS can indeed provide better services to its users without unduly affecting the background traffic.

Finally, note that in practice there are cases in which it makes sense to violate constraint (2). In particular, an ISP may choose to allocate more bandwidth to a OverQoS provider at the expense of the background traffic as long as this is justified by the price structure of the best-effort and the OverQoS services. For instance, an ISP can offset the decrease in the throughputs of the background flows by reducing the price per bit for this traffic, and recoup the difference by correspondingly pricing the bandwidth used by the OverQoS provider. In turn, the OverQoS provider can ask its customers to pay a higher price in exchange for better service.

4. Implementing CLVLs

This section describes two different ways of building CLVLs: a pure FEC-based solution and a hybrid solution which is a combination of FEC and ARQ. Recall that a CLVL abstraction aims to bound the bundle loss rate to $q \ll p$. Since burstiness of cross-traffic is usually unpredictable, we define q as a statistical bound on the average loss rate observed over some larger period of time (on the order of seconds).

A purely ARQ-based solution for building CLVLs is easy to construct. In a reliable transmission ($q = 0$), a packet is repeatedly retransmitted until the sender receives an acknowledgment from the receiver. In contrast, to achieve a non-zero target loss rate, q, it is enough to retransmit any lost packet at most $L = \log_{\bar{p}} q$ times, where $\bar{p}$ represents the average loss rate over the interval over which we want to bound q.

4.1 FEC-based CLVL construction

In an FEC-based approach, we divide time into windows, where a window is a unit of encoding/decoding. We consider an *erasure code* such as Reed-Solomon, characterized by (n, k), where k is the number of packets arriving at the entry node during the window, and $(n - k)$ represents the number of redundant packets added. Define the redundancy factor, $r = (n - k)/n$. The FEC problem reduces to determining a minimum redundancy factor, r, such that the target loss rate q is achieved.

This problem is challenging because packet losses are unpredictable, and the algorithm must handle drastic changes in loss rate and correlated packet losses. Since the value of q may be one or two orders of magnitude smaller than p, we may not be able to afford to wait for feedback from the receiver about bursty losses in a window. Also, the time period of a burst may be comparable to the time for obtaining feedback from the receiver. So, rather than trying to predict the occurrence and magnitude of the next burst, we follow a conservative approach: We compute a statistical bound on the fraction of packets lost in a window due to bursts based on past history and set the redundancy factor to this bound. A burst induces unrecoverable losses in a window if the fraction of packets lost outnumber the redundancy factor. We calculate this bound such that the net losses caused by such bursts is less than q.

More precisely, let $f(p)$ denote the PDF of the loss rate p, where each value of p is measured over an encoding/decoding window. Then, for a given target loss rate q, we need to compute the smallest r such that:

$$\int_r^1 pf(p)dp \leq q. \quad (1)$$

Computing r requires the knowledge of the distribution $f(p)$. In practice, we estimate $f(p)$ as follows. The OverQoS exit node for a bundle computes the loss rate p for each window and sends it back to the entry node. In turn, the entry node uses these samples to construct a histogram, and then uses this histogram to estimate $f(p)$. Finally, the entry node computes r for the next window based on the estimated $f(p)$.

It turns out that our algorithm requires $2/q$ loss samples to accurately estimate r for a given value of q. Since we compute the histogram only using the last $2/q$ samples, we require the stationary property to hold only over relatively short time periods (of the order of minutes).

4.2 FEC+ARQ based CLVL construction

While the FEC based solution is relatively easy to implement, it can incur a high overhead when the loss rate p is bursty (e.g., when $f(p)$ is heavy-tailed). To reduce this overhead, we outline a hybrid FEC/ARQ approach that extends the previous FEC solution.

Due to delay constraints for loss recovery, we restrict the number of retransmissions to at most one. We divide packets into windows and add a redundancy factor of r_1 for each window in the first round. In the second round, if a window is non-recoverable, the entry node retransmits the lost packets with a redundancy factor r_2.

We need to estimate the parameters, r_1 and r_2. As in the previous case, let $f(p)$ model the fraction of packets lost in a given window. The expected packet loss rate after two rounds is equal to $G(r_1) \times G(r_2)$ where:

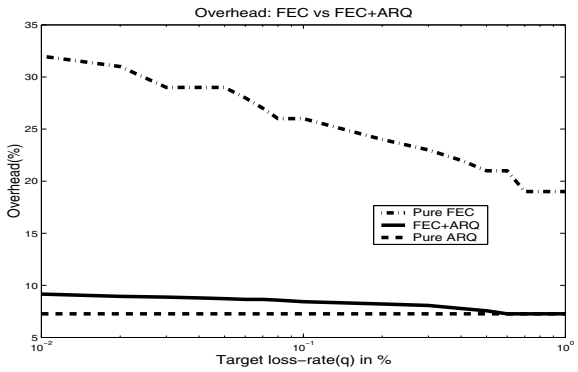


Figure 3: Overhead (r): FEC+ARQ vs Pure FEC. In both cases $b = 2Mbps$ and the bottleneck link is $10Mbps$ with a $9Mbps$ self similar background traffic.

$$G(r) = \int_r^1 pf(p)dp. \quad (2)$$

The expected overhead, O, is simply $r_1 + G(r_1)(1 + r_2)$. This yields the following optimization problem: Given a target loss rate q, determine the redundancy factors r_1 and r_2 that minimize the expected overhead, $O = r_1 + G(r_1) \times (1 + r_2)$, subject to the target loss constraint: $G(r_1) \times G(r_2) \leq q$.

Fortunately, for many loss distributions that occur in practice, the optimal solution for this problem is when $r_1 = 0$. This solution implies that it is better not to use FEC in the first round, and use FEC only to protect retransmitted packets.

Figure 3 compares the overhead characteristics for FEC+ARQ with pure-FEC and pure-ARQ based approaches. We make two observations. First, the overhead of the FEC+ARQ algorithm is much smaller than that of the pure-FEC algorithm. This is because, FEC+ARQ applies FEC only to the retransmitted packets, and the number of retransmitted packets is much smaller than the total number of packets. Second, when $q \geq p_{avg}^2$, the FEC+ARQ algorithm reduces to the pure-ARQ algorithm, where $r_1 = r_2 = 0$. This is because in this case each lost packet is retransmitted only once; this is enough to achieve a target loss-rate $\leq p_{avg}^2$.

While FEC+ARQ is more efficient than pure-FEC, FEC+ARQ may require more time to recover. With the pure-FEC algorithm, the worst-case recovery time is W, where W is the length in time of the encoding/decoding window. In contrast, with the FEC+ARQ algorithm it may take $RTT+W_1+W_2$ time to recover from losses, where RTT is the round-trip time between the entry and the exit node, and W_1 and W_2 are the sizes of the windows corresponding to rounds 1 and 2. For the simulation results shown in Figure 3, the values of RTT, W and W_1 are set to 100ms each. The value of W_2 depends on the number of retransmitted packets in a window which in turn is dependent on the loss-rate experienced by the window (worst-case: $W_2 = W_1$, average-case: $W_2 = p_{avg} \times W_1$).

5. Examples of Using CLVLs

In this section, we discuss three concrete examples of how CLVLs can be used by OverQoS to provide different types of services for end-to-end flows. The three examples we consider are: per-flow bandwidth differentiation, statisti-

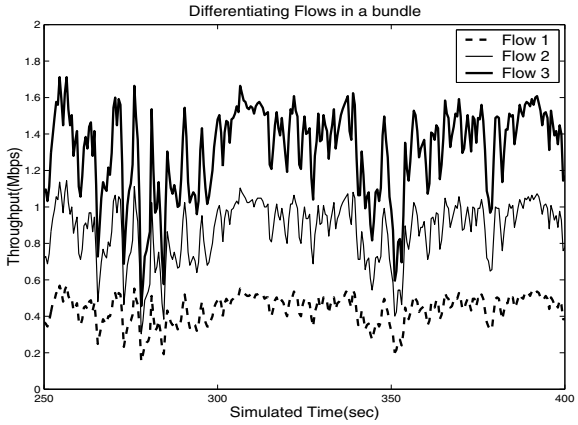


Figure 4: Differential rate allocation for three classes within a bundle in the ratio $1:2:3$.

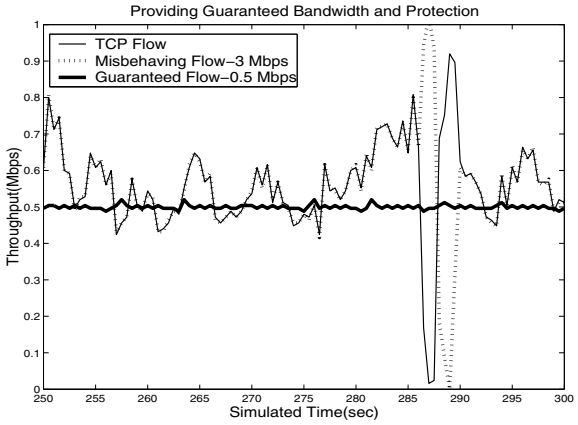


Figure 5: Providing statistical bandwidth guarantees to QoS flows and protection to available-bandwidth ones.

cal bandwidth guarantees, and explicit-rate end-to-end congestion control algorithms. All these examples represent different ways in which the entry node distributes the CLVL available bandwidth, c, among the competing flows.

5.1 Differential Rate Allocation

A simple way of distributing c among the flows is to allocate the bandwidth in a proportionally fair manner. We can implement a Diffserv-like service by separating OverQoS flows into different classes and proportionally distributing c among the different classes. Figure 4 demonstrates this service for three different classes of traffic across a single CLVL. In this simulation, the bundle bandwidth is allocated in the ratio $1:2:3$ using a DRR scheduling discipline [12]. Here, we assume a 10 Mbps link, and a 6 Mbps self similar background traffic. This ratio is strictly preserved even in the presence of a varying bundle bandwidth b (as estimated by N times the rate of a single TCP flow). While this example illustrates the differentiation across a pair of OverQoS nodes, we can easily extend the same model across the entire path in the overlay network.

5.2 Statistical Rate and Loss Assurances

OverQoS can provide statistical rate and loss assurances that may be useful for streaming media delivery.

If the net arrival rate at the entry node is less than c, then the entry node doesn't have to drop any packets. Since c itself varies in time, it may not always be possible to avoid losses at the entry node. If the values taken by c can be modeled by a distribution, we can estimate a value c_{min}, such that $P(c < c_{min})$ is small. c_{min} is a stable value as long as the underlying distribution of c is stationary. If c_{min} is non-zero, the entry node can admit flows with pre-specified bandwidth requirements such that the net bandwidth requirement is less than c_{min}. Since the net arrival rate of these flows is less than c with high probability, the CLVL abstraction can provide these flows with both statistical loss and bandwidth guarantees. We refer to these flows as *QoS flows*. Of course, we need an admission control module at the entry overlay node to allocate the bandwidth c_{min} amongst the QoS flows. Also, these flows can be admitted only over the period for which c_{min} is stable and fixed (typically of the order of minutes), and flows may renegotiate admission.

The remaining part of the available bandwidth can be distributed amongst the other *available-bandwidth* flows in the bundle.

We illustrate the CLVL's capability to provide this service model using a simple simulation. Consider a virtual link between two overlay nodes running a CLVL bundle for a target loss rate of $q = 0.1\%$ on top of an N-TCP pipe with $N = 10$. The virtual link traverses a bottleneck link of 10 Mbps and the cross-traffic comprises 50 long-lived TCP flows. By observing samples of c, the entry node determines $c_{min} = 0.5$ Mbps since $P(c < 0.5\text{Mbps})$ is negligible. The entry node of the bundle implements the DRR scheduling discipline. The bundle consists of three flows: (1) a QoS flow requiring a bandwidth guarantee of 0.5 Mbps, (2) a 3 Mbps CBR flow, and a (3) TCP flow. Flows 2 and 3 are available-bandwidth flows.

Figure 5 plots the average bandwidth achieved by the three flows as a function of time. Flow 1 receives its guaranteed bandwidth with a loss-rate of only 0.02%. This is two orders of magnitude lower than the network loss rate of 2.44%. The TCP flow is protected against the aggressive 3 Mbps CBR (both flows have the same weight). Furthermore, the TCP flow in the bundle receives more bandwidth than a regular TCP since it experiences a lower end-to-end loss rate. Finally, when the TCP flow experiences a timeout, flow 2 takes advantage of this and uses the excess bandwidth. None of this requires any QoS machinery in the IP routers on the virtual link; all the required functionality is implemented exclusively at the overlay nodes.

5.3 Explicit-rate Congestion Control

An end-to-end path obtained by "stitching together" a sequence of CLVLs enables new end-to-end congestion control algorithms without IP router support. For simplicity, consider the case when all end-to-end flows traverse exactly one CLVL. Since the entry node knows c at any point in time, it can decide how to allocate c amongst the currently active flows in the bundle, ensuring that each c_i allocated to flow i satisfies the constraints that $c_i \leq a_i$ (the arrival rate of flow i) and $\sum_i c_i = c$. Then, by providing this information to each flow as feedback analogous to XCP [9], cooperating end-hosts can send at rate c_i, a rate that will not cause more

than a small number of observable losses. One can extend this to flows that traverse multiple CLVL's by setting the sender's flow transmission rate to the minimum of the c_i's returned along the reverse path.

6. Discussion

We now discuss a few important aspects of the OverQoS design: The benefits of overlays, the power of controlled-loss and scalability issues.

Why overlays? Overlays are easy to deploy compared to changes to the IP infrastructure. More importantly, they empower third-party entities other than traditional ISPs to offer enhanced communication services to clients. Similarly, OverQoS enables enterprises to build their own VPNs to provide communication services superior to the ones offered by traditional ISPs.

The key to provide better services in OverQoS is the CLVL abstraction. CLVL allows applications to provide per-flow bandwidth differentiation, statistical rate assurance, and implement new congestion control algorithms without any QoS support in routers. There are two properties of CLVL that make it possible to implement these services in an overlay network: the ability to control the loss rate, and traffic aggregation.

Controlled-loss: CLVLs achieve a predefined target loss rate, potentially at the expense of a reduction in the bundle's throughput. This is different from other similar QoS mechanisms (e.g., Diffserv's Assured Forwarding class) that provide a fixed-bandwidth abstraction but without any loss guarantees. There are two advantages of providing a controlled-loss abstraction.

First, a controlled-loss abstraction gives more flexibility to the entry router in allocating a bundle's resources among flows in a situation where no form of admission control is present. If the available bandwidth decreases, the entry node can choose to protect "important" flows by dropping the packets of the "less-important" ones. Second, since the loss seen by a bundle is in general much lower than the loss in the underlying network, we can more readily deploy new explicit-rate congestion control algorithms for flows within the bundle.

Scalability: Scalability is an important concern in OverQoS: we look at the amount of state, the FEC overhead, and the number of OverQoS bundles below.

Traditional solutions to provide fine-granularity services require to perform per-flow buffer management, scheduling, and eventually admission control. For very large bundles maintaining and managing the state for each flow in the bundle may not be feasible. To get around this problem we can use scalable techniques that were proposed at the IP layer to enhance Internet's QoS, such as end-host based admission control [4], or dynamic packet state (DPS) [13].

There are two components of the FEC overhead: communication and processing. The communication overhead scales well with the bundle's rate. In fact, the percentage of redundant traffic *decreases* as the rate of the bundle increases. This is because the number of packets sent during the same time window increases with the rate of the bundle. Our current untuned implementation can process 200 Mbps of FEC traffic on a 866 MHz Pentium III. Furthermore, in-order arrival of packets is well-suited to an implementation using pipelined, high-bandwidth FEC ASICs for Reed-Solomon codes. Such ASICs are commercially available [6].

In practice, we expect multiple OverQoS networks to coexist, an important question concerns the effect of multiple CLVLs sharing the same congested link. This question remains open, but our preliminary simulation results indicate that the N-TCP abstraction allows any number of CLVLs to seamlessly coexist and share the bandwidth of a congested link. In addition, our simulation results indicate that a bundle using N-TCP abstraction is also fair to the background TCP traffic.

References

[1] D. Andersen, H. Balakrishnan, M. Kaashoek, and R. Morris. Resilient Overlay Networks. In *Proc. 18th ACM SOSP*, pages 131–145, Banff, Canada, Oct. 2001.

[2] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An architecture for differentiated services, Oct. 1998. Internet Draft.

[3] R. Braden, D. Clark, and S. Shenker. Integrated services in the Internet architecture: An overview, June 1994. Internet RFC 1633.

[4] L. Breslau, E. W. Knightly, S. Shenker, I. Stoica, and H. Zhang. Endpoint admission control: Architectural issues and performance. In *Proc. of ACM SIGCOMM'00*, pages 57–69, Stockholm, Sweden, Sept. 2000.

[5] Y. Chu, S. G. Rao, S. Seshan, and H. Zhang. Enabling conferencing applications on the Internet using an overlay multicast architecture. In *Proc. of ACM SIGCOMM 2001*, San Diego, CA, Aug. 2001.

[6] Advanced Hardware Architectures. `http://www.aha.com/`.

[7] A. Feldmann, A. C. Gilbert, and W. Willinger. Data networks as cascades: Investigating the multifractal nature of internet wan traffic. In *Proc. of ACM SIGCOMM 1998*, Vancouver, Canada, Aug. 1998.

[8] J. Jannotti, D. K. Gifford, K. L. Johnson, M. F. Kaashoek, and J. O'Toole. Overcast: Reliable multicasting with an overlay network. In *Proc. USENIX OSDI*, San Diego, CA, Oct. 2000.

[9] D. Katabi, M. Handley, and C. Rohrs. Internet congestion control for future high bandwidth-delay product environments. In *Proc. of ACM SIGCOMM 2002*, Pittsburg, PA, Aug. 2002.

[10] S. Lin and D. Costello. Error control coding: Fundamentals and applications. In *Prentice Hall*, New York, NY, Feb. 1983.

[11] T. S. E. Ng, D. Stephens, I. Stoica, and H. Zhang. Supporting best-effort traffic with fair service curve. In *Proc. of GLOBECOM 1999*, Rio de Janeiro, Brazil, Dec. 1999.

[12] M. Shreedhar and G.Varghese. Efficient fair queueing using deficit round robin. In *Proc. of ACM SIGCOMM 1995*, Cambridge, MA, Aug. 1995.

[13] I. Stoica. *Stateless Core: A Scalable Approach for Quality of Service in the Internet*. PhD thesis, Carnegie Mellon University, dec 1995. CMU-CS-00-176.

[14] I. Stoica, H. Zhang, and T. S. E. Ng. A hierarchical fair service curve algorithm for link-sharing, real-time and priority service. In *Proc. of ACM SIGCOMM 1997*, Cannes, France, Aug. 1997.

From Protocol Stack to Protocol Heap – Role-Based Architecture

Robert Braden
USC Information Sciences Institute
4676 Admiralty Way
Marina del Rey, CA
Braden@isi.edu

Ted Faber
USC Information Sciences Institute
4676 Admiralty Way
Marina del Rey, CA
Faber@isi.edu

Mark Handley
International Computer Science Institute
1947 Center St, Suite 600
Berkeley, CA 94704
mjh@icir.org

ABSTRACT

Questioning whether layering is still an adequate foundation for networking architectures, this paper investigates non-layered approaches to the design and implementation of network protocols. The goals are greater flexibility and control with fewer feature interaction problems. The paper further proposes a specific non-layered paradigm called *role-based architecture*.

Categories and Subject Descriptors

C.2.2 [**Computer-Communication Networks**]: Network Protocols—*Protocol Architecture*

Keywords

Non-layered architecture, role-based, modularity, metadata, signaling, processing rules

1. INTRODUCTION

Traditional packet-based network architecture assumes that communication functions are organized into nested levels of abstraction called *protocol layers* [7], and that the metadata that controls packet delivery is organized into *protocol headers*, one for each protocol layer [4].

Protocol layering has served well as an organizing principle, but it worked better for the more strict end-to-end model of the original Internet architecture than it does today. We see constant pressure for "layer violations" (which are often assumption violations), and unexpected feature interactions emerge. In part this is due to the rapid proliferation of "middle boxes" (firewalls, NAT boxes, proxies, explicit and implicit caches, etc.), but other multi-way interactions such as QoS, multicast, overlay routing, and tunneling are also guilty.

The complex interactions that result are difficult to describe using strict layering, and the implicit "last on, first off" assumption of layering often makes a new service fit poorly into the existing layer structure. The result is an inability to reason about feature interaction in the network. A reluctance to change working implementations and long-standing inter-layer interfaces often lead designers to insert new functionality between existing layers rather than modify existing layers.[1] In addition, it is very hard to evolve network protocols, especially at the network level. Partly this is for performance reasons[2], but partly it is because layering tends to lead to a relatively coarse granularity of protocol functionality.

The limits of layering are clear from even simple examples. There is currently no way for a TCP SYN packet to port 80 (HTTP) to signal that it does not want to be redirected to an application-layer web cache. Also, network congestion is signaled by a router (network layer), but rate-control occurs at the flow level (transport layer), so signaling between the flow and the network is difficult. In both cases, layering is an important part of the problem.

These considerations suggest that layering may not be a sufficiently flexible abstraction for network software modularity. This inflexibility might be considered desirable, as it forces compliance with existing standards, but in practice it often results in short-sighted solutions that may violate assumptions made by other protocols.

If protocol layering is inadequate as an abstraction, we need an alternative organizational principle for protocol functionality. Our task is to allow more flexible relationships among communication abstractions, with the aim of providing greater clarity, generality, and extensibility than the traditional approach allows. This paper proposes a non-stack approach to network architecture that we call *role-based architecture* or RBA.

1.1 Role-based Architecture

Instead of using protocol layers, an RBA organizes communication using functional units called *roles*. Since roles are not generally organized hierarchically, they may be more richly interconnected than are traditional protocol layers. The inputs and outputs of a role are application data payloads and controlling metadata that is addressed to specific roles.

With a non-layered approach, layer violations should be replaced by explicit and architected role interactions. Of course, "role violations" will still be possible, but the generality of the mechanism should typically make them unnecessary, and suitable access controls over metadata can make them difficult.

HotNets I, Princeton, NJ, USA, October 2002. This research was sponsored in part by DARPA contract F30602-00-1-0540 and by AT&T Research.

[1] For example, MultiProtocol Label Switching was inserted at "layer 2.5", IPsec at "layer 3.5", and Transport-Layer Security at "layer 4.5".

[2] For example, IPv4 options are rarely used.

The intent is that roles will be building blocks that are well-defined and perhaps well-known. To enable interoperability, a real network using RBA would need a relatively few (tens to hundreds) of *well-known* roles defined and standardized.[3] However, the number of special-purpose, experimental, or locally defined roles is likely to be much greater.

An important attribute of role-based architecture is that it can provide explicit signaling of functionality. The lack of architected signaling is one of the main reasons why middleboxes do not fit into the current layered architecture. For example, there is no defined way to signal to an end-system that a packet really did traverse the firewall protecting the site, or to signal that an end-system does not want its request redirected to a web cache. RBA is designed to perform such signaling in a robust and extensible manner.

Role-based architecture allows *all* the components comprising a network to be explicitly identified, addressed, and communicated with. RBA could allow allow re-modularization of current "large" protocols such as IP, TCP, and HTTP into somewhat smaller units that are addressed to specific tasks. Examples of such tasks might be "packet forwarding", "fragmentation", "flow rate control", "byte-stream packetization", "request web page", or "suppress caching". Each of these comprise separable functionality that could be performed by a specific role in a role-based architecture.

The purpose of this paper is to examine the general properties of any role-based architecture rather than to describe in detail any particular instance of an RBA. It is intended to suggest a fruitful research direction, which holds some promise for improving the clarity and generality of network protocol design. Since moving to a non-layered architecture requires a distinct shift in thinking, the next section examines the implications of removing layering constraints. Section 2 then outlines what role-based architecture might actually look like in principle and gives some simple examples of the use of RBA. Section 3 describes a range of approaches to applying the RBA ideas in the real world of networking and discusses implementation issues.

1.2 Non-Layered Architecture Implications

The concept of a non-layered protocol architecture has immediate implications. Layering provides modularity, a structure and ordering for the processing of metadata, and encapsulation. Modularity, with its opportunity for information hiding and independence, is an indispensable tool for system design. Any alternative proposal must provide modularity, but also adequately address the other aspects of layering:

Metadata Structure: Without layering, the structure of metadata carried in a packet header no longer logically forms a "stack", it forms a logical "heap" of protocol headers. That is, the packet header is replaced by a container that can hold variable-sized blocks of metadata, and these blocks may be inserted, accessed, modified, and removed in any order by the modular protocol units.

Processing Rules: A non-layered architecture requires new rules to control processing order and to control access to metadata, to replace the rules implicit in layering.

Consider ordering first. In the simplest case when roles are completely independent, a non-layered architecture specifies no processing order; protocol modules may operate in any order, or even simultaneously, on various subsets of the metadata. More commonly, however, an appropriate partial ordering is required among specific roles.

Other rules must specify how access to metadata is to be controlled. By controlling the association between program and (meta)data, the architecture can explicitly control interactions among the different protocol modules, enhancing security as well as extensibility.

Encapsulation: In a layered architecture, each layer is encapsulated in the layer below. In non-layered architectures, a different organizational principle is needed for the data and metadata in a packet. Encapsulation does not disappear, but its role is much reduced. Encapsulation is no longer the main enforcer of processing order. It is reserved for cases where the functionality is that of a container or adaptor, such as the encapsulation of a reliable byte stream within a flow of packets. Even in such cases, metadata about the data being encapsulated need not be itself encapsulated, as it would be in a layered architecture.

1.3 Prior Work

There have been very many papers about different ways to generalize protocol design and processing, to ease the limitations of strictly-layered stacks of complex protocols and of monolithic implementations.[4]

Early work [3, 9] emphasized the modular construction of protocol processing stacks for flexibility and extensibility. Many later papers have discussed the decomposition of complex protocols into *micro-protocols*, either for reusability, customization, and ease of programming [1, 5, 8, 6], or to improve protocol processing performance using parallelism [2, 10]. Here micro-protocols roughly correspond to our roles (as abstractions) or to our actors (as a protocol processing modules); see [1] for example.

Some of these papers suggest generalizations from strict layering, but their primary emphasis is on protocol implementations rather than on the protocols themselves. This paper focuses on the protocols themselves (the distributed algorithms and the "bits on the wire") and on what it means to completely give up layering. Our work has also been aimed at the new architectural issues raised by middleboxes. RBA is a general proposal for achieving modularity through a non-layered protocol paradigm, non-layered in both protocol headers and processing modules.

However, in order to realize an RBA it will be necessary to specify ways to define and structure roles and to design machinery for instantiating and executing roles. Here the prior research will become highly relevant and may provide important answers. We therefore believe that this paper is complementary to much of the earlier work on generalized protocol processing.

2. THE IDEALIZED RBA

In light of the discussion above, we propose *role-based architecture* (RBA) as a particular non-layered architecture in which the modular protocol unit is called a *role*. A role is a functional description of a communication building block

[3]Note that role-based architecture will not remove the need for standardization.

[4]This paper can list only some of the papers that are particularly relevant to RBA.

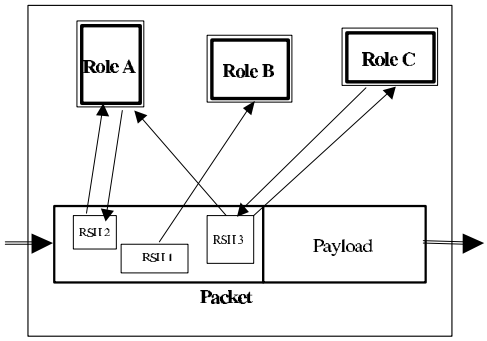


Figure 1: Role-Specific Header Processing within a Node

that performs some specific function relevant to forwarding or processing packets. Roles are abstract entities, and it should be possible to reason about them somewhat formally.

Roles are instantiated in nodes by code called *actors*. For many purposes, the distinction between a role and its actors will not matter, so we can often simply speak of roles as having particular properties. A role may logically span multiple nodes, so it may be distributed in an abstract sense, while each actor executes in a particular node.

The metadata in a packet, called *role data*, is divided into chunks called *role-specific headers* (RSHs). This is illustrated in Figure 1, which shows a node containing three roles that read and perhaps write specific RSHs in a packet header.

Roles are the principal form of addressing in a role-based architecture, in that RSHs are *addressed to roles*. Whether the RSH also specifies which actor should instantiate the role is optional. Typically a role will name some modular functionality or mechanism. As network functionality is often distributed with the actors forming a closely coupled system, sometimes a role will name a distributed mechanism, and sometimes it will name the components of such a mechanism. Only deployment experience can determine what the exact functional breakdown into roles should be.

An end system does not necessarily know that a packet it sends will encounter a node that cares about a particular role. For example, there may be no web-cache-redirector role on a particular path, but if there is, including a signaling RSH addressed to this role will ensure that the cache receives the metadata. Any node along the path can add an RSH to a passing packet. For example, suppose that a firewall has some lightweight way of signing that a packet has been examined and found to conform to the site's security policy; it can include this signature in an RSH attached to the data packet and addressed to the host firewall role.

2.1 RBA Objectives

RBA is designed to achieve the following objectives.

Extensibility: RBA is inherently extensible, both mechanically and conceptually. Mechanically, RBA uses a *(type, length, value)* mechanism to encode all role data. Conceptually, the flexible modularity of RBA should enhance extensibility of networked systems.

Portability: The role abstraction is designed to be independent of the particular choice of nodes in which a role will be performed. This *portability* of roles enables flexible network engineering, since functions can be grouped into boxes as most appropriate. Role portability may, but won't generally, imply role *mobility*, where a role can migrate between nodes.

Explicit Architectural Basis for Middle Boxes: RBA is intended to allow endpoints to communicate explicitly with middle boxes and middle boxes to communicate with each other.

Controlled Access to Metadata: RBA includes a general scheme for controlling access to metadata, allowing control over which nodes can read and modify specific subsets of the metadata as well as the application data. This access control implies control of the services that can be requested and control of the operations that can be performed on a packet.

Auditability: An endpoint may wish to *audit* received data packets to ascertain that they were subjected to requested processing, for example through a validator or an encrypted firewall. Auditability is not generally feasible with the current layered architecture because the relevant metadata will have been processed and removed before the data reaches the receiver. RBA role data can be used to indicate that the requested service was provided.

2.2 General Properties of Roles

A role has *well-defined inputs and outputs* in the form of RSHs whose syntax and semantics can be tightly specified. It may also have specified APIs to other software components in the local node.

A role is identified by a *unique name* called a **RoleID**. A RoleID reflects the functionality provided. A full RoleID may have a multicomponent structure like a file name; the hierarchy would reflect the derivation of a specific role from a more generic one (Section 2.4). For efficient transport and matching, a corresponding short-form fixed-length integer RoleID will normally be used.

The RoleID only addresses meta-data to the provider of a type of functionality; it does not indicate which node will perform that functionality. RSHs in packets can also be addressed to the specific actor that instantiates a role[5]. RBA requires that node interfaces have unique addresses called NodeIDs, which would correspond to "network addresses" in the traditional layered architecture. Symbolically, we denote a **role address** in the form RoleID@NodeID, or RoleID@* if the NodeID is to be left unspecified.

To perform their tasks, role actors may contain internal **role state**. Establishing, modifying, and deleting role state generally requires signaling, which is done through the exchange of RSHs.

Some roles operate in a pair of nodes to enforce some condition in the intervening data path; simple examples are the pairs: (*Fragment, Reassemble*), (*Compress, Expand*) or (*Encrypt, Decrypt*). We call these **reflective roles**. It is possible to consider a reflective role to be either a pair of distinct sub-roles or to be a single distributed role.

Other special role categories may emerge as the role-based model is developed further. These sort of categories are

[5]We speak of the *address* of a role, meaning the address of one of its actors.

useful in bounding the flexibility that RBA provides, so that we can reason about the interaction between roles.

There are **families** of related roles that differ in detail but perform the same generic function. This generic function may be abstractly represented by a *generic role*. Specific roles may be derived from the generic role through one or more stages of specification (see Section 2.4). For example, corresponding to the generic role *ReliableDataDelivery* there might be specific roles for reliable ordered byte streams and for reliable datagrams.

2.3 Role Data

Under the idealized RBA model, all data in a packet, including the payload, is role data that is divided into RSHs. The set of RSHs in a particular header may vary widely depending on the services requested by the client and can vary dynamically as a packet transits the network. The relationship between roles and RSHs is generally many-to-many – a particular RSH may be addressed to multiple roles, and a single role may receive and send multiple RSHs.

Just as roles modularize the communication algorithms and state in nodes, so RSHs modularize the metadata carried in packets. RSHs divide the metadata along role boundaries, so an RSH forms a natural unit for ordering and access control on the metadata; it can be encrypted or authenticated as a unit.

The granularity of RSHs is a significant design parameter, since role data cannot be shared among roles at a smaller granularity than complete RSHs. At the finest granularity, each header field could be a distinct RSH; this would avoid any replication of data elements required by multiple roles. However, overhead in both packet space and processing requirements increases with the number of RSHs in a packet, so such fine-granularity RSHs are not generally feasible. As in all modularity issues, the optimal division into RSHs will be an engineering trade-off.

A role might modify its activity depending upon the particular set of RSHs in the packet. Furthermore, the presence of a particular RSH may constitute the request for a service from subsequent nodes. Thus, the RSHs provide a form of signaling that may piggy-back on any packet.

The format of an RSH is role-specific. It might have the fixed-field format of a conventional protocol header or it might have a (type, length, value) format, for example. An RSH contains a list of role addresses to which this RSH is directed and a body containing role data items. We denote this symbolically as:

RSH(<*RoleAddressList*> ; <*RSHBody*>)

For example, RSH(Expand@*N3*, Decrypt@*; ...) represents an RSH addressed to the role named *Expand* at node *N3* and to the role named *Decrypt* at any node.

RBA provides a model for packet header processing, not a mechanism for routing packets. Rather, RBA incorporates whatever forwarding mechanism is in use through a generic *Forward* role, which may depend upon global or local state in each node. A mechanism to create that state, e.g., a distributed SPF routing calculation, is simply another application from the viewpoint of RBA. Once the forwarding rules determine the actual route taken by a packet, the RBA sequence and scheduling rules come into play to determine the sequence of operations.

2.4 Technical Issues

Further definition of RBA requires specific solutions to a number of technical problems.

- Role Matching

  Rules must be specified for matching role addresses in RSHs with actors, taking into account the access control rules. An actor may have access to an RSH either because the RSH was explicitly addressed to that actor or because the actor was promiscuously "listening" for particular RSHs (again subject to access control rules.) An actor may read or write (add, modify or delete) an RSH (see the arrows in Figure 1).

- Actor Execution Scheduling

  Once the matching actors are selected, the node must determine in what order they should be executed. This scheduling problem must consider ordering requirements imposed by roles; these requirements are called *sequencing rules*. For example, such rules might prevent undesirable sequences like *Encrypt, Compress* (compression is not useful after encryption) or *Expand, Compress* (wrong order), or *Compress, Encrypt, Expand, Decrypt* (reflective pairs are improperly nested). These rules must consider dynamic precedence information carried in packets as well as static precedence associated with the actors in the nodes.

- RSH Access Control

  By controlling access to RSHs, RBA allows nodes, including end systems, to control what network services can be applied to specific packets. RBA provides two levels of access control, de jure and absolute. De jure access control is provided by bits in each RSH that grant specific roles read and/or write permission for the RSH. Write access would provide the ability to modify or delete the RSH from the packet.

  De jure access control is sufficient as long as nodes follow the RBA rules. Otherwise, nodes can absolutely control access to RSHs by encrypting these RSHs; of course, this greater certainty has greater cost.

- Role Definition

  To fully define a specific role, it is necessary to define its internal state, its algorithms, and the RSHs to be sent and received. In addition, some roles have non-network interfaces that must be defined.

  It remains to be seen whether RBA is amenable to the use of formal protocol specification techniques. One possible direction is to exploit the analogy between object-oriented programming and the derivation of specific roles from generic roles. If roles correspond to classes, then actors are instantiations of these classes, and RBA communication can be modeled by actors communicating via message passing.

- Role Composition

  Two roles R_a and R_b that communicate directly with each other using RSHs (which may originate and terminate in the two roles, or may be passing through one or both) should be composable into a larger role R_c. This binds R_a and R_b into the same node, and allows inter-role communication to be replaced by internal communication, e.g., shared data.

Conversely, a complex role may be decomposed into component roles, replacing shared data by explicit role data communication using RSHs.

2.5 RBA Examples

2.5.1 Simple Datagram Delivery

As a simple RBA example, the RBA equivalent to a simple IP datagram might be a packet containing the four RSHs:

{ RSH(LinkLayer@*NextHopAddr*;),
RSH(HbHForward@∗; *destNodeID*),
RSH(HbHSource@∗; *sourceNodeID*),
RSH(DestApp@*destNodeID*; *AppID*, *payload*) }

Here the *LinkLayer* role presents the link layer protocol, and its RSH is addressed to the next hop node. The *DestApp* role is the generic destination application role that delivers the payload in the role data to the application-level protocol specified by *AppID*. The *HbHForward* role represents a hop-by-hop forwarding function, invoked in every node along the path, with the destination address as its role data. It is one specific rule derived from the generic *Forward* role, which is the fundamental action of a router and of most middle boxes. It uses role data to determine one or more outgoing interfaces or next hops. *HBHSource* indicates the node ID that can be used to return a response by hop-by-hop forwarding.

2.5.2 Network Address Translators

Regardless of their architectural merit, network address translators (NATs) make a good RBA example since they do not fit well into a layered architecture. A NAT is essentially a packet relay that separates two different addressing realms. Complication is added by application-level protocols that are unaware of the NAT's existence but need to communicate addresses or ports end-to-end.

There are essentially two types of NAT. *Pure NATs* perform a dynamic but one-to-one mapping between a small pool of external addresses and a larger number of internal addresses. *NAPTs* perform a one-to-many mapping between a single external address and many internal addresses, by overloading of TCP or UDP port fields.

Pure NATs are simple to accommodate using RBA. The NAT simply inserts a RSH addressed to the role called *nat-receiver* giving the original address, and all software on any downstream nodes can listen to the *nat-receiver* role, see that the translation has occurred and act accordingly.

The RBA equivalent of a NAPT is a little more complex. A NAPT can behave exactly like a pure-NAT in its insertion of the *nat-receiver* RSH, but it also needs some way to demultiplex incoming packets to the correct internal address. One way to do this might use a general-purpose *echo* role. On outgoing packets, the NAPT inserts an RSH addressed to the *echo* role, giving a token that is unique to the internal address. All RBA systems should be aware of the *echo* role. If any RBA node generates a response to a packet containing a RSH addressed to the *echo* role, it should echo the token by including an RSH in the response packet addressed to the *echo-sender* role. This token mechanism is not NAT-specific, and it can form a useful building block for many new mechanisms.

This NAT example raises an interesting issue with regard to role naming. If, in the traditional manner, we named the RSHs rather than the roles we would have called the RSHs *token* and *token-echo*. The RBA philosophy is that it is the role played by the recipient that is named, and not the RSH. The distinction is an important and subtle one, as it significantly affects how future extensions may be deployed.

3. REALIZATION OF RBA

The role-based architecture approach described in earlier sections may be applied to network protocol design in a variety of ways. Further research will be required to determine which of these directions will be most fruitful.

In the extreme, one could build an architecture that is entirely role-based, i.e., all protocol functions from the present link layer to the present application layer as well as all middlebox functions would be replaced by roles or sets of roles. This would yield a completely layer-free, remodularized architecture.

There are two possible directions for less extreme ways to use the RBA approach. First, one can apply RBA only above a particular layer of the stack, retaining layering below that point. These partial stack implementations of RBA trade off generality and flexibility for efficiency and reality. For example, we may consider link-layer protocols to be immutable, since they are designed by industry groups to match particular technological constraints. A practical RBA subset might therefore retain the link layer as a distinct layer "below" RBA. Furthermore, retaining the IP layer as the highly-optimized common end-to-end packet transport service could significantly help to solve the efficiency issues with RBA; RBA processing would be needed only in end systems and middleboxes. A less strong argument could be made to retain the transport layer and apply RBA only as an application-layer architecture (note that this could still help immensely with the middlebox problem.)

The other possible direction is to use RBA to provide an unlayered network control (signaling) mechanism for essentially the current general modularity. From this viewpoint the network functionality would be divided into major *protocol entities* that might (or might not) assume particular roles. This viewpoint emphasizes the addressability of a role; RSHs would generally be created by protocol entities but addressed to, and received by, roles assumed by other entities.

Finally, the idealized RBA may be useful simply as an abstraction for reasoning about protocols, their functions and interactions.

A critical design decision when instantiating a role-based architecture is designing the packet format. There is a clear tradeoff between making the RSH header format quite powerful and general, versus wasting too many bytes on role addressing relative to the size of the information carried in each RSH. In the earliest sketch of RBA, we imagined a small number of well-defined roles and a field as small as 6 bits to address each RSH. Later we realized that RBA would be much more powerful if we could address RSHs more generally, and so the addressing information grew to include NodeIDs and larger RoleIDs. This has a direct effect - it is probably not cost-effective to split very simple low-level functionality into separate roles. The advantage is that at higher levels we have a more powerful mechanism for expressing complex interactions.

Furthermore, forwarding performance is an important real-world issue. In a very large network like the Internet, there

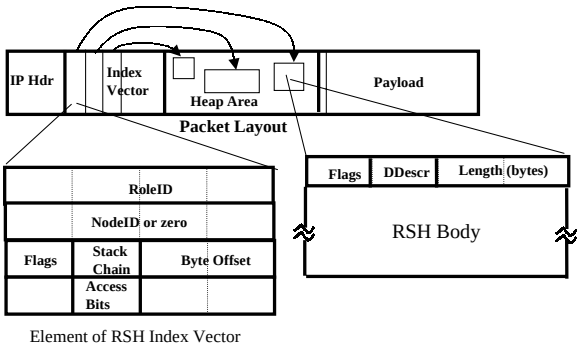


Figure 2: Possible RBA Packet Layout

are strong reasons to keep the basic packet forwarding machinery as simple and efficient as possible. A practical RBA would therefore retain the IP layer of the Internet, with its high-speed forwarding machinery and efficient packet header. RBA would then be applied above the Internet layer, i.e., it would replace only the transport and application layers. As a result, RBA would be implemented only in end systems and middle boxes, where performance requirements are much less severe than within the core network.

These assumptions could be incorporated into RBA by declaring that the generic *Forwarding* role and the reflective pair (*Fragment*, *Reassemble*) are "built in". These simplifications should not interfere with a major rationale for RBA, providing a consistent architectural basis for middle boxes, but they should make a RBA approach a realistic proposition for real networks.

3.1 Packet Structure

Figure 2 suggests a possible packet format for a practical RBA. This figure shows an encapsulating IP header, assuming that RBA is going to be applied only above the IP layer; this assumption is not necessary. If the network layer were brought under RBA, the IP header would be replaced by the actual *Forward.HbH* RSH, placed at a fixed location at the beginning of the packet to allow very efficient hardware-based forwarding through the network core.

The role address lists (see Section 2.3) of all RSHs in the packet are gathered together into a *index vector* of fixed-length entries. This should allow efficient processing of packets under RBA. Each entry includes a pointer (byte offset) to the corresponding RSH body in the heap area.

The RoleID is a globally-unique 32-bit short name for a role to which the RSH specified by this index element is addressed. As suggested earlier, it can be generated as a hash of the long name. This is shown as a 32-bit IPv4 address of the node to which this index element is addressed (RoleID@NodeID), or zero to indicate a wildcard (RoleID@∗)

As a packet traverses the network, RSHs may be added and deleted from its header. There are many engineering strategies that can help to keep this reasonably simple and efficient. There is no specified maximum size for the RSH space – the index vector or the heap area – but generally a source will have a good guess on how much space to reserve between IP header and payload for RSHs. The boundary between index vector and heap can be flexible, and these two segments can grow towards each other. A series of deletions and additions of RSHs could force garbage collection of the heap or movement of the payload to expand the heap size in an intermediate node. This is relatively complex and expensive, but it should seldom be necessary. In case a node is unable to add a new RSH to a packet, it can send a "RSH Overflow" RBA control message back to the sender node, requesting a larger RSH space in succeeding packets.

4. CONCLUSIONS

This document has proposed role-based architecture to simplify the design and deployment of communication protocols in today's world, where the complex interactions among networking elements often do not follow a strict layering model. RBA provides a uniform way to structure protocols and protocol processing without the confines of strict layering.

The generality of RBA does not come without cost, of course. The layered-network model has been a very powerful tool for conceptualizing and designing protocols. We need to satisfy ourselves that roles will provide a tool that is at least as good as, if not better than, layers for developing protocols. Furthermore, RBA requires a more general data structuring in packet headers, which has a cost in implementation, packet size, and execution performance. We must show that these costs are containable.

5. ACKNOWLEDGMENTS

We are grateful to the other members of the NewArch project, who have given much encouragement on developing the RBA concepts. We especially thank Dave Clark, John Wroclawski, Karen Sollins, Noel Chiappa, and Aaron Falk.

6. REFERENCES

[1] N. Bhatti and R. Schlichting. A System for Constructing Configurable High-Level Protocols. *Proc. ACM SIGCOMM '95*, 138-150, 1995.

[2] Z. Haas. A Protocol Structure for High-Speed Communication over Broadband ISDN. *IEEE Network*, 5(1):66-70, January 1991.

[3] N. Hutchinson and L. Peterson. The x-Kernel: An Architecture for Implementing Network Protocols. *IEEE Trans on Software Eng.*, 17(1):64-76, 1991.

[4] ISO. Information Processing Systems - Open Systems Interconnection - Basic Reference Model. ISO 7498, 1984.

[5] E. Kohler, M. Kaashoek and D. Montgomery. A Readable TCP in the Prolac Protocol Language. Proc SIGCOMM '99, 1999.

[6] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. Kaashoek. The Click modular router. *ACM Trans on Computer Sys.*, 18(3):263-297, 2000.

[7] E. Meyer. ARPA Network Protocol Notes. RFC 46, Network Working Group, April 1970.

[8] S. O'Malley and L. Peterson. A Dynamic Network Architecture. *ACM Trans. Comp. Sys.*, 10(2):11–143, May 1992.

[9] C. Tschudin. Flexible Protocol Stacks *Proc. ACM SIGCOMM '91*, 197-204, 1991.

[10] M. Zitterbart, B. Stiller, A. Tantawy. A Model for Flexible High-Performance Communication Subsystems *IEEE JSAC*, 11(4):507-518, May 1993.

Network Pointers

Christian Tschudin
Uppsala University and
CS Dept, University of Basel
Bernoullistrasse 16
CH–4056 Basel, Switzerland
Christian.Tschudin@unibas.ch

Richard Gold
Department of Computer Systems
Uppsala University
Box 325
SE–75105 Uppsala, Sweden
rmg@docs.uu.se

ABSTRACT

The Internet architecture can be characterized as having a rather coarse grained and imperative style of network packet handling: confronted with an IP packet and its source and destination addresses, the infrastructure almost blindly and unalterably executes hundreds of resolution, routing and forwarding decisions. There are numerous attempts that try to "extend" the Internet in order to either reduce the immediate impact an arbitrary packet can have (e.g., NAT) , or to insert diversions from the normal processing paths in order to better use the existing resources (e.g., content delivery). In this paper we argue that we need a more fine grained control, in the hands of end nodes, over how packets are handled. The basic abstraction presented here is that of networking pointers, which we show to relate to low level concepts like ARP caches, but also high level routing decisions for terminal mobility, content delivery networks, or peer-to-peer overlay forming. We report on first implementation experiences of an "underlay" networking approach which uses pointer tricks underneath IP in order to provide new network layer services.

1. INTRODUCTION

The universal connectivity provided by the Internet architecture is the major source of the stress that it is currently experiencing. Because the Internet has one, and only one method to identify a destination (IP address), to deliver datagrams (routing tables managed by third parties) and to address services (ports), there have been many attempts to introduce more flexibility into the Internet:

- Overlays, be it for peer-to-peer, virtual private networks, or route optimization à la the Resilient Overlay Networks project [1], redo what the IP layer is supposed to do: packet forwarding. The difference is that end nodes explicitly want to have a say in how forwarding should be done.
- The IP address merges three different networking concepts in a perhaps elegant but troublesome way: identity, location and access. Changing location whilst preserving identity is the source for the Mobile IP proposal. On the other hand, the conflict between identity and access is where firewalls and NATs intervene.

Changing the Internet's packet forwarding behaviour really means changing the way the Internet manages state. We ask the question: based upon which header fields, which routing tables and which lookup processes should forwarding decisions be taken? There is a myth associated with the Internet that it is stateless. However, this only refers to per-connection state and does not apply to the infrastructure itself. On the contrary, a surprisingly large amount of configuration state is present in the network: default routes, address ranges in DHCP, BGP tables, OSPF tables, DNS content, port to service mappings, and more recently the addition of per-connection state in the form of NAT mappings.

1.1 Network Pointers in a Nutshell

Network pointers provide a conceptual and programming framework for packet processing in general. In the first place, a network pointer is an arbitrary packet processing function that can be "addressed" through an opaque pointer value.

A simple example would be the sending of an IP packet to an end point: the destination IP address is mapped to the IP address of the default gateway. This is then mapped to the ethernet address of the gateway's interface and cached in an ARP table entry. In fact, after the aforementioned mapping procedure, the use of any (local) pointer value identifying the cache entry would suffice to forward data to the destination. What we propose is to make that resolution from names and other entities to pointers an explicit architectural element across the whole network. Network pointers and their remote instantiation play a key role in providing "directable indirection" to the network users and operators.

Before going into more details we now review some items of related work which concern themselves with the manipulation of state (and thus packet processing functionality) in the network. This state is typically used to alter the amount of direction and indirection in the network.

This work was done under the VINOVA project SCANET (Self-Configuring Ad hoc Networks). We gratefully acknowledge the support from the European COST.263 action on "Quality of Future Internet Services".

1.2 Related Work

The *Resilient Overlay Networks* (RON) project [1] is designed to increase the reliability of the routing paths present in today's Internet routing infrastructure. It works by creating an overlay network of RON nodes which all agree to forward packets on behalf of the other nodes in the overlay network. During typical operation a RON node probes the standard paths provided to it by the network for failure. When such path failure is detected, the node then attempts to route around the failure. We see RON as re-creating loose source routing at the overlay level. It builds alternate routes (indirection) over an IP routing layer that cannot be steered i.e., which is too direct. Similarly to RON, Mobile IP had to also invent its own routing agents in order to work around the lack of influence an end-user has on routing decisions.

The fact that the Internet suffers from being overly direct has also been picked up by the Internet Indirection Infrastructure (i3) project [9]. They see indirection, as a generic concept, is sorely needed in the current Internet. In order to provide this, they use a Peer-to-Peer lookup service which implements a Rendezvous approach i.e., meeting in the middle. A receiver puts a key called a trigger into the lookup service. This trigger is then used by the sender to route a packet through the i3 overlay network to the receiver. Triggers are much like network pointers, except that i3 restricts itself to IP addresses as the only type of pointer values and to IP forwarding as the single supported packet processing function.

Indirection can also be a problem, as exemplified by NAT boxes and Firewalls which introduce indirection in an implicit way. Many tunneling mechanisms have been devised (e.g., PPP-over-SSH [3]) in order to bring back a direct connection (IP-wise) between two end systems when it is desired. Being unable to control when direction and indirection occurs is a major hindrance in the Internet of today and has resulted in many patches in order to get over these problems. We see NAT and overlay networks as a general symptom of this lack of control.

1.3 Got state?

In the face of this inaccessible state and inability to influence the directness of the current Internet, we propose a program of Internet deconstruction: breaking the packet processing black box into components and allowing its recombination in a user-controlled manner via pointer manipulation. Network Pointers, like their programming language cousins, allow the user to control the amount of in- and re-direction. We then present our own architecture based on the idea of Network Pointers and late binding of address semantics. By shifting the focus to underlaying IP (instead of overlaying IP), much richer and optimizable network recombinations are possible, as we are able to combine the components of the IP stack in different ways. IP (and its associated socket library) then becomes an access mechanism and emulation target, whereas overlays – now supported at the same level as IP – are a way to create partial and transient clouds of emulated directness.

2. NETWORK DECONSTRUCTION

In this section we briefly discuss three network usage scenarios that serve as prototypical examples for the deficiencies of the Internet's one-model-fits-all approach. We will also show how these scenarios would benefit from "network pointers". A network pointer basically stands for a packet processing function. If a packet is "sent to a pointer's value", it will be processed according to that pointer's function. This function can be, for example, the forwarding of a packet whose header contains a link layer address and the pointer value of a next (remote) network pointer. How these pointers are allocated and managed will be discussed in the following section.

2.1 Case 1: Routing in Ad hoc Networks

Ad hoc networks are supposed to set up Internet connectivity in a completely decentralized way, without any help from preconfigured networking elements, and in very short time. One class of successful ad hoc routing protocols works "on-demand" i.e., they explore the network topology only when there is traffic to be delivered. In this case, the source node establishes delivery *paths* that run through the network. These paths funnel all traffic for a destination in the right direction. Expressing this in other terms, on-demand ad hoc routing protocols establish per-target "connection trees" whose root is located at the target node.

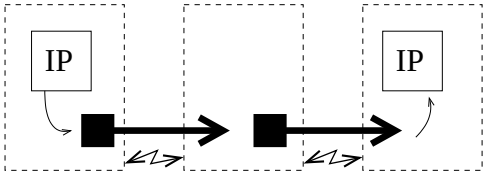


Figure 1: ARP forwarding and pointer selection in wireless ad hoc networking.

We can easily represent that tree with forwarding pointers being the nodes *and* the connecting edges. Instead of labeling these pointers with the target's name, we imagine that a pointer (node) has a *local* pointer value and that all neighbours just point to that address (i.e., "link layer" details plus pointer value at the remote node). This is shown in figure 1.

Our approach differentiates the "access" (pointer value, which is a simple label) from the "location" (full delivery path). Data packets sent to the first *access* pointer will be correctly forwarded, regardless of the final destination's *location*.

2.2 Case 2: Asymmetric Connectivity (NAT + Firewalls)

NAT boxes, either introduced for coping with the scarcity of IP addresses or for controlling network traffic, break the basic universal connectivity model of the Internet. Some nodes or services behind the NAT are not accessible from the "ordinary" Internet, while they can see and connect to the "ordinary" nodes. In order to overcome this asymmetry, people have started to setup tunnels and reverse tunnels to access machines behind NAT boxes. A legitimate use of such tunneling is a traveling user wanting to access some files that he left on his home machine unfortunately located behind a NAT. First, before leaving, the user would have had to create a persistent tunnel from his home machine to some

waypoint machine that has full connectivity to the Internet. Secondly, when abroad, the user has to login to the waypoint machine, and from there through the reverse tunnel connect to the home machine. Overall, this exports one port i.e., SSH from behind the NAT to the outside world.

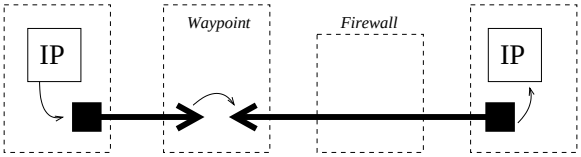


Figure 2: Using an outgoing tunnel to pass through the firewall.

By using network pointers in this scenario, it is possible to concatenate together application-layer tunnels (e.g., SSH) as well as standard IP paths into one seamless construct. Typically the traffic from a node in the traditional Internet to a node behind a NAT box has to be manually directed to a waypoint, and then from the waypoint into the tunnel to traverse the NAT box as shown in figure 2. By allowing user-defined mappings to be established between pointer values and their associated packet processing functions, we can use different layers of the networking stack on a per hop basis. This enables us to present nodes behind NAT boxes as nodes with full Internet connectivity.

2.3 Case 3: Mobile Personal Area Networks (PAN)

An important new challenge for computer networks are small area networks linked to a person, which interconnect the cellular phone with the embedded PC in the belt, the wireless ear plug, and the MP3 player in the pocket etc. Networked devices travel as a clique and rely on special nodes (e.g., the cellular phone) to connect them to the rest of the Internet. Because of the intermittent connectivity and often changing access points to the fixed network, it would be problematic if all devices in the PAN were to handle routing and addressing on an individual basis.

In this case we use a routing data structure comprised of two trees where one tree (i.e., set of pointers) encapsulates the PAN and its components. In figure 3 this is the tree from the fixed network to the PAN, where the dotted lines represent logical paths to nodes inside the PAN and the unbroken line represents the physical path. Another tree is then used from inside the PAN to point from the current gateway to the PAN components to allow them access to the fixed network. This process is shown in figure 3. This pointer from the fixed network to the PAN is addressable i.e., other nodes can then use this pointer as a way of communicating with the PAN.

The pointer inside the fixed network can also be called a "point of presence", or rendez-vous place in terms of the Internet Indirection Infrastructure proposal. Again, we dissociate identity from location and change the pointer values and names locally and on demand: the data structure's end points (the PAN members, and the "point of presence" in the fixed network), remain stable, while the intermediate pointers can change without requiring the end nodes to re-

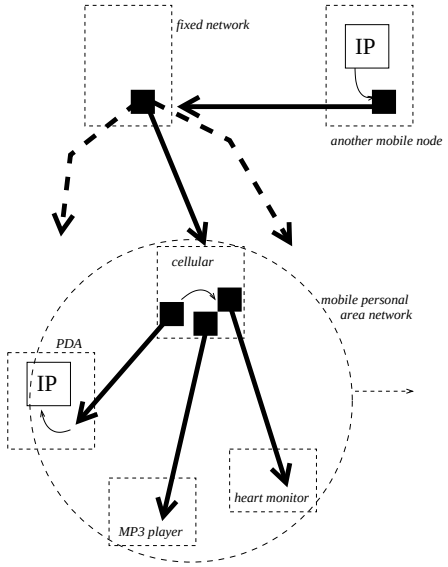


Figure 3: Changing forwarding pointers to the PAN as a whole.

act on this.

2.4 The Deconstruction Plan

Our general approach with network pointers is to facilitate the introduction of new functionality into the network in the following way:

We keep IP and applications running on end nodes as they are. We then go under IP as we wish to have the ability to influence the functionality of the lower layers rather than just treating them as a black box. This allows us to configure the logical layer 2 and the ability to perform redirection at this layer. Addresses at the IP layer are translated to internal access names i.e., pointer labels. Pointers are responsible for either forwarding a datagram to a neighbour, or for modification of packet headers (address translation, tunneling, header compression, etc).

Once we are in the "pointer space", we can begin to build our own abstractions. For example, we can bring an Ad-Hoc networking cloud into the IP space without IP having to be concerned about the multihop conditions of the underlying network. Simplification is also the order of the day with the Firewall/NAT traversal scenario. Rather than having multiple manual steps from the NAT'd machine to the waypoint and then from the user's machine to the waypoint, we make pointer allocation methods available that enable a semi-automated setup of NAT tunnels.

An important aspect of a pointered network architecture is the management of pointer state which we require to pass through a mandatory resolution step. Pointers become a natural translation point not only for addresses and tunnel splicing, but also for management domains. By restricting resolution e.g., linking it to authentication, we yield more control over a network's infrastructure to the operator. Ultimately, the plan is to deconstruct IP forwarding and addressing semantics in order to build new network layer ser-

vices in our own "Pointer Space" that can be mapped back into the IP space. How resolution and pointer allocation works is explained in more detailed in the following section.

3. SELECTORS AND RESOLUTION

3.1 Terminology

A network pointer is a packet processing function that has a local address, called a selector. To send a packet to a network pointer, you need to know the pointer's address (selector) and the context in which that selector is valid. Inside a context, which is typically a host, packets can be routed through different pointers based on selectors only. In order to send packets to pointers residing in other contexts, one also needs to specify a transport specific context address (Ethernet, IP, Application-layer tunnels etc.). In this case of a packet forwarding function, the network pointer itself contains the address of the downstream function i.e., the link layer address and the remote selector.

In case of Ethernet, packets would carry a link layer address and a selector. The link layer address determines the context in which the function can be selected. A forwarding chain of network pointers would rewrite a packet's link layer address and selector value on each hop i.e., the network pointers contain the address of the downstream node as well as the selector value of the remote pointer.

Selectors are local i.e., they apply to the local context. This enables one to route packets through local pointers in a uniform way: network pointers become the back plane of protocol stack components (see also section 4). At the same time, selectors can point to network pointers that transport packets to other contexts. In this way contexts can be nested, which leads to layers and tunnels. In the tunnel case, the selector would point to a tunnel entry function that pushes its own addressing data in front of the packet header. The tunnel exit function, activated by the tunnel selector on the last forwarding leg, strips out that additional addressing. Protocol layering is obtained by letting each protocol entity of a layer sending packets for higher layers to the selector of the next higher protocol entity.

We can use the network pointer abstraction to represent forwarding state (possibly instantiated by end nodes) in the network *or* in the packet header. If contexts are globally addressable we can use network pointers as remote relay points in order to implement a cached form of loose source routing: packets only need to carry the name of the first pointer, from where they will be passed on from one pointer to the other. Alternatively we can send packets to specific transport pointers that use the packet header as a parameter stack by reading the destination parameters immediately following the packet's selector field.

3.2 Network Pointers and C-Pointers

The term "network pointers" relates to the standard forwarding behavior of network nodes: Routers "redirect" incoming packets to other directions. This is basically a dereferencing operations (usually involving some lookup in a router table) and matches well the classical concept of pointers in programming languages. The main difference is of course the distributed control: in our network pointer view, each data packet becomes a thread of control that works its way through a chain of dereferencing operations. Whereas a list traversal in the classical programming world would be carried out by a single CPU. The other difference is that we generalize the concept of pointer and include with it the possiblity to mangle packets, hence we see network pointers as generic packet processing functions. Note that there is not a one-to-one mapping between packets in and packets out for a network pointer. It may produce two or more outgoing packets (if multicast semantics are required) for each incoming packet or it may require two or more incoming packets per outgoing packet.

3.3 Selectors as Local Addresses

Network pointers are addressed by a pointer value (selector) that we prepend to each data packet. In order to avoid spanning the name space for all potential pointers (and having to enumerate them at a global level), we require that selectors are purely local names. Only those pointers can be addressed that actually are resident in a specific context. The selector, typically implemented as an opaque number, will be used at the receiving context to find the network pointer that the packet should be processed by. We leave the issue of end-to-end addressing to whichever higher-level protocols are using the network pointer infrastructure.

The local naming scheme means that selectors have no global meaning and that, for example, packets being forwarded from one context to the other have to change their selector on each hop. Selector (or header) translation thus is an intrinsic property of a pointered network architecture. To some extent this also applies to current IP networking, where the destination address is repeatedly rewritten to some local ethernet address that is different for each hop.

3.4 Name Resolution

Due to the fact that local pointer names shall be the generic and sole addressable items, we need a way to map end-to-end destination addresses or server names to local selector values to other kinds of identifiers, including global addresses. To this end we provide a generic resolution function residing on all network nodes to which we assign a well–known selector value. Using this well-known selector, an application can request address resolution and get a local pointer name that "stands for" the item to resolve.

For example, a request for resolving a neighbour's IPv4 name to the Ethernet address (à la ARP) would result in a local delivery pointer being instantiated on the fly which takes care of delivering any packets sent to it to this neighbour. In fact, such function instantiation on the fly already occurs today inside protocol stacks where an ARP cache keeps that mapping in memory. The selector view makes this state explicit and presents the associated network pointer in the form of an addressable selector. We note here the difference between resolution and translation. First, the IPv4 name has to be resolved. Subsequent packets however will be subject to header translation (which is much cheaper), essentially rewriting the local delivery selector into the associated Ethernet address details.

Extending this view we map routing to selectors too. Finding the next hop is in fact a resolution request for a route.

This can be done for each single packet or for a complete packet flow, as for example, in on-demand routing protocols for ad hoc networks. As a result, selectors become the (local) names of delivery path entries.

Explicit name resolution is also the entry point for substituting content descriptors or locators with access paths to the (closest) place from which content can be fetched, which is very similar to the approach proposed in TRIAD [7].

3.5 Start-Addresses, not End-Addresses

Unlike IPv4 addresses, a selector does not define a packet processing semantics in advance and across the whole network. Instead, explicit resolution activities are required to bind a selector to some network pointer (packet handler). We have thus moved from an end-address point of view to a start-address point of view. This enables to repointer parts or all of a packet processing chain in order to cope with mobility without having to change higher layer applications (see e.g., example 3 on moving personal area networks).

3.6 SelNet - An Implementation for Network Pointers

We have implemented a Selector Network for the Linux operating system. From former explorations we already knew that packet forwarding based on simple selector lookup rather than IP routing, can be done with 30% less overhead [13]. What we wanted to understand better was the role of the resolution protocol as a general tool for setting up the network pointers.

Consequently we implemented an "eXtensible Resolution Protocol" (XRP) and combined it with the SAPF [11] packet format which represents our network pointers. As a proof of concept for our underlay approach we applied the pointer networking approach to wireless ad hoc networks. The goal was to fool IP about the multihop nature of an ad hoc network by implementing ARP forwarding: ARP requests would return the name of a full delivery tunnel to use instead of a next hop link layer address.

The resulting LUNAR ad hoc routing protocol (Lightweight Underlay Network Ad hoc Routing [12]) works by trapping ARP requests and translating them into XRP queries that are sent to neighbouring nodes. Based on these queries, the neighbours create new network pointers and propagate the request until the destination is found. To our surprise, we could get a first LUNAR version up and running very fast (in hours). Also, the implementation matched the performance of well-established ad hoc routing protocols, although requiring only 30 to 60% of their code size. A stripped down version of LUNAR was even ported to the Lego Mindstorm embedded devices [8].

4. FUTURE DIRECTIONS

We envisage network pointers to become a pivotal element in future protocol architectures, both at the conceptual and the implementation level. We describe in this section some possible future applications of network pointers.

Protocol heaps are proposed in [2] as a middleware concept for letting applications influence the processing of packets through the network. Beside the correspondence at the representation level (heap and pointers), we believe that network pointers provide an essential building block for steering packets through processing components and for identifying processing instances. For example, the NAT example presented in [2] makes use of opaque "cookies" which we would map to selectors in a much more natural way: network pointers would then play the role of a cabling back plane for protocol heap modules.

Related to protocol stack engineering we point to work done a decade ago on speeding up packet processing called **active messages**: several header parsing steps can be avoided by putting the upcall's memory address directly into the packet's header [6]. This works well in the controlled environment of distributed systems: for an open network environment, however, additional protection of handler addresses is required. In our network pointer approach we use local selectors as well as an explicit resolution step for retrieving the dynamic selector values.

The use of selector-like state identifiers has been recently proposed in [4]: **Ephemeral state processing** (ESP) enables to instantiate short and predefined remote computations in a lightweight fashion. Each packet can carry a single ESP instruction in order to operate on the "ephemeral state store" residing in each node. State entries are identified with a randomly chosen 64-bit tag, which corresponds very well with our selector concept to address state and functions.

Finally we point to our own ongoing work on **stored program router** [10] where the forwarding table is turned into a general programming area. Individual table entries contain single instructions (e.g., forward, push label, conditional forward) such that packets can be processed by routing them through a chain of table entries. Because packets can also influence the table's content, we obtain a complete computing model where threads of computations are carried by single packets. The chaining of execution steps is done via the packet's selector value which becomes a direct *instruction pointer*. In terms of the network pointer concept, we organize the routing table as a single context consisting of an array of tiny network pointers.

5. SUMMARY & OUTLOOK

A "network pointer" basically is a packet processing function similar to the indirection support proposed in the i3 project. However, instead of using IP addresses as the target we propose to use a separate local name space (selectors) for labeling network pointers. This enables a precise separation of the IP address meanings (identity, location, access), although only a single naming concept is used.

More generally, network pointers represent packet processing functions that can range from ad hoc path discovery to multicast forwarding, content access, or VPN encryption. Network pointers bring back part of the Catenet architecture [5] where explicit control of address translation and concatenation was a key concern before the flat Internet model began to rule the world. Now that the Internet has become asymmetric and fragmented for good or for bad reasons, we

need to empower the end nodes, enabling them to recreate end–to–end services by rewiring the network's packet processing functions.

Acknowledgements

The authors wish to gratefully acknowledge the useful discussions and comments of , Per Gunningberg, Erik Nordström, Arnold Pears and David Lundberg. The comments of the anonymous reviewers also helped form the later version of the paper.

6. REFERENCES

[1] David G Andersen, Hari Balakrishnan, M. Frans Kaashoek and Robert Morris. Resilient Overlay Networks Proc. 18th ACM SOSP 2001. http://nms.lcs.mit.edu/papers/ron-sosp2001.html.

[2] R. Braden, T. Faber, and M. Handley. "From Protocol Stack to Protocol Heap – Role-Based Architecture". To appear in: First Workshop on Hot Topics in Networks (HotNets-I) 2002.

[3] Scott Bronson. VPN PPP-SSH Mini HOWTO. http://www.ibiblio.org/pub/Linux/docs/HOWTO/mini/other-formats/html_single/ppp-ssh.html.

[4] Kenneth L. Calvert, James Griffioen and Su Wen. Lightweight Network Support for Scalable End-to-End Services. In: SIGCOMM 2002. http://www.acm.org/sigcomm/sigcomm2002/papers/esp.pdf

[5] Vint Cerf. The Catenet model for Internetworking, 1978. http://www.isi.edu/in-notes/ien/ien48.txt.

[6] T von Eicken, D. E. Culler, S. C. Goldstein, and K. E. Schauser. Active Messages: a Mechanism for Integrated Communication and Computation. In: Proceedings of the 19th Int'l Symp. on Computer Architecture. May 1992.

[7] M. Gritter and D. Cheriton. An Architecture for Content Routing Support in the Internet. Usenix Symposium on Internet Technologies and Systems 2001. http://www-dsg.stanford.edu/triad/usits.ps.gz.

[8] Anders Linden, Johan Loennberg, Olof Rensfelt, and David Rosen. μLUNAR, 2002. http://www.docs.uu.se/selnet/lunar/ulunar/.

[9] Ion Stoica, Dan Adkins, Sylvia Ratnasamy, Scott Shenker, Sonesh Surana and Shelley Zhuang. Internet Indirection Infrastructure. IPTPS 2002. http://www.cs.berkeley.edu/~shelleyz/papers/iptps2002.pdf.

[10] Christian Tschudin. Stored Program Routers. Work-in-Progress, 1999–2002. http://user.it.uu.se/~tschudin/pub/cft-wp-spr.pdf

[11] Christian Tschudin and Dan Decasper. Active Networks Request for Comment: Simple Active Packet Format (SAPF), 1998. http://www.docs.uu.se/~tschudin/pub/cft-1998-sapf.txt.

[12] Christian Tschudin and Richard Gold. LUNAR: Lightweight Underlay Network Ad-hoc Routing. Submitted for publication. 2002. http://www.docs.uu.se/selnet/lunar/lunar.pdf.

[13] Tilman Wolf, Dan Decasper and Christian Tschudin. Tags for high-performance Active Networking. Proc *Openarch 2000*.

Internet Research Needs Better Models

Sally Floyd Eddie Kohler

ICSI Center for Internet Research, Berkeley, California

{floyd, kohler}@icir.org

1 INTRODUCTION

Networking researchers work from mental models of the Internet's important properties. The scenarios used in simulations and experiments reveal aspects of these mental models (including our own), often including one or more of the following implicit assumptions: Flows live for a long time and transfer a lot of data. Simple topologies, like a "dumbbell" topology with one congested link, are sufficient to study many traffic properties. Flows on the congested link share a small range of round-trip times. Most data traffic across the link is one-way; reverse-path traffic is rarely congested.

All of these modeling assumptions affect simulation and experimental results, and therefore our evaluations of research. But none of them are confirmed by measurement studies, and some are actively wrong. Some divergences from reality are unimportant, in that they don't affect the validity of simulation results, and simple models help us understand the underlying dynamics of our systems. However, as a community we do not yet understand which aspects of models affect fundamental system behavior and which aspects can safely be ignored.

It is our belief that lack of good measurements, lack of tools for evaluating measurement results and applying their results to models, and lack of diverse and well-understood simulation scenarios based on these models are holding back the field. We need a much richer understanding of the range of realistic models, and of the likely relevance of different model parameters to network performance.

2 NETWORK MODEL PRINCIPLES

By *network model*, we mean the full range of parameters that might affect a simulation or experiment: network topology, traffic generation, end-node protocol behavior, queue drop policies, congestion levels, and so forth. Internet experiments are difficult to replicate, verify, or even understand [17] without the stability and relative transparency provided by a simulator (such as ns [15]), emulator (such as the University of Utah's Emulab [18]), or self-contained testbed; and experimental design for these platforms includes the design and implementation of an explicit and concrete network model.

Network models used in practice often have little relationship to Internet reality, or an unknown relationship to Internet reality. This isn't necessarily a problem. Divergences between models and reality can be unimportant, in that they don't affect the validity of simulation results, or useful, in that they clarify behavior in simple cases. Some divergences are necessary in order to investigate the Internet of the future instead of the Internet of the past or present. However, the research community has not yet determined which divergences are acceptable and which are not. We simply don't know whether the models we use are valid. This basic question has led to difficulties both in our own research and in our evaluation of other work.

We need better models and better tools for evaluating our own and others' models. We need to know when a model might lead to bad results, and what those results might be. In particular, we believe:

Models should be specific to the research questions being investigated. We wouldn't recommend trying to construct a single model of the global Internet, with a single set of simulation scenarios, for use by all researchers. The Internet cannot be simply and accurately modeled in the same way that one might model a machine that one could hold in one's hand. Researchers should instead concentrate on modeling properties relevant to their research, and finding valid simplifications or abstractions for other properties. The very process of deciding which properties are relevant, and testing those decisions, gives insight into the dynamics of the questions under investigation. Building a single global model, in contrast, would make people's simulations run slower without necessarily improving their precision, clarity, or applicability.[1]

For example, one area of particular interest to us is congestion-related mechanisms at a queue in a router. This includes such research topics as differentiated services, active queue management, ECN, QoS, aggregate-based congestion control, fairness, and so forth, and touches on other issues, such as design of end-host protocols. Models for these topics must include characteristics of congested links, the range of round-trip times for flows on a congested link, and the effects of congestion elsewhere on the network. A fully-worked-out topology isn't necessary, however; the range of round-trip times, and an understanding of the congestion experienced elsewhere, sufficiently represents the topology. Table 1 describes typical models used in other research areas, such as unicast and multicast congestion control, routing lookups, and peer-to-peer systems.

We need to understand how models' parameter settings affect experimental results. As a model for a given research question is built, researchers should explore the model's parameter space. For example, do some parameters change results only slightly, or are results sensitively dependent on one or more parameters? Section 3 explores this in detail for several research questions. An understanding of the realm of possibilities, and their causes, can prove invaluable for interpreting results, and should be codified and distributed as part of the research community's shared knowledge base.

Modeling must go hand-in-hand with measurement. It is necessary to fully explore the range of parameter settings, but researchers should agree on particularly important settings to facilitate comparison of results. Network research should not founder on dis-

[1] Application-specific modeling is becoming a shared agenda in the research community, with work into application-driven topology modeling, for example [20].

First Workshop on Hot Topics in Networks, Princeton, New Jersey, October 28–29, 2002

Research Topics	Typical Models	Supporting Measurements
AQM, scheduling, differentiated services.	A dumbbell topology, with aggregate traffic.	Characteristics of congested links, range of round-trip times, traffic characterization (distribution of transfer sizes, etc.), reverse-path traffic, effects of congestion elsewhere.
Unicast congestion control.	A single path, with competing traffic.	Characteristics of links, queue management along path, packet-reordering behavior, packet corruption on a link, variability of delay, bandwidth asymmetry.
Multicast congestion control.	A single multicast group in a large topology.	Router-level topologies, loss patterns, traffic generation by group members.
Routing protocols.	A large topology.	Router-level topologies, AS-level topologies, loss patterns.
Routing lookups.	A lookup trace, or a model of the address space.	Ranges of addresses visible at a link.
Web caching and CDNs, peer-to-peer systems.	Models of large topologies with application traffic.	Topologies, application-level routing, traffic patterns.
Controlling DDoS attacks.	Models of large topologies with aggregate traffic.	Topologies, attack patterns.
Web cache performance.	A single cache with many clients and servers, as in Web Polygraph.	Detailed client behavior, server behavior.

TABLE 1—Some research topics, with typical models and required supporting measurements.

agreements over the network models and simulation scenarios that should be used. (Section 3 describes cases where we are close to that state of affairs.) Measurement can help settle these disagreements by saying what parameters, or ranges of parameters, are actually observed in practice.

We want models that apply to the Internet of the future, as well as to the Internet of today. Due to the Internet's vast heterogeneity and rapid rate of change [17], we must pay close attention to what seems to be invariant and what is rapidly changing, or risk building dead-end models. Measurement, for example, should be an ongoing program, so that old measurements don't congeal into widely accepted, but inappropriate, parameter settings.

Better models will make the Internet community's research efforts more effective. Lack of agreement over models complicates comparison and collaboration, and researchers risk expending valuable effort on dead ends caused by invalid models. Better models will therefore immediately improve the state of Internet research, and perhaps the Internet itself.

3 "ROGUES' GALLERY"

This section describes some modeling issues in our own, and others', network research. Some of the research we discuss has flaws, caused by inappropriate models, that might have been avoided given a better understanding of the network models appropriate for specific research topics. Some of it has not received a thorough evaluation because the models underlying the research have not been evaluated. The point is not to scold others (or ourselves!). Concrete examples are simply the most effective way to communicate the range of problems that can crop up when models aren't treated carefully enough.

Again, if models used today could be counted on to give similar results to one another, and if their results could be counted upon to be relevant to the current and/or future Internet, then there would not be a problem. However, different models and different simulation scenarios do give different results when used to evaluate the same research question, and have different degrees of relevance to the actual Internet.

3.1 Phase Effects

For example, some simulations demonstrate sensitive dependence on precise parameter settings. This rich behavior is not relevant to the modern Internet; it is an artifact of unrealistic simulation scenarios, such as those with long-lived traffic, packets the same size, and no reverse-path traffic. We would like to discourage researchers from investigating in depth the rich behavior of these unrealistic and irrelevant scenarios [19].

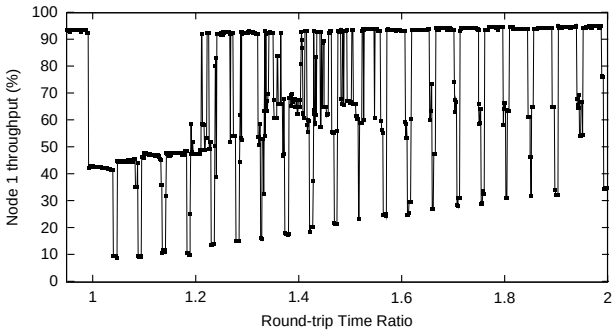


FIGURE 1—Flow 1's throughput as a function of the ratio of the two flows' round-trip times.

Figure 1 (taken from [6]) illustrates phase effects, where a small change in the propagation delay of a single link completely changes the fraction of link bandwidth received by one of two TCP flows sharing a Drop-Tail queue. Each dot on the graph represents the result of a single simulation; the *y*-axis shows the throughput of flow 1 in that simulation. The simulation topology is a simple dumbbell. When the propagation delays of the two competing flows' access links are equal, then both flows have the same round-trip time and receive the same fraction of the link bandwidth. However, as the propagation delay of one of the access links changes slightly, flow 1 can shift to receiving almost all of the link bandwidth, or to receiving very little of the link bandwidth, depending on the exact propagation delays of the two access links. In real networks, of course, the traffic mix includes short-lived flows, and small control packets as well as large data packets, and probably more than two

competing flows, all making phase effects much less likely. The lesson is not that phase effects are a significant or important dynamic to address in current networks, but rather that simulations can be very tricky and unrealistic, and that the combination in a simulation scenario of DropTail queue management with one-way long-lived traffic can be deadly indeed.

3.2 Active Queue Management: Parameters

Random Early Detection (RED) was one of the first proposals for Active Queue Management, and the 1993 paper on RED [7] included a number of simulations, investigating scenarios with a range of round-trip times; varying traffic load over the life of the simulation; two-way traffic including TCP connections with a range of transfer sizes; scenarios including bursty and less-bursty traffic; and a range of values for the configured target average queue size.

However, the 1993 paper neglected to address some key issues:

– The paper did not investigate performance in scenarios with high packet drop rates.

– The paper did not investigate performance for a range of link bandwidths for the congested link.

– The paper did not explore the potential for oscillations in the average queue size, in particular for scenarios with large propagation delays and long-lived traffic.

Partly because the paper neglected to address these issues, a lengthy literature was spawned on the limitations of RED, and nine years later Active Queue Management has still not seen widespread deployment in the Internet.

For instance, all of the paper's simulations were of scenarios with small packet drop rates, so performance looked quite nice. However, it was soon pointed out that performance looked less good when the packet drop rate exceeded RED's configured parameter max_p.[2] In 1997, the default value for max_p in the NS simulator was changed from 0.02, an unrealistically optimistic value, to 0.1. In 1999 the 'gentle' variant was added to RED to give increased robustness when the average queue size exceeded the maximum threshold, and Adaptive RED was developed in 2001 to adapt RED parameters to changing network conditions [5]. All of this might have been done much sooner if the authors of the RED paper (i.e., one of the co-authors of this paper) had paid more attention in 1993 to RED performance in scenarios with high packet drop rates.

Similarly, while the original RED paper gave guidelines for the setting of the queue weight parameter w_q, all of the scenarios in the paper had a congested link of 45 Mbps. This led to work by others using NS's default value of the queue weight parameter for a range of inappropriate scenarios, e.g., with 10 Gbps links, so that the average queue size was estimated over too small of a time interval, only a fraction of a round-trip time. The use of an overly-small value for w_q, particularly in an environment of one-way, long-lived traffic, can exacerbate RED's problems with oscillations of the queue size [5]. Again, if the authors of [7] had investigated and thought carefully about a wider range of simulation scenarios in 1993, it would have reduced the amount of work necessary later on. Even now that the default NS parameters have been changed to reasonable values, the effects those parameters had on simulation results should sensitize us to the importance of understanding the models we use.

[2]The parameter max_p gives the packet dropping probability imposed when the average queue size exceeds the maximum threshold.

An evaluation of AQM mechanisms in progress [16] shows that, for many simulation scenarios, all considered mechanisms perform similarly. However, simulation scenarios can be devised that show each mechanism in a bad light. In scenarios with long round-trip times and mostly long-lived flows, RED and Adaptive RED exhibit queue oscillations (see the next section). In scenarios with mostly web traffic, or with changes in the level of congestion over time, the Proportional-Integral Controller (PI) [8] and Random Early Marking (REM) [2] perform badly. Many scenarios with Drop-Tail or Adaptive Virtual Queues (AVQ) [13] give competitive performance in terms of delay-throughput tradeoffs, but also give high packet drop rates. It would be helpful to have more grounding in deciding which models and simulation scenarios were critical to explore, and which are edge cases that were less likely to occur in practice. It is unsettling to feel that one could construct a simulation to show almost anything that one wanted, and that there is so little agreement within the research community about why one chooses to explore one set of simulation scenarios rather than another.

3.3 Active Queue Management: Oscillations

Much research effort in active queue management mechanisms comes down to an implicit disagreement about which simulation scenarios are the most important to address. For example, [14] discusses oscillations with RED in scenarios with one-way, long-lived traffic, while [5] criticizes reliance on such scenarios. Queue oscillations are widely considered a serious potential problem with RED active queue management. However, moderate changes in the traffic mix can strongly affect oscillation dynamics. In particular, adding short-lived flows, reverse-path traffic, and a range of round-trip times—characteristics ubiquitous on the Internet—changes simple oscillations into more complex bursty behavior. This dramatic change highlights the importance of the network model. If we understood better the ways in which different models can affect experiment dynamics, perhaps we would be further along in addressing AQM behaviors.

To illustrate, we examine three simulations with somewhat similar parameter settings, but quite different results in terms of the queue dynamics at the congested link. The simulations share a dumbbell topology with a 15 Mbps, 10 ms congested link with Adaptive RED queue management; they all have similar, small amounts of reverse-path traffic; and they all run for 100 seconds. The simulations differ in their traffic mixes and flow round-trip times.[3] Figures 2 through 4 show, for each simulation scenario, the instantaneous queue size over the second half of the simulation, with the dashed line showing the average queue size estimated by RED.

In Figure 2, traffic consists mostly of 80 long-lived flows with large receiver's advertised windows, and with all round-trip times equal to 240 ms in the absence of queueing delay. This resembles models used in literature on RED oscillations [12], and indeed, although the packet drop rate is 2.8% over the second half of the simulation and the link utilization over the second half is also good at 98.6%, oscillations in the instantaneous queue size are quite pronounced.

Traffic observed at Internet routers, however, tends to exhibit a wide range of round-trip times, including relatively short round trip times ($<$ 50 ms) [1, 11]. Figure 3 changes the model of Figure 2 by introducing a wide range of round-trip times, which now vary between 20 and 460 ms. Stable oscillations in queue size have been replaced by more irregular behavior. The simulation might actually be used to argue that oscillations are not a problem on the Internet, because of the absence of regular oscillations of the queue size. The

[3]This scenario was adapted from [5, Section 5.1].

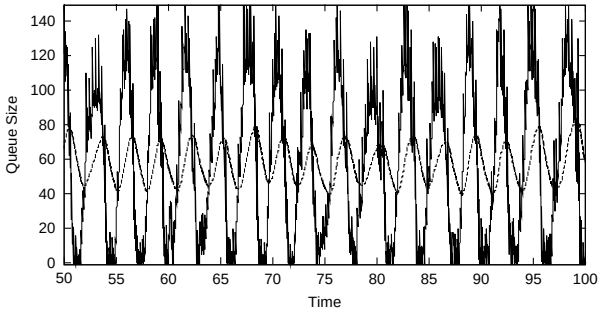


FIGURE 2—Long-lived traffic, 240 ms RTTs.

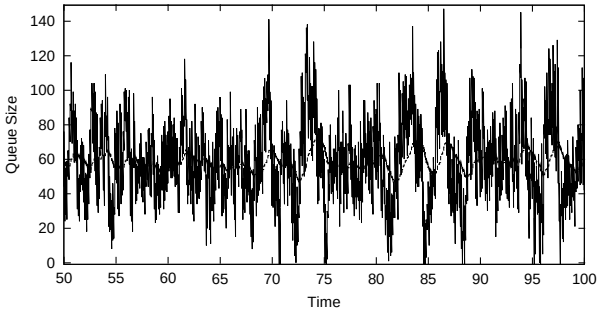


FIGURE 3—Long-lived traffic, 20–460 ms RTTs.

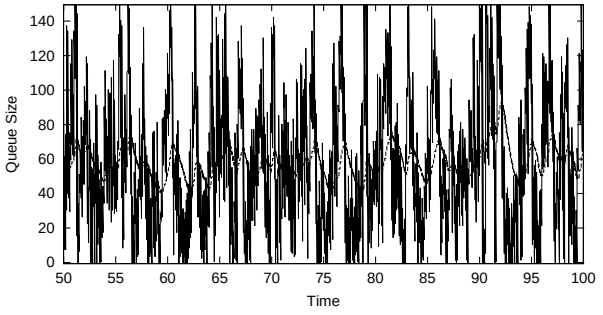


FIGURE 4—Mostly web traffic, 20–460 ms RTTs.

packet drop rate is now 4.6% over the second half (higher because of the influence of flows with very short round-trip times), and link utilization over the second half is now 99.9%.

But the traffic in Figure 3 still consists of all long-lived flows, while most flows on the Internet tend to have short lifetimes [4]. Figure 4 therefore introduces shorter-lived flows into the mix: traffic now mostly comes from the web traffic generator in NS, with a smaller number of long-lived flows (fifteen). The demand from the web traffic generator was chosen to give roughly same packet drop rate as Figure 2 over the second half of the simulation, in this case of 2.6%; the link utilization over the second half is also good, at 98.9%. The queue dynamics and the distribution of queuing delay are rather different, however. The queue size varies more extremely than in Figure 3, and unlike that simulation, the average queue size also varies significantly.

To some extent, we have lacked tools for evaluating the models our simulations actually use. For instance, do the round-trip times seen on the congested link in Figure 4 correspond to the full range we expect, and does that range correspond in a meaningful way to measured Internet data? It turns out that simple mechanisms can enable evaluation of aspects of a simulation's model.

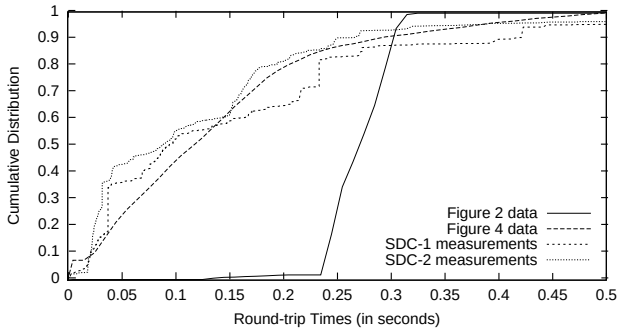


FIGURE 5—Distributions of packet round-trip times on the congested link of two simulations, with data measured on the Internet for comparison.

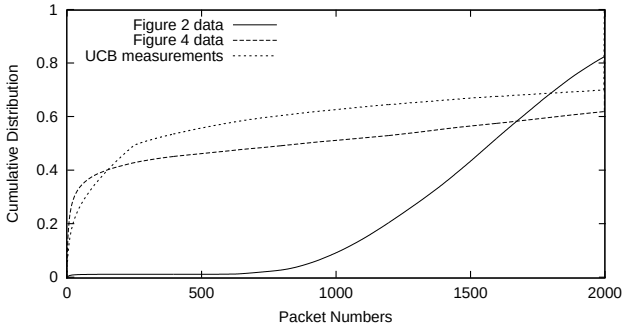


FIGURE 6—Distributions of packet numbers on the congested link over the second half of two simulations, with data measured on the Internet for comparison.

Figure 5 shows one way to evaluate the range of round-trip times in a simulation. We added mechanisms to the NS simulator to record the simulated TCP senders' estimated round-trip times for packets on the congested link. The figure shows a cumulative per-packet distribution of these measurements. It clearly demonstrates Figure 2's narrow range of round-trip times, from 240 to 310 ms, and Figure 4's much wider range, from almost 0 to more than 500 ms.[4] We have also included two representative measurements of an OC3 access link at UC San Diego, calculated by Jiang and Dovrolis using a passive TCP round-trip time estimator [11, Figure 13]. We used these measurements to guide our setting of link propagation delays, and as a result Figure 4 matches the measurements far better than Figure 2. Note that although the average round-trip time of a TCP connection in Figure 4 is still 240 ms in the absence of queueing delay, most of the packets come from the TCP connections with shorter round-trip times, as one would expect.

In order to better evaluate the mix of connection lengths in a simulation, we also added mechanisms to NS to record packet numbers seen on the congested link. The first packet sent in a flow is numbered 1, as are any of its retransmissions. The next packet is numbered 2, and so forth. Thus, graphing a cumulative distribution of packet numbers shows the fraction of packets sent during connection startup (slow start). This quantity is largely determined by the distribution of flow sizes in the simulation, but has independent interest.

Figure 6 show the cumulative distribution of packet numbers for the simulations in Figures 2 and 4, as well as from a July 2000

[4]Very short round-trip times are from the first packet in each connection, which reports an estimated round-trip time of 0 ms.

trace of wide-area traffic to and from UC Berkeley.[5] We used these measurements, in part, to guide our setting of the relative number of web sessions and of long-lived flows. As Figure 6 shows, almost all the packets in the second half of Figure 2's simulation were at least the 500th packet in their respective flows. This means there were no slow-start dynamics in that part of that simulation. In contrast, short-lived flows in Figure 4 gave rise to a substantial number of packets with small packet numbers in the second half of the simulation. The corresponding increase in slow-start dynamics probably influenced the simulation results.

These simulations raise the question of which is more important to explore, the pronounced oscillations in a scenario with long-lived flows all with the same round-trip time, or the variability of demand over shorter time scales that comes from a traffic mix and round-trip time distribution closer to that observed on real links? It is not obvious that the mechanisms proposed to address the oscillations in Figure 2 also perform well in scenarios with more diverse traffic (as in Figure 4), or in other scenarios that more stringently stress the responsiveness of the underlying queue management.

3.4 TCP Variants

It is not just AQM research that suffers from modeling issues. As examples of transport protocols, we show below how the designs of several TCP variants were influenced by implicit network models. In the case of Reno TCP [10], the model has proved false, and as a result Reno TCP has terrible performance in some scenarios that are common in practice. In the case of Vegas TCP [3], we aren't sure how frequently the underlying model applies in practice, making evaluation difficult.

Reno TCP added Fast Recovery to TCP in 1990, following Jacobson's introduction of congestion control in Tahoe TCP in 1988 [9]. Fast Recovery makes a key contribution of allowing the TCP sender to avoid slow-starting in response to congestion—with Fast Recovery, the TCP sender halves its congestion window and avoids a slow-start. Reno TCP works well when only one packet is dropped from a window of data, but generally requires a Retransmit Timeout, and the attendant slow-start, when multiple packets are dropped from a window. This response would be perfectly appropriate if single packet drops were the typical occurrence, and multiple packet drops in a window of data in fact represented more serious congestion calling for a more serious congestion control response. Unfortunately, this is not the case; losses often come in bursts, particularly with Drop-Tail queue management, and Reno TCP responds to those bursts with long timeouts. Reno TCP's attendant performance problems led to a spate of papers proposing a range of mechanisms in the network to reduce multiple packet drops from a window of data, while better models—for instance, including the typical burstiness of flows slow-starting at different times—might have prevented Reno's performance problems with multiple packet drops in the first place. It is straightforward to modify Fast Recovery to avoid Reno's unnecessary Retransmit Timeouts, as illustrated by later TCP variants such as NewReno TCP, which fixes this bug.

As a second example of how limitations in modeling assumptions affect transport design, we consider Vegas TCP [3]. Vegas is optimized for environments with very low levels of statistical multiplexing (e.g., only a few active TCP connections), where the sending rate of an individual TCP connection strongly affects the queue size at the router. In such a scenario, increases in the congestion window past its optimal size only increase the queueing delay, rather than increasing the connection's sending rate. Thus, once increased queueing delay is detected, Vegas TCP refrains from further increases in the congestion window.[6] However, under different models—with higher levels of statistical multiplexing, for example, where the queueing delay and packet drop rate experienced by a connection have very little to do with the sending rate of that flow—Vegas TCP performs significantly worse than in the environment with small-scale statistical multiplexing.

We actually know very little about where Internet congestion occurs, or where it can be expected to occur in the future. Are congested links lower-bandwidth access links with low levels of statistical multiplexing, or high-bandwidth transoceanic links with high levels of statistical multiplexing, or both (as would seem to be the case)? What are typical levels of congestion, or of packet reordering, or of packet corruption? The more we know about the range of realistic network conditions, and of how this range might be changing over time, the better we can make informed choices in our design of transport protocols.

4 MOVING FORWARD: A PROPOSAL

Researchers could conceivably use existing measurements and analysis methodologies to understand the models they use for their simulations. Unfortunately, those measurements and methodologies have never been synthesized into a convenient, coherent whole. We lack an agreed-upon set of best modeling practices, partially because we have not yet recognized that creating such best practices is a legitimate research goal in its own right.

We hope this paper helps broaden discussion within the research community about the models we use. In addition, we have laid out a path for our own research that leads towards more relevant Internet models. The rest of this section lays out that path in outline.

We intend to begin with specific research questions, such as questions around congestion-related mechanisms at router queues. Analysis of the research questions will lead to a description of the experimental parameters relevant for constructing models. Sections 3.2 and 3.3, for example, showed that bottleneck link bandwidth, the range of expected round-trip times of flows on the link, and the range of flow lengths are all relevant parameters for AQM models.

Next, simulation experiments will show how parameter settings affect the observed behavior of existing techniques. Relevant experiments will be based on published research in the area. For settings that do affect behavior, new measurement studies and analysis of the measurement literature will describe how the settings look on the real Internet.

We will distill this work into a set of best practices for model construction. This may include ready-made simulation setups, papers, RFC-like documents, and so forth. We eventually hope to facilitate the creation of a shared repository of models and simulation scenarios for use by all.

Of course, changes in the network might expose the importance of different parameters. Our work will not determine the complete set of interesting simulations for a research area. Rather, it will point out the parameters that have proved important in the past, provide observations of their values on the Internet, and describe expected effects of other values.

Finally, we will make the measurement programs we borrow, or create, available to the research community as high-quality, maintained software tools. This will make it easy for the community to keep the best-practice models up to date with changing Internet

[5]The per-byte distribution of packet numbers was calculated from a list of connections, along with the total number of packets and of bytes for each connection, derived by Ratul Mahajan from a July 2000 trace file of wide-area traffic to and from UC Berkeley.

[6]Vegas TCP can be seen in part as a reaction to the poor performance of Reno TCP in the presence of multiple packet drops.

conditions.

Note that while we welcome collaborators, we don't think we've found the only, or even necessarily the right, approach. More important is to address the problem itself: the need for better models in Internet research.

5 CONCLUSIONS

In summary:

– Network research, and Internet research in particular, has a great need for better models, and for better common evaluation of models.

– Specific research problems require their own models—problem- or application-driven modeling, rather than global Internet modeling.

– We need a better understanding of exactly which aspects of models are critical for a particular research issue.

– Models must be based on network measurement when necessary.

– We want models that apply to the Internet of the future, as well as to the Internet of today.

– We have some ideas that we plan to put into practice, but this project can only flourish with the commitment of the research community as a whole.

The simulation scenarios we used to generate figures in this paper may be found at `http://www.icir.org/models/sims.html`.

ACKNOWLEDGMENTS

We gratefully thank Hao Jiang and Constantinos Dovrolis for generously providing the data on measured round-trip-time distributions shown in our Figure 5, and Ratul Mahajan for help creating the data on measured packet-number distributions shown in our Figure 6. We also thank anonymous reviewers for comments on previous drafts, and the National Science Foundation for funding our future work on this research program.

REFERENCES

[1] M. Allman. A Web Server's View of the Transport Layer. *ACM Computer Communication Review*, 30(5), Oct. 2000.

[2] S. Athuraliya, S. Low, and D. Lapsley. Random Early Marking. In *QofIS 2000*, Sept. 2000.

[3] L. Brakmo, S. O'Malley, and L. L. Peterson. TCP Vegas: New Techniques for Congestion Detection and Avoidance. In *SIGCOMM 1994*, Aug. 1994.

[4] C. Cunha, A. Bestavros, and M. Crovella. Characteristics of WWW Client-based Traces, 1995. Boston University Technical Report BUCS-95-010, 1995.

[5] S. Floyd, R. Gummadi, and S. Shenker. Adaptive RED: An Algorithm for Increasing the Robustness of RED's Active Queue Management, Aug. 2001. `http://www.icir.org/floyd/papers/adaptiveRed.pdf`.

[6] S. Floyd and V. Jacobson. On Traffic Phase Effects in Packet-Switched Gateways. *Internetworking: Research and Experience*, 3(3), Sept. 1992.

[7] S. Floyd and V. Jacobson. Random Early Detection Gateways for Congestion Avoidance. *IEEE/ACM Trans. on Networking*, 1(4):397–413, Aug. 1993.

[8] C. Hollot, V. Misra, D. Towsley, and W. B. Gong. On Designing Improved Controllers for AQM Routers Supporting TCP Flows. In *INFOCOM 2001*, Apr. 2001.

[9] V. Jacobson. Congestion Avoidance and Control. In *SIGCOMM 1988*, Aug. 1988.

[10] V. Jacobson. Dynamic Congestion Avoidance / Control (long message). Email message, 11 Feb. 1988. `http://www-nrg.ee.lbl.gov/nrg-email.html`.

[11] H. Jiang and C. Dovrolis. Passive Estimation of TCP Round-Trip Times. *ACM Computer Communication Review*, 32(3), July 2002.

[12] K. B. Kim and S. H. Low. Analysis and Design of AQM for Stabilizing TCP. Technical Report caltechCSTR:2002.009, Caltech, Mar. 2002.

[13] S. Kunniyur and R. Srikant. Analysis and Design of an Adaptive Virtual Queue (AVQ) Algorithm for Active Queue Management. In *SIGCOMM 2001*, pages 123–34, Aug. 2001.

[14] V. Misra, W. Gong, and D. Towsley. Fluid-based Analysis of a Network of AQM Routers Supporting TCP Flows with an Application to RED. In *SIGCOMM 2000*, Aug. 2000.

[15] NS project. The Network Simulator—ns-2 (Web site). `http://www.isi.edu/nsnam/ns/`.

[16] J. Padhye, S. Floyd, and S. Shenker. Simulation-based Comparison of AQM Schemes. Work in progress, citation for acknowledgement purposes only.

[17] V. Paxson and S. Floyd. Difficulties in Simulating the Internet. *IEEE/ACM Trans. on Networking*, 9(4):392–403, Aug. 2001.

[18] Utah Emulab project. Emulab.net, the Utah network testbed (Web site). `http://www.emulab.net/`.

[19] A. Veres and M. Boda. The Chaotic Nature of TCP Congestion Control. In *INFOCOM 2000*, Mar. 2000.

[20] E. Zegura. New Directions and Half-Baked Ideas in Topology Modeling. viewgraphs, IPAM workshop, June, 2002, `http://www.cc.gatech.edu/projects/joint/pubs/IPAM-concluding-workshop.ppt`.

The Scaling Hypothesis: Simplifying the Prediction of Network Performance using Scaled-down Simulations

Konstantinos Psounis[†], Rong Pan[†], Balaji Prabhakar[†], Damon Wischik[‡]
[†] Stanford Univerity, [‡] Cambridge University
Email: kpsounis, rong, balaji@stanford.edu, D.J.Wischik@statslab.cam.ac.uk

ABSTRACT

As the Internet grows, so do the complexity and computational requirements of network simulations. This leads either to unrealistic, or to prohibitely expensive simulation experiments.

We explore a way to side-step this problem, by combining simulation with sampling and analysis. Our hypothesis is this: if we take a sample of the traffic, and feed it into a suitably scaled version of the system, we can extrapolate from the performance of the scaled system to that of the original.

We find that when we scale a network which is shared by TCP-like flows, and which is controlled by a variety of active queue management schemes, then performance measures such as queueing delay and the distribution of flow transfer times are left virtually unchanged. Hence, the computational requirements of network simulations and the cost of experiments can decrease dramatically.

1. INTRODUCTION

Measuring the performance of the Internet and predicting its behavior under novel protocols and architectures are important research problems. These problems are made difficult by the sheer size and heterogeneity of the Internet: it is very hard to simulate large networks and to pinpoint aspects of algorithms and protocols relevant to their behavior. This has prompted work on traffic sampling [1, 2]. Sampling certainly reduces the volume of data, although it can be hard to work backwards—to infer the performance of the original system.

A direct way to measure and predict performance is with exhaustive simulation: If we record the primitive inputs to the system, such as session arrival times and flow types, we can in principle compute the full state of the system. Further, through simulation we can test the behavior of the network under new protocols and architectures. But such large-scale simulation requires massive computing power.

Reduced-order models can go some way in reducing the burden of simulation. In some cases [3, 13] one can reduce the dimensionality of the data, for example by working with traffic matrices rather than full traces, while retaining enough information to estimate the state of the network. The trouble is that this requires careful traffic characterization and model-building. The heterogeneity of the Internet makes this time-consuming and difficult, since each scenario might potentially require a different new model.

In this paper we explore a way to reduce the computational requirements of simulations and the cost of experiments, and hence simplify network measurement and performance prediction. We do this by combining simulations with sampling and analysis. Our basic hypothesis, which we call SHRiNK[1], is this: if we take a *sample* of the traffic, and feed it into a *suitably scaled* version of the system, we can *extrapolate* from the performance of the scaled system to that of the original.

This has two benefits. First, by relying only on a sample of the traffic, SHRiNK reduces the amount of data we need to work with. Second, by using samples of actual traffic, it shortcuts the traffic characterization and model-building process while ensuring the relevance of the results.

This approach also presents challenges. At first sight, it appears optimistic. Might not the behavior of a large network with many users and higher link speeds be intrinsically different to that of a smaller network? Somewhat surprisingly we find that, in several essential ways, one can mimic a large network using a suitably scaled-down version. The key is to find suitable ways to scale down the network and extrapolate performance.

The outline of the paper is as follows: In Section 2 we study the scaling behavior of an IP-network whose traffic consists of long-lived TCP-like flows arriving in clusters. Networks with such traffic have been used in the literature to test the behavior of control algorithms and queue management schemes. Using simulations and theory we find that when such a network is suitably scaled, performance measures such as queueing delay and drop probability are left virtually unchanged. In Section 3 we study IP networks at which flows arrive at random times (i.e. unclustered) and whose sizes are heavy-tailed. Such networks are representative of the Internet. We find that a different scaling to that in Section 2 leaves the distribution of the number of active flows and of their normalized transfer times unchanged. A simple theoretical argument reveals that the method we suggest for "SHRiNKing" networks in which flows arrive at random times will be widely applicable (i.e. for a variety of topologies, flow transfer protocols, and queue management schemes). By contrast, we find that the theoretical underpinning for SHRiNKing networks at which flows arrive in clusters depends on the type of queue management scheme

[1]SHRiNK: Small-scale Hi-fidelity Reproduction of Network Kinetics

used at the routers.

A word about the organization of the paper: Space limitations have necessitated a selective presentation of the material. We have chosen to describe the method in detail. The theoretical complement and the validation using simulations are abbreviated. More details can be found in a longer version of the paper [10].

2. SCALING BEHAVIOR OF IP NETWORKS WITH LONG-LIVED FLOWS

In this section we explore how SHRiNK applies to IP networks used by long-lived TCP-like flows that arrive in clusters, and controlled by queue management schemes like RED.

First, we explain in general terms how we sample traffic, scale the network, and extrapolate performance.

Sampling is simple. We sample a proportion α of the flows, independently and without replacement.

We scale the network as follows: link speeds and buffer sizes are multiplied by α. The various AQM-specific parameters are also scaled, as we will explain in the following section 2.1. The network topology is unchanged during scaling. In the cases we study, performance measures such as average queueing delay are virtually the same in the scaled and the unscaled system.

Our main theoretical tool is the recent work on fluid models for TCP networks [8]. While [8] shows these models to be reasonably accurate in most scenarios, the range of their applicability is not yet fully understood. However, in some cases the SHRiNK hypothesis holds even when the fluid model is not accurate, as shown in Section 2.1.2.

2.1 RED

The key features of RED are the following two equations, which together specify the drop (or marking) probability. RED maintains a moving average q_a of the instantaneous queue size q; and q_a is updated whenever a packet arrives, according to the rule

$$q_a := (1-w)q_a + wq,$$

where w is a parameter that determines the size of the averaging window. The average queue size determines the drop probability p, according to the equation

$$p_{\text{RED}}(q_a) = \begin{cases} 0 & \text{if } q_a < min_{th} \\ p_{max}\left(\frac{q_a - min_{th}}{max_{th} - min_{th}}\right) & \text{if } min_{th} \le q_a < max_{th} \\ 1 & \text{if } q_a > max_{th} \end{cases} \quad (1)$$

We scale the parameters p_{max}, min_{th}, max_{th} and w as follows: min_{th} and max_{th} are multiplied by α; p_{max} is fixed at 10%; the averaging parameter w is multiplied by α^{-1}. The reason of choosing these parameters will become clear later in Section 2.1.1.

The Basic Setup

We consider two congested links in tandem, as shown in Figure 1. There are three routers, $R1$, $R2$ and $R3$; and three groups of flows, $grp1$, $grp2$, and $grp3$. The link speeds are 100Mbps and the buffers can hold 8000 packets. The RED parameters are $min_{th} = 1000$, $max_{th} = 2500$ and $w = 0.000005$. For the flows: $grp0$ consists of 1200 TCP flows each having a propagation delay of 150ms, $grp1$ consists of 1200 TCP flows each having a propagation delay of 200ms, and $grp2$ consists of 600 TCP flows each having a propagation delay of 250ms. Note that 75% of $grp0$ flows switch off at time 150s.

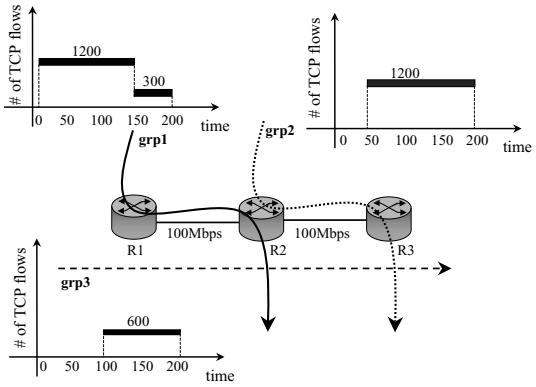


Figure 1: Basic network topology and flow information

This network is scaled-down by factors $\alpha = 0.1$ and 0.02, and the parameters are modified as described above.

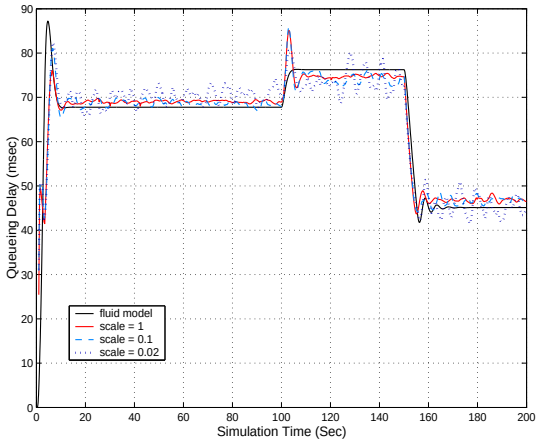


Figure 2: Basic Setup: Average Queueing Delay at Q1

We plot the average queueing delay at Q1 in Figure 2. The drop probability at Q1 is shown in Figure 3. Due to a lack of space, we omit the plot of the average queueing delay and drop probability for Q2 whose behavior is similar to those of Q1. We see that *the queueing delay and the drop probabilities are almost identical at different scales.* We draw attention to two features which we shall comment upon later: (i) The transient behaviors (e.g. the overshoots and undershoots) are quite well-mimicked at the smaller scales, and (ii) the variability increases as the scale reduces.

Since the queueing dynamics and drop probabilities essentially remain the same, the dynamics of the TCP flows are also unchanged. In other words, an individual flow which survives the sampling process essentially cannot tell whether it is in the scaled or the unscaled system.

We have also tested the case where the scale is $\alpha = 0.01$. In this case, only three flows in $grp1$ are present during the period 150s to 200s. Hence the sample is too meager to reproduce the queueing dynamics. Certainly, 1% is not the limit of the scaling hypothesis *in general*. Further study needs to be conducted to find out theoretically where this limit is.

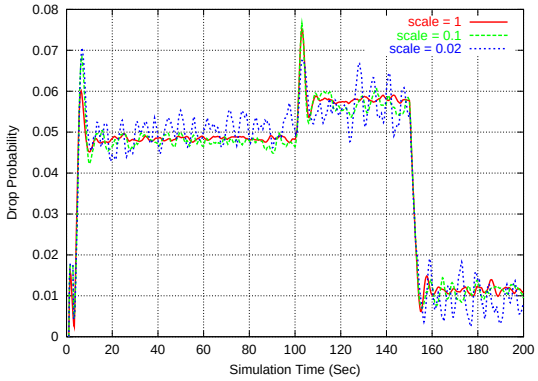


Figure 3: Basic Setup: Drop Probability at Q1

2.1.1 With Faster Links

Suppose we alter the basic setup, by increasing the link speeds to 500Mbps, while keeping all other parameters the same. Figure 4 (zoomed in to emphasize the point) illustrates that, once again, scaling the network does not alter the queueing delay at Q1 (Q2 shows the same scaling behavior). Note that high link speeds cause the queue to oscillate. There have been various proposals for stabilizing RED [5, 9]. We are not concerned with stabilizing RED here: we mention this case to show that SHRiNK can work whether or not the queue oscillates.

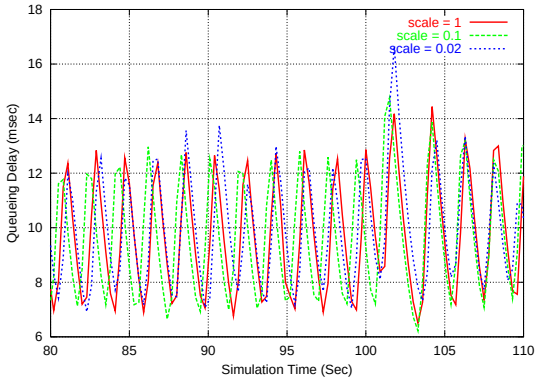


Figure 4: With faster links: Average queueing delay at Q1 (zoomed in)

Theory

We now show that these simulation results are supported by the fluid model of TCP/RED [8].

Consider N flows sharing a link of capacity C. Let $W_i(t)$ and $R_i(t)$ be the window size and round-trip time of flow i at time t. Here $R_i(t) = T_i + q(t)/C$, where T_i is the propagation delay for flow i and $q(t)$ is the queue size at time t. Let $p(t)$ be the drop probability and $q_a(t)$ the average queue size at time t.

The fluid model describes how these quantities evolve; or rather, since these quantities are random, the fluid model describes how their expected values evolve. Let $\bar{X}$ be the expected value of random variable X. Then the fluid model equations are:

$$\frac{d\bar{W}_i(t)}{dt} = \frac{1}{R_i(\bar{q}(t))} - \frac{\bar{W}_i(t)\bar{W}_i(t-\tau_i)}{1.5R_i(\bar{q}(t-\tau_i))}\bar{p}(t-\tau_i) \quad (2)$$

$$\frac{d\bar{q}(t)}{dt} = \sum_{i=1}^{N} \frac{\bar{W}_i(t)}{R_i(\bar{q}(t-\tau_i))} - C \quad (3)$$

$$\frac{d\bar{q}_a(t)}{dt} = \frac{\log(1-w)}{\delta}\bar{q}_a(t) - \frac{\log(1-w)}{\delta}\bar{q}(t) \quad (4)$$

$$\bar{p}(t) = p_{\text{RED}}(\bar{q}_a(t)) \quad (5)$$

where $\tau_i = \tau_i(t)$ solves $\tau_i(t) = R_i(\bar{q}(t-\tau_i(t)))$, δ is the average packet inter-arrival time, and p_{RED} is as in (1) [2]. Suppose we have a solution to these equations

$$\big(\bar{W}_i(\cdot), \bar{q}(\cdot), \bar{q}_a(\cdot), \bar{p}(\cdot)\big).$$

Now, suppose the network is scaled and denote by C', N', etc., the parameters of the scaled system. When the network is scaled, the fluid model equations change, and so the solution changes. Let $\big(\bar{W}_i'(\cdot), \bar{q}'(\cdot), \bar{q}_a'(\cdot), \bar{p}'(\cdot)\big)$ be the solution of the scaled system. It can be theoretically verified (but we do not do this here due to lack of space) that

$$\big(\bar{W}_i'(\cdot), \bar{q}'(\cdot), \bar{q}_a'(\cdot), \bar{p}'(\cdot)\big) = \big(\bar{W}_i(\cdot), \alpha\bar{q}(\cdot), \alpha\bar{q}_a(\cdot), \bar{p}(\cdot)\big),$$

which means the queueing delay $\bar{q}'/C' = \alpha\bar{q}/\alpha C$ is identical to that in the unscaled system. The drop probability is also the same in each case, i.e. $\bar{p}(t) = \bar{p}'(t)$. Thus, we will have theoretical support for the observations in the previous section.

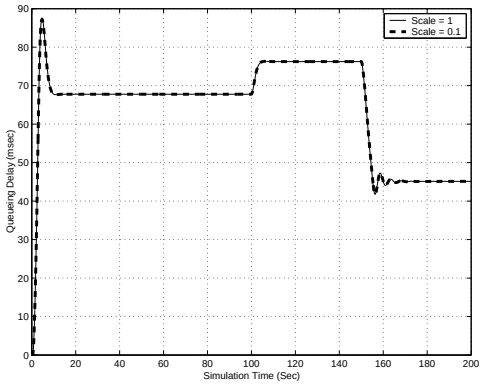


Figure 5: Fluid model predicts scaling behavior

Figure 5 presents the solution of the fluid model for the queueing delay at Q1 under the scenario of Figure 1 for the scale parameters $\alpha = 1$ and 0.1. As can be seen, both the solutions are virtually identical, illustrating the scaling property of the differential equations mentioned above.

2.1.2 When the theory is not appropriate

Suppose we alter the basic setup, by decreasing the link speeds to 50Mbps, while keeping all other parameters the same. Once again, scaling the network does not alter the queueing delay. Due to limitations of space we omit the corresponding plot. For such a simulation scenario, especially in the time frame 100sec-150sec, the fluid model is not a good fit as shown

[2]We have the constant 1.5 in (2), not 2 as in [8]. This change improves the accuracy of the fluid model; due to limited space we omit the derivation.

in [12] and verified by us via simulations: actual window and queue sizes are integer-valued whereas fluid solutions are real-valued; rounding errors are non-negligible when window sizes are small as is the case here. The range of applicability of the fluid model is not our primary concern in this paper: we mention this case to show that SHRiNK can work whether or not the fluid model is appropriate.

Figure 6: DropTail: Average queueing delay at Q2

2.2 DropTail

While all the simulations above show the validity of SHRiNK, the scaling behavior does not hold when we change the queue management scheme to DropTail. Figure 6 shows the average queueing delay at Q2. Clearly, the queueing delays for different scale do not match. This scheme drops all the packets that arrive at a full buffer. As a result, it could cause a number of consecutive packets to be lost. These bursty drops underlie the reason the scaling hypothesis fails in this case [11]. Besides, when packet drops are bursty and correlated, the assumption that packet drops occur as a Poisson process (see [8]) is violated and the differential equations become invalid.

2.3 Summary

Besides the examples we have studied in this section, we have also validated SHRiNK with heterogeneous end-systems (TCP, general AIMD and MIMD protocols, UDP, HTTP), with a variety of active queue management policies such as the PI controller [6] and AVQ [7], with a range of system parameters, and with a variety of network topologies (tandems, stars, meshes). We have found that, in cases where TCP-like flows are long-lived and drops are not bursty, basic performance measures such as queueing delay are left unchanged, when we sample the input traffic and scale the network parameters in proportion.

3. SCALING BEHAVIOR OF IP NETWORKS WITH SHORT AND LONG FLOWS

It has been shown that the size distribution of flows on the Internet is heavy-tailed [14]. Hence, Internet traffic consists of a large fraction of short and and a small fraction of long flows. It has been observed that sessions arrive as a Poisson process[3]. In this section we take these observations into account and study the scaling behavior of IP networks carrying heavy-tail distributed, Poisson flows. Our finding is that with a somewhat different scaling than in the previous section, the distributions of a large number of performance measures, such as the number of active flows and the delay of flows, remain the same.

3.1 Simulations

We perform simulations using ns-2 for the same topology as in Figure 1. There are three routers, $R1$, $R2$ and $R3$, two links in tandem, and three groups of flows, $grp1$, $grp2$, and $grp3$. The link speeds are 10Mbps. We present simulations with both RED and DropTail. The RED parameters are $min_{th} = 100$, $max_{th} = 250$ and $w = 0.00005$. When using DropTail, the buffer can hold 200 packets.

Within each group flows arrive as a Poisson process with some rate λ. We vary λ to study both uncongested and congested scenarios. (We use the ns-2 built-in routines to generate sessions consisting of a single object each. This is what we call a flow.) Each flow consists of a Pareto-distributed number of packets with average size 12 packets and shape parameter equal to 1.2. The packet size is set to 1000 bytes. The propagation delay of each flow of $grp0$, $grp1$, and $grp2$, is 50ms, 100ms, and 150ms respectively.

Sampling and Scaling

The heavy-tailed nature of the traffic makes sampling a bit more involved than before, because a small number of very large flows has a large impact on congestion. To guarantee that we sample the correct number of these flows, we separate flows into large (elephants) and small (mice) and sample exactly a proportion α of each.

Scaling the system is slightly different from Section 2. As before, we multiply by α the link speeds. However, we do not scale the buffer sizes or the RED thresholds. Further, we multiply by α^{-1} the propagation delay of each flow[4]. We will elaborate on the intuition and theory behind these choices after we present the simulation results.

Since we sample flows which arrive at random times and have random sizes, quantities like the queueing delay cannot be expected to scale as functions of time. However, simulations and theory show that we can exhaustively compare the *distributions* of related quantities.

We run the experiments for scale factors $\alpha = 1$ and 0.1, and compare the distribution of the number of active flows as well as the histogram of the normalized delays of the flows in the original and the scaled system. (The normalized delays are the flow transfer times multiplied by α.) We will also compare more detailed performance measures such as the distribution of active flows that are less than some size and belong to a particular group. Due to limitations of space we will not present results when the links are uncongested, but only compare distributions for the more interesting case where drops occur. (The performance of uncongested networks also scale.) The flow arrival rate is set to be 60 flows/sec within each group. The results don't depend on whether the rates are larger or smaller.

Simulation Results

We will start by comparing distributions when RED is used.

[3]Further, for certain models of TCP bandwidth sharing, the equilibrium distribution of the number of flows in progress is *as if* flows arrive as a Poisson process, not just sessions [4].

[4]One should also multiply by α^{-1} the various protocol timeouts. In practice, since it is very rare for a timeout to expire, leaving timeouts unscaled does not affect the results.

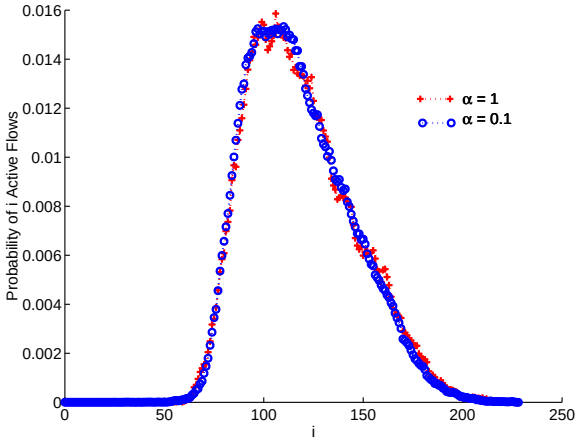


Figure 7: Distribution of number of active flows on the first link.

Figure 7 plots the distribution of the number of active flows in the first link between routers $R1$ and $R2$. It is evident from the plot that the two distributions match. A similar scaling holds for the second link.

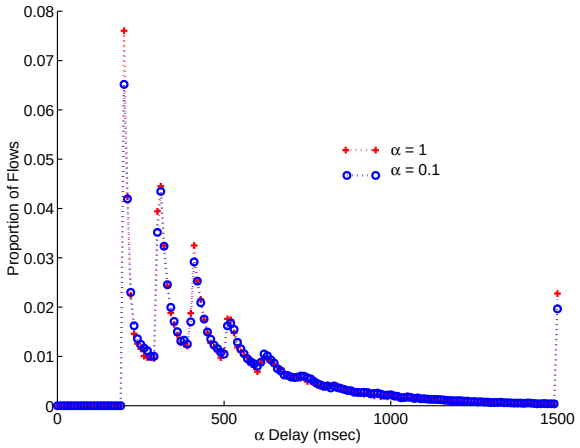


Figure 8: Histogram of normalized delays of *grp*0 flows.

Figure 8 plots the histogram of the flow transfer times (delays) of the flows of *grp*0 multiplied by α. To generate the histogram, we use delay chunks of $\frac{10}{\alpha}$ms each. There are 150 such delay chunks in the plot, corresponding to flows having a delay of 0 to $\frac{10}{\alpha}$ms, $\frac{10}{\alpha}$ms to $\frac{20}{\alpha}$ms, and so on. The last delay chunk is for flows that have a delay of at least $\frac{1500}{\alpha}$ms. It is evident from the plot that the distribution of the normalized delays match. The results for the other two groups of flows are also the same. The peaks in the delay plot are due to the TCP slow-start mechanism. The left-most peak corresponds to flows which send only one packet that face no congestion, the portion of the curve between the first and second peaks corresponds to flows which send only one packet but face congestion (but no drops), the next peak corresponds to flows which send two packets and face no congestion, and so on.

We will now investigate if distributions scale when DropTail is used.

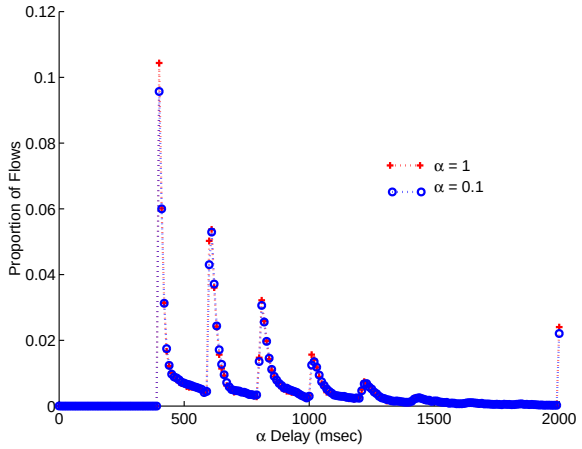


Figure 9: Histogram of normalized delays of *grp*1 flows when DropTail is used.

Figure 9 plots the histogram of the flow transfer times of the flows of *grp*1 multiplied by α, when routers employ DropTail. The distributions of the normalized delays match as before. Although not shown here, the distribution of the number of active flows also scales under DropTail.

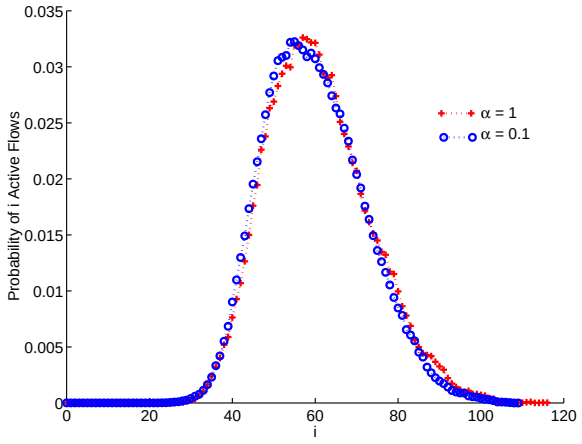


Figure 10: Distribution of number of active *grp*2 flows with size less than 12 packets.

What about more detailed performance measures? As an example, we compare the distribution of active flows belonging to *grp*2 that are less than 12 packets long. The AQM scheme used at the routers is RED (DropTail behaves similarly). Figure 10 compares the two distributions from the original and scaled system. Again, the plots match.

3.2 Theoretical support

The above results can be theoretically supported. First, consider a simplified model: suppose that flows arrive as a Poisson process, and that the service time for each flow is independent and drawn from some common distribution (perhaps heavy-tailed). This is known in queueing theory as an M/GI model.

Suppose also that the service capacity of the link is shared between currently active flows according to the classic equation for TCP throughput. (We will shortly extend the argument to allow for a detailed packet-level model of the link.) This is the sort of flow-level model used in [4].

Let $J(t)$ be the number of jobs in this system at time t. Now scale the process of arriving flows, by multiplying flow interarrival times by $1/\alpha$. Also, multiply the service capacity of the link by α. It is not hard to see that the scaled system looks exactly like the original system, watched in slow motion. Specifically, if $\tilde{J}(t)$ is the number of jobs in the scaled system at time t, then $\tilde{J}(t) = J(\alpha t)$.

Suppose that instead of stretching the flow arrival process in time we had sampled it, retaining each flow independently with probability α. It is a simple but far-reaching property of the Poisson process that these two processes have exactly the same distribution. In particular, if $\hat{J}(t)$ is the number of jobs in the system which is scaled by sampling, then for each t (assuming the queues are in equilibrium), $\hat{J}(t)$ and $\tilde{J}(t)$ have the same distribution, which is the distribution of $J(t)$. That is, the marginal distribution of the number of jobs is the same in the two systems.

This argument does not in fact depend on how the link shares its bandwidth. It could be first-come-first-served, or use priorities. More interestingly, let us instead model the behavior of the link as a discrete event system. Suppose that flows arrive as a Poisson process as before, and that they arrive with a certain number of packets to send, independent and with a common distribution. Consider a time-line showing the evolution in time of the discrete event system. How could we scale the parameters of the system, in order to stretch out the time-line by a factor $1/\alpha$?

To be concrete, suppose that $\alpha = 0.1$, and that a flow with 12 packets takes 1 second to transfer in the original system. We would like to ensure that its transfer time in the scaled system is 10 seconds. We can do this by making sure that each of the 12 packets takes 10 times as long to transfer in the scaled system. Now, the transmission time of a packet is the sum of its queueing delay and propagation delay. We can multiply the queueing delay by 10 by reducing the link speed by a factor of 10; we should also multiply the propagation delay by 10.

In general, we should multiply propagation times by $1/\alpha$, and the service times by the same factor, which means multiplying the service rate by a factor α. As before we would multiply flow interarrival times by $1/\alpha$, which has the same effect as sampling with probability α. Note that this model takes account of retransmissions due to drops (assuming either there are no timeouts, or that the timeout clock is also scaled). Note also that the packet buffer at the link is *not* scaled. The conclusion holds just as before: the marginal distribution of the number of jobs in the system is unchanged.

4. CONCLUSION

In this paper we have presented a method, SHRiNK, to reduce the complexity of network simulations and performance prediction. Our main finding is that when a sample of the network traffic is fed to a suitably scaled replica of the network, performance measures of the original network are accurately predicted by the smaller scale replica. In particular, (i) when long-lived flows arrive in clusters, queueing delays and drop probabilities in the two networks are the same as a function of time in many interesting scenarios, and (ii) when flows arrive at random times and their size is heavy-tailed, the distribution of performance measures under *any* network topology, active queue management mechanism, and transport protocol remains unchanged. We have shown these results using simulations and theory.

Further work consists of validating SHRiNK in large experimental testbeds, obtaining a better understanding of the theory, and trying to extend the approach to web-server farms and to wireless networks.

5. REFERENCES

[1] K. Claffy, G. Polyzos, and H.-W. Braun. Applications of sampling methodologies to network traffic characterization. In *Proceedings of SIGCOMM*, 1993.

[2] C. Estan and G. Varghese. New directions in traffic measurement and accounting. In *Proceedings of ACM SIGCOMM Internet Measurement Workshop*, 2001.

[3] Fluid models for large, heterogeneous networks. `http://www-net.cs.umass.edu/fluid/`, accessed January 2002.

[4] S. Ben Fredj, T. Bonalds, A. Prutiere, G. Gegnie, and J. Roberts. Statistical bandwidth sharing: a study of congestion at flow level. In *Proceedings of SIGCOMM*, 2001.

[5] C.V. Hollot, V. Misra, D. Towlsey, and W. Gong. A control theoretic analysis of RED. In *Proceedings of INFOCOM*, 2001.

[6] C.V. Hollot, V. Misra, D. Towlsey, and W. Gong. On designing improved controllers for AQM routers supporting TCP flow. In *Proceedings of INFOCOM*, 2001.

[7] S. Kunniyur and R. Srikant. Analysis and design of an Adaptive Virtual Queue (AVQ) algorithm for active queue management. In *Proceedings of SIGCOMM*, 2001.

[8] V. Misra, W. Gong, and D. Towsley. A fluid-based analysis of a network of AQM routers supporting TCP flows with an application to RED. In *Proceedings of SIGCOMM*, 2000.

[9] T. Ott, T. Lakshman, and L. Wong. SRED: Stabilized RED. In *Proceedings of INFOCOM*, 1999.

[10] R. Pan, B. Prabhakar, K. Psounis, and D. Wischik. Shrink: A method for scalable performance prediction and efficient network simulation. `http://www.stanford.edu/~kpsounis/scale1.html`, accessed October 2002.

[11] R. Pan, K. Psounis, B. Prabhakar, and M. Sharma. A study of the applicability of the scaling-hypothesis. In *Proceedings of ASCC*, 2002.

[12] S. Shakkottai and R. Srikant. How good are deterministic fluid models of internet congestion control. In *Proceedings of Infocom 2002, to appear*, 2002.

[13] J. Walrand. A transaction-level tool for predicting TCP performance and for network engineering. In *MASCOTS*, 2000. `http://walrandpc.eecs.berkeley.edu/Papers/mascots1.pdf`.

[14] W. Willinger, M.S. Taqqu, R. Sherman, and D.V. Wilson. Self-similarity through high-variability: Statistical analysis of ethernet lan traffic at the source level. *IEEE/ACM Transactions on Networking*, 5(1):71–86, 1997.

Toward an Optimization-Driven Framework for Designing and Generating Realistic Internet Topologies

David Alderson
Management Science and Engineering
Stanford University
alderd@stanford.edu

John Doyle
Control and Dynamical Systems
California Institute of Technology
doyle@cds.caltech.edu

Ramesh Govindan
Computer Science
USC
ramesh@usc.edu

Walter Willinger
AT&T Labs-Research
walter@research.att.com

ABSTRACT

We propose a novel approach to the study of Internet topology in which we use an optimization framework to model the mechanisms driving incremental growth. While previous methods of topology generation have focused on explicit replication of statistical properties, such as node hierarchies and node degree distributions, our approach addresses the economic tradeoffs, such as cost and performance, and the technical constraints faced by a single ISP in its network design. By investigating plausible objectives and constraints in the design of actual networks, observed network properties such as certain hierarchical structures and node degree distributions can be expected to be the natural by-product of an approximately optimal solution chosen by network designers and operators. In short, we advocate here essentially an approach to network topology design, modeling, and generation that is based on the concept of *Highly Optimized Tolerance (HOT)*. In contrast with purely descriptive topology modeling, this opens up new areas of research that focus on the *causal* forces at work in network design and aim at identifying the economic and technical drivers responsible for the observed large-scale network behavior. As a result, the proposed approach should have significantly more predictive power than currently pursued efforts and should provide a scientific foundation for the investigation of other important problems, such as pricing, peering, or the dynamics of routing protocols.

Keywords

Internet topology, network optimization, robustness, complex systems, highly optimized tolerance

1. INTRODUCTION

The Internet is a complex conglomeration of rapidly growing, fragmented, and competing networks. As a research project in its early days, the design, development and deployment of Internet infrastructure was coordinated among relatively few organizations, and the location, capacity, and interconnectivity of this infrastructure was known with relative certainty. However, after the decommissioning of the NSFNET backbone in 1995, when management of the Internet was given over to commercial entities, the number of Internet Service Providers (ISPs) and their corresponding infrastructure have grown dramatically. In this new market environment, a desire to preserve competitive advantage has provided incentives for ISPs to maintain secrecy about their network infrastructure, including detailed connectivity maps.

The resulting decentralized nature of the Internet and the complexity and diversity in the number of infrastructure owners and operators, coupled with the incentive for these providers to obscure their infrastructure topologies,[1] have made comprehensive knowledge of Internet connectivity increasingly difficult. As an example, consider the *AS* graph of the Internet which depicts the connectivity between individual autonomous systems (ASs). In such a graph, each node represents an AS, while a link between two nodes indicates that the two ASs have a "peering relationship" (i.e., there exists at least one direct router-level connection between the two ASs). In this sense, AS graphs reflect business relationships among an ever increasing number of ASs. While AS-level connectivity can, in principle, be inferred from BGP-derived measurements, the fully distributed and decentralized nature of BGP makes it very difficult to obtain "complete" AS-level connectivity [14]. A second type of graph describes Internet connectivity at the router-level. Here, nodes represent individual routers, links represent one-hop connectivity at the IP level between routers, and the resulting graphs reflect physical connectivity at the router-level as seen by IP.[2] While many providers view their router-level maps as containing proprietary and business-sensitive information, reconstructing router-level topologies of individual ASs or ISPs is, in principle, feasible and relies on information obtained from selective traceroute measurements. However, as in the case of AS-level connectivity, the available data are known to provide incomplete router-level maps, and aiming for more complete topologies remains an active area of research [19, 15, 28].

[1]Note that while in the present context the notion of "topology" is almost exclusively used to mean "connectivity" (i.e., links and nodes without further annotation), we use it here—whenever appropriate—to mean "connectivity" plus "resource capacity" (i.e., links and nodes are annotated with for example link speed, delays, router capacity).

[2]See also Section 2.4 regarding Level-2 technologies and our proposed framework.

Gaining a basic understanding of the existing and future Internet topologies and of their evolution over time is of primary importance for networking research. As pointed out in [30], although topology should not affect the *correctness* of networking protocols, it can have a dramatic impact on their *performance*. Topology is therefore important for the design and evaluation of networking protocols, and it may also play an important role in gaining a basic understanding of certain aspects of current large-scale network behavior. As the infrastructure of the Internet continues to evolve, trends in connectivity formation are likely to yield insight into future behavior and may suggest novel strategies for provisioning requirements, traffic engineering, and operations of tomorrow's networks.

The difficult, yet important task of designing, modeling, and generating realistic Internet topologies has attracted a great deal of research attention, especially within the last few years. The prevailing approach of most of these efforts has been to focus on matching a sequence of easily-understood metrics or observed features of interest; e.g., explicitly imposed connectivity properties or hierarchical structures (see [33] and references therein), empirical node degree distributions [21, 23, 1, 7], clustering coefficients [8], etc. (see for example [30] for additional candidate metrics). However, this type of *descriptive* or *evocative* modeling can be misleading, since the question of which metrics or features are the most important ones for judging and comparing different Internet topologies remains largely unresolved, and any particular choice tends to yield a generated topology that matches observations on the chosen metrics but looks very dissimilar on others.

These observations strongly argue for a radically different approach to designing and modeling Internet topologies, one that moves beyond evocative models and instead advocates the careful development and the diligent validation of *explanatory* models that concentrate on the *causal* forces at work in the design and evolution of real topologies. We propose formulating appropriate optimization problems to model the process by which Internet connectivity is established and evolves. The basic idea is that when deploying their infrastructures, network owners and operators are, in fact, approximately solving optimization problems (either explicitly or implicitly) that express the ways in which they build up and evolve their networks. To the extent that we can identify and formulate (even at a high level of abstraction) the objectives and constraints of these optimization problems, solving the latter can be expected to lead to the generation of realistic topologies (i.e., connectivity information as well as resource provisioning information), where the observed characteristics of the resulting graph structures are well explained and understood in terms of the underlying mechanisms that are directly reflected in the optimization formulations.

2. OUR APPROACH IN A NUTSHELL

Our approach seeks to capture and represent realistic drivers of Internet deployment and operation to create a topology generation framework that is inherently *explanatory* and will perforce be *descriptive* as well. Instead of fitting certain characteristics of measured Internet topologies, any such agreements with empirical observations would instead be evidence of a successful explanatory modeling effort. We take as the basic unit of study the solitary ISP since there are a large number of important networking issues—such as configuration, management, pricing, and provisioning—that are naturally and precisely relevant at the level of the ISP. An understanding of the key issues facing ISPs combined with the ability to generate "realistic, but fictitious" ISP topologies would greatly enhance the ability of networking researchers to address these important problems. A second reason for studying the topology of the solitary ISP is that, at an appropriate level of abstraction, the Internet as a whole is simply a conglomeration of interconnected ISPs. Our belief is that by understanding the forces driving topology evolution for the single ISP, one can make great progress to understanding Internet topology at a broader level.

2.1 Driving Forces: Econ and Tech

Our starting premise is that any explanatory framework for Internet topology modeling and generation needs to incorporate both *economic* factors and *technical* factors faced by ISPs. Because of the costly nature of procuring, installing, and maintaining the required facilities and equipment, the ISP is economically constrained in the amount of its physical plant that it can support. Indeed, the economic survival of the ISP depends on carefully balancing limited revenue streams with its capital expenditures. As a result, the buildout of the ISP's topology tends to be incremental and ongoing. At the same time, there are economic aspects of networking that are largely determined by "external" forces, but impose potentially major constraints on the design and evolution of ISP topologies. For example, ignoring economic realities (e.g. most customers reside in the big cities, most high-bandwidth pipes are found between big cities, or most national or global ISPs peer for interconnection in the big cities) can be expected to result in topologies that are too generic to be of practical value.

Similarly, the layout of ISP topologies reflects technical constraints imposed by physical or hardware realities. For example, while routers can only be directly connected to a limited number of neighboring routers due to the limited number of interfaces or line cards they allow, no such limitations exist per se when it comes to the number of peering relationships an ISP can enter in with competing ISPs. Other tech-based factors that may seriously constrain the interconnectivity of ISP topologies are certain Level 2 technologies (e.g., Sonet, ATM, WDM) or the availability and location of dark fiber. Collectively, these economic and technical factors place bounds on the network topologies that are feasible and actually achievable by ISPs.

Given that we require our approach to be driven by economic and technical factors, we next assume that many of these factors can be effectively captured and represented in a mathematical model using a combinatorial optimization framework. A challenge in using such a framework is to demonstrate that some of the crucial economic or technical factors can indeed be expressed in terms of some sort of combinatorial optimization problem. We also need to identify the appropriate optimization formulations (objectives, constraints, parameters) for representing a range of ISP behavior, from very generic to highly specific. Finally, we have to illustrate with convincing examples how knowing

the causal forces expressed via these optimization formulations explains the properties of the resulting topology and advances our knowledge about the design and evolution of Internet topologies.

2.2 Modeling ISP Topology

In modeling the topology of an ISP, we are necessarily trying to construct a router-level graph that is consistent with the decisions being made by an ISP in the same market environment. Some key questions to be answered for this model include *What are the economic and technical factors facing individual ISPs?* and *What are plausible objectives, constraints, parameters for an optimization formulation?* This formulation can take one of several forms. In a *cost-based* formulation, the basic optimization problem is to build a network that minimizes cost subject to satisfying traffic demand. Alternatively, a *profit-based* formulation seeks to build a network that satisfies demand only up the point of profitability—that is, economically speaking where marginal revenue meets marginal cost.

No matter the formulation, one of the key inputs for this approach is a model for traffic demand. A natural approach to traffic demand is based on *population* centers dispersed over a *geographic* region. In this manner, one could derive the topology of a single "national ISP" from the demand for traffic between people across the country or across town. Furthermore, the tremendous size of a national ISP makes it often convenient to decompose the network into separate problems whenever possible. Most often, this decomposition comes in the form of network hierarchy. It commonly takes the form of backbone networks (wide area networks or WANs), distribution networks (metro area networks or MANs), and customers (local area networks or LANs). Using this approach, the size, location and connectivity of the ISP will depend largely on the number and location of its customers, and it is possible to generate a variety of local, regional, national, or international ISPs in this manner.

2.3 Modeling Internet Topology

Given the ability to effectively model the router-level topology of an ISP (including the placement of peering nodes or points of presence), issues about *peering* become limited to interconnecting the router-level graphs. The relevant questions in this context are *What are the economic and technical factors facing peering relationships between ISPs?* and as before *What are plausible objectives, constraints, parameters for an optimization formulation?* Here, it will be important to leverage previous work on the general economics underlying Internet peering relationships [4], optimal location of peering points between peers [3], and the gaming issues of interdomain traffic management [22]. Furthermore, we believe there may be opportunities to consider peering relationships from the perspective of competitive games.

2.4 Caveats

Using an optimization approach we will generate a solution that is a function of the problem formulation (objective and constraints), the problem data (parameter values), and in cases where the problem cannot be solved exactly *the approximation technique itself*. There are many possible reasons why this approach could fail. For example, we may not be successful in capturing the dominant economic and technical forces driving topological growth. It is also possible that real decisions are neither consistent nor rational and thus do not correspond to any abstract mathematical formulation, although the effort will even then yield beneficial insights into what *ought* to be done. In particular, we expect this approach to shed light on the question of how important the careful incorporation of Level-2 technologies and economics is. Note that current router-level measurements are all IP-based and say little about the underlying link-layer technologies.

3. WILL IT WORK?

Despite the early and somewhat speculative nature of this work, it is supported by both theoretical and empirical evidence and would significantly enrich the current attempts of developing of a unified and integrated theory of the Internet.

3.1 Theoretical Support

A major theme in the physics literature for more than a decade has been the ubiquity of power law distributions in natural and artificial complex systems [5]. Engineered systems such as the Internet have recently been added to that list (e.g., see references in [32]). However, even though the orthodox physics view tends to associate power laws unambiguously with critical phase transitions [5], it is easy to refute this apparent connection—at least in the specific case of the Internet [32]. A radically different alternative view has recently been proposed by Carlson and Doyle [11, 12], relies on the concept of HOT (for *Highly Optimized Tolerance*), and has already been proven to be far more powerful and predictive than the orthodox theory [11, 12, 32].

Highly Optimized Tolerance (HOT): By emphasizing the importance of design, structure, and optimization, the HOT concept provides a framework in which the commonly-observed highly variable event sizes (often referred to as power law behavior) in systems optimized by engineering design are the results of tradeoffs between yield, cost of resources, and tolerance to risk.[3] *Tolerance* emphasizes that robustness (i.e., the maintenance of some desired system characteristics despite uncertainties in the behavior of its component parts or its environment) in complex systems is a constrained and limited quantity that must be diligently managed; *Highly Optimized* alludes to the fact that this goal is achieved by highly structured, rare, non-generic configurations which—for highly engineered systems—are the result of deliberate design. In turn, the characteristics of HOT systems are high performance, highly structured internal complexity, apparently simple and robust external behavior, with the risk of hopefully rare but potentially catastrophic cascading failures initiated by possibly quite small perturbations [12]. The challenge alluded to in Section 4 below then consists of applying HOT in the specific context of network topology by relying on the vast body of existing literature on network optimization.

Heuristically Optimized Tradeoffs: The first explicit attempt to cast topology design, modeling, and generation

[3]HOT thus suggests that these tradeoffs lead to highly optimized designs that perforce allow for a wide range of event sizes, in particular for occasional extreme sizes as a result of cascading failures.

as a HOT problem was by Fabrikant et al. [16] who suggest *heuristically optimized tradeoff* as an alternative (and we think attractive) acronym for HOT. They proposed a toy model of incremental access network design that optimizes a tradeoff between connectivity distance and node centrality. They showed that by changing the relative importance of these two factors to the overall objective function, the resulting topology can exhibit a range of hierarchical structures, from simple star-networks to trees. Furthermore, by tuning the relative importance of the two factors, the authors proved that the resulting node degree distributions can be either exponential or of the power-law type.

3.2 Measurement-Based Support

The work by Faloutsos et al. [17] was the first to report observing power laws for the node degree distributions of measured AS graphs of the Internet. This unexpected finding has stirred significant interest and has led to a drastic increase in the number of studies related to Internet topology modeling and generation (e.g., see [14] and references therein). As for router-level maps, attempts to infer Internet-wide maps have been reported in [25, 9, 15]. For our purpose, the more recent studies described in [19], and especially in [28], are of particular relevance as they focus on recovering the router-level maps of individual ISPs.

While some of these and related studies claim that both the inferred router- and AS-level graphs have similar structural properties, a more careful inspection of the underlying data and their analysis (see, in particular, [14] for the case of AS graphs and [28] for the case of ISP topologies) suggests that there may be indeed different mechanisms at work for generating them. While very different optimization problems can lead to very similar topologies, we believe that the optimization formulations —their objectives, constraints, and parameters—for generating the router-level graph and AS graph are very different. At a minimum, the identification of these formulations will be useful. A second possibility is that the router- and AS-level graphs are more different than similar, but that the standard metrics used to assess network similarity are not the right ones (e.g., see [30, 31]).[4]

3.3 A Piece of a Bigger Puzzle

The Internet serves as ideal starting point for a scientific exploration of the broader issues of robustness in complex systems, partularly those throughout engineering and biology. In most of these systems, complexity is driven by the need for robustness to uncertainty in system components or the operating environment far more than by the need for increased functionality. At the same time, most of this complexity tends to be hidden, deliberately creating the illusion of superficially simple systems and inviting the subsequent development of specious theories. However, motivated largely by the study of HOT systems, recent research efforts have provided for the first time a nascent but promising foundation for a rigorous, coherent, verifiable, and reasonably complete mathematical theory underpinning Internet technology (e.g., see [13] and references therein). This new theory emphasizes the importance of protocols that organize highly structured and complex modular hierarchies to achieve system robustenss, but also tend to create fragilities to rare, neglected, or unknown perturbations. It addresses directly the performance and robustness of both the "horizontal" decentralized and asynchronous nature of control in TCP/IP as well as the "vertical" separation into the layers of the TCP/IP protocol stack from the application layer down to the link layer. At present, the theory is mainly concerned with the transport layer, where the new findings generalize notions of source and channel coding from information theory as well as decentralized versions of robust control. The resulting new theoretical insights about the Internet also combine with our understanding of its origins and evolution to provide a rich source of ideas about complex systems in general.

The work proposed in this paper aims at extending this nascent mathematical theory of the Internet to layers below the transport layer by exploring the decentralized mechanisms and forces responsible for realizing a physical Internet infrastructure, which in turn provides a major ingredient for investigating both the "horizontal" (spatially distributed) and "vertical" aspects of IP routing. The ability to make detailed measurements at the relevant layers and an in-depth knowledge of how the individual parts work and are interconnected facilitate the validation of the proposed approach, make it possible to unambiguously diagnose and "reverse engineer" any claims or findings, and allow a clean separation between sound and specious theories.

4. NETWORK ACCESS DESIGN

There are a multitude of network design problems facing the modern ISP. While the length of this position paper prohibits a comprehensive review of these interesting and important problems, we have chosen to focus our initial attention on the problem of designing a distribution network that provides local access for its customers.[5] Typically, this design problem occurs at the level of the metropolitan area, and it is subject to the service demands of the individual customers and the technical constraints of the equipment in use. The purpose of selecting this problem as a starting point is to illustrate how simple, yet reasonable formulations can provide insight into resulting topology.

The network access design problem was originally studied in the context of planning local telecommunication access (see [6], [18], and references therein).[6] In general, these formulations incorporate the fixed costs of cable installation and the marginal costs of routing, as well as the cost of installing additional equipment, such as concentrators. An emphasis on cost in these formulations leads to solutions that are *tree* (or *forest*) topologies. However, other formulations are possible and can necessarily lead to vastly different topologies.[7]

[4]In fact, we expect that the right optimization formulation will help to identify the appropriate metrics and help to highlight the discrepancies if they exist.

[5]In this context, a customer is anyone who wants direct network access via a dedicated connection and is not using some other infrastructure (e.g. DSL over public telephone lines, cable networks) for this purpose.

[6]To the extent that the methods and intuition resulting from these studies were used in building real telecom networks, the access design problem is of even greater importance to our study, since many of the early Internet topologies piggybacked on these design principles and existing network structures.

[7]For example, adding a path redundancy requirement

4.1 Buy-At-Bulk Access Design

The network access design problem has received renewed attention within the last few years because of the *buy-at-bulk* nature of provisioning fiber-optic cables within ISPs [26, 2]. Buy-at-bulk means that when building its network, the ISP has for its installed links a choice of several different {capacity, cost} combinations, which we refer to as *cable types*. Specifically, each cable type $k \in \{1, 2, \ldots K\}$ has an associated capacity u_k, a fixed overhead (installation) cost σ_k, and a marginal usage cost δ_k. Collectively, the cable types exhibit *economies of scale* such that for $u_1 \leq u_2 \leq \cdots \leq u_K$, one has $\sigma_1 \leq \sigma_2 \leq \cdots \leq \sigma_K$ and $\delta_1 > \delta_2 > \cdots > \delta_K$. In words, larger capacity cables have higher overhead costs, but lower per-bandwidth usage costs.

The problem of interest is to construct a graph that connects some number of spatially distributed customers to a set of central (core) nodes, using a combination of cables that satisfies the traffic needs of the customers and incurs the lowest overall cost to the ISP. Details of the formulation can be found in [26, 2]. Since the decision for the optimal traffic routes and cables choices are interdependent, both problems must be solved simultaneously.

Despite the simple nature of this formulation, problems of this type are hard to solve to optimality. Indeed, constrained network access design problems belong within the family of *minimum cost spanning tree (MCST)* and *Steiner tree* problems, and the buy-at-bulk network design problem is known to be NP-Hard [26]. The best approximation algorithm known is the randomized algorithm by Meyerson et al. [24] who provide a constant factor bound on the quality of the solution independent of problem size.

4.2 Preliminary Results

In a preliminary investigation of the buy-at-bulk network access design problem, we have found that the approximation method in [24] yields tree topologies with exponential node degree distributions. These initial results were obtained using fictitious, yet realistic parameters[8] for cable capacities and costs. We believe that these results are consistent with those in [16], however a thorough search of the parameter space remains to be completed. In either case, we are already finding that the approach embodied by these methods is yielding valuable insights and exposing directions for new research.

5. RESEARCH AGENDA

While this work is still at its early stages, we have identified a number of areas that will require novel contributions within this broad research initiative.

What are the causal relationships between the objectives and constraints of a network design problem and the resulting topology? While there is deep understanding for how to solve combinatorial network optimization problems, the emphasis has traditionally been limited to the accuracy of the solutions and the computational complexity of achieving them. The scientific challenge here is to exploit the HOT perspective of network design to associate concrete optimization formulations with specific characteristic of the resulting network topology. While [16] is a promising first step in this direction, it has little to do with designing real networks.

What is the relative importance of such concrete formulations to real ISP topology design? This includes both backbone networks and distribution networks. We believe there are significant opportunities to learn from the best practices of seasoned network operators [20] as well as the potential for valuable contributions to the large-scale planning of Internet infrastructure. Gaining a basic understanding of this issue will require close interactions with network designers and operators and can benefit from exploiting a variety of different economic data relevant to the ISP and related market sectors.

What metrics and measurements will be required to validate or invalidate the resulting class of explanatory models? Diligent model validation (as for example outlined in [32]) will be an essential aspect of the proposed explanatory topology modeling work, and empirical studies such as those in [15, 28] are a necessary first step. However, by insisting on empirical verification of the causes underlying the advocated HOT-based approach to network topology against available (or yet to be measured) data, we enter an area where significant research efforts will be required for the development of novel and scientifically sound techniques and tools. Again, the result of Fabrikant et al. in [16] and their explicit multi-objective optimization formulation provide a concrete starting point for attempting to validate the forces at work in a proposed HOT-based topology models against feasible measurements.

Is is possible to accurately, yet anonymously characterize an ISP topology? Given that the current market environment for ISPs is only likely to become more competitive, we should consider how to devise technical solutions for the current barriers to information exchange.

What can economic theory tell us about the current and future interaction between competing ISPs? As the Internet becomes integrated as a critical infrastructure for our daily lives, it will be increasingly important that the environment for these companies is stable and that their behavior is predictable.

breaks the tree structure of the optimal solution.

[8]Parameters were chosen to be consistent with the assumptions of the algorithm and the current marketplace.

6. ACKNOWLEDGMENTS

This work was supported by the Institute for Pure and Applied Mathematics at UCLA as part of the 2002 Program on Large-Scale Communication Networks.

7. REFERENCES

[1] W. Aiello, F. Chung, and L. Lu. A random graph model for massive graphs. In *STOC 2000*, 2000.

[2] M. Andrews and L. Zhang. The access network design problem. *39th IEEE FOCS*, 1998.

[3] D. Awduche, J. Agogbua, J. McManus. An approach to optimal peering between autonomous systems in the Internet. *Proc. of Inter. Conf. on Comp. Commun. and Networks*, 1998.

[4] P. Baake and T. Wichmann. On the economics of Internet peering. *Netnomics*, 1(1), 1999.

[5] P. Bak. *How Nature Works: The Science of Self-Organized Criticality* Copernicus, New York, 1996.

[6] A. Balakrishnan, T.L. Magnanti, A. Shulman, and R.T. Wong. Models for planning capacity expansion in local access telecommunication networks. *Ann. of Oper. Res.*, 33, pp. 239–284, 1991.

[7] A.L. Barabasi and R. Albert. Emergence of Scaling in Random Networks. *Science*, 286:509–512, 1999.

[8] T. Bu and D. Towsley. On distinguishing between Internet power law topology generators. *Proc. IEEE INFOCOM'02*, 2002.

[9] H. Burch and B. Cheswick. Mapping the Internet. *IEEE Computer*, 32(4), 1999.

[10] K. Calvert, M. Doar, and E.W. Zegura. Modeling Internet topology. *IEEE Communications Magazine*, 35, pp. 160–163, 1997.

[11] J.M. Carlson and J.C. Doyle. Highly Optimized Tolerance: Robustness and design in complex systems. *Phys. Rev. Lett.*, 84, pp. 2529–2532, 2000.

[12] J.M. Carlson and J.C. Doyle. Complexity and Robustness. *Proc. Nat. Acad. Sci.*, 99, suppl. 1, pp. 2538–2545, 2002.

[13] J. Doyle, J. Carlson, S. Low, F. Paganini, G. Vinnicombe, W. Willinger, J. Hickey, P. Parillo, and L. Vandenberghe. Robustness and the Internet: Theoretical Foundations. http://netlab.caltech.edu/pub/papers/RIPartII.pdf

[14] H. Chang, Q. Chen, R. Govindan, S. Jamin, S. Shenker, and W. Willinger. The origins of power laws in Internet topologies revisited. *Proc. IEEE INFOCOM'02*, 2002.

[15] k. claffy, T. Monk, and D. McRobb. Internet tomography. *Nature*, 1999.

[16] A. Fabrikant, E. Koutsoupias, and C. Papadimitriou. Heuristically Optimized Trade-offs: A new paradigm for power laws in the Internet. *ICALP 2002.* pp. 110–122. 2002.

[17] C. Faloutsos, P. Faloutsos, and M. Faloutsos. On power-law relationships of the Internet topology. *Proc. ACM SIGCOMM'99*, pp. 251–262, 1999.

[18] B. Gavish. Topological design of telecommunication networks—local access design methods. *Ann. of Oper. Res.*, 33, pp. 17–71, 1991.

[19] R. Govindan and H. Tangmunarunkit. Heuristics for Internet map discovery. *Proc. IEEE INFOCOM'00*, 2000.

[20] G. Huston. *ISP Survival Guide: Strategies for Running a Competitive ISP.* John Wiley & Sons, New York, 2000.

[21] C. Jin, Q. Chen, and S. Jamin. Inet: Internet topology generator. Tech. Rep. CSE-TR-433-00, Univ. of Michigan, Ann Arbor, 2000.

[22] R. Johari and J. Tsitsiklis. Routing and peering in a competitive Internet. Presentation at *IPAM Workshop on Large-Scale Comm. Networks: Topology, Routing, Traffic, and Control*, UCLA, 2002.

[23] A. Medina, A. Lakhina, I. Matta, and J. Byers. BRITE: An approach to universal topology generation. *Proc. MASCOTS'01*, 2001.

[24] A. Meyerson, K. Mungala, S. Plotkin. Designing networks incrementally. *41st IEEE FOCS*, 2001.

[25] J. Pansiot and D. Grad. On routes and multicast trees in the Internet. *ACM Comp. Comm. Rev.*, 1997.

[26] F.S. Salman, J. Cheriyan, R. Ravi, and S. Subramanian. Buy at bulk design: approximating the single-sink installation problem. In *Proc. 8th ACM-SIAM Symp. on Discrete Algorithms*, pp. 619–628, 1997. (Revised, 1998.)

[27] H. Simon. On a class of skew distribution functions. *Biometrika*, 42:425–440, 1955.

[28] N. Spring, R. Mahajan, and D. Weatherall. Measuring ISP topologies with rocketfuel. *Proc. ACM SIGCOMM'02* (to appear).

[29] H. Tangmunarunkit, J. Doyle, R. Govindan, S. Jamin, S. Shenker, and W. Willinger. Does AS size determine degree in AS topology? *ACM Comp. Comm. Rev.*, 31, pp. 7–10, 2001.

[30] H. Tangmunarunkit, R. Govindan, S. Jamin, S. Shenker, W. Willinger. Network topology generators: Degree-based vs. structural. *Proc. ACM SIGCOMM'02* (to appear).

[31] D. Vukadinović, P. Huang, and T. Erlebach. A spectral analysis of the Internet topology. ETH TIK-NR. 118, 2001.

[32] W. Willinger, R. Govindan, S. Jamin, V. Paxson, and S. Shenker. Scaling phenomena in the Internet: Critically examining criticality. *Proc. Nat. Acad. Sci.*, 99, suppl. 1, pp. 2573–2580, 2002.

[33] E. Zegura, K. Calvert, and M. Donahoo. A quantitative comparison of graph-based model for Internet topology. *ACM/IEEE Trans. on Networking*, 5(6):770–783, 1997.

Lowering the Barrier to Wireless and Mobile Experimentation

Brian White Jay Lepreau Shashi Guruprasad
School of Computing, University of Utah
{bwhite,lepreau,shash}@cs.utah.edu www.flux.utah.edu www.netbed.org

ABSTRACT

The success of *ns* highlights the importance of an infrastructure that enables efficient experimentation. Similarly, Netbed's automatic configuration and control of emulated and live network environments minimizes the effort spent configuring and running experiments. Learning from the evolution of these systems, in this paper we argue that a live wireless and mobile experimental facility focusing on ease of use and accessibility will not only greatly lower the barrier to research in these areas, but that the primary technical challenges can be overcome.

The flexibility of Netbed's common abstractions for diverse node and link types has enabled its development from strictly an emulation platform to one that integrates simulation and live network experimentation. It can be further extended to incorporate wireless and mobile devices. To reduce the tedium of wireless and mobile experimentation, we propose automatically allocating and mapping a subset of a dense mesh of devices to match a specified network topology. To achieve low-overhead, coarse repeatability for mobile experiments, we outline how to leverage the predictability of passive couriers, such as PDA-equipped students and PC-equipped busses.

1. INTRODUCTION

Instruments can catalyze an entire field. In the hard sciences the right instruments are crucial, and are frequently objects of research in their own right. Closer to home, we've seen the broad impact of the *ns* simulator [1], which is used in a high fraction of network research. The benefits to the networking community of a common experimental environment have been discussed in the simulation context [1]. These include improved protocol validation, a rich infrastructure for developing and testing new protocols, a controlled experimental environment, and easier comparison of experimental results. Many of these benefits are the byproduct of a community-endorsed environment and are not unique to a simulation environment.

We have already seen impressive results with Netbed [23], an infrastructure supporting emulation, simulation, and live network experimentation. However, the nature of mobile and wireless devices both exacerbates existing problems of sound network evaluation and raises new challenges to testbed design. We believe that community-accessible wireless and mobile testbeds will have even greater impact than a wired testbed and that the technical challenges can be overcome.

This paper addresses the following limitations of most wireless and mobile experimental environments:

Lack of validation: Simulation's abstraction of low-level detail may come at the expense of accuracy. The tradeoff between accuracy and efficiency is particularly acute in a wireless network. For example, network simulators [1, 9, 24] typically incorporate idealized radio propagation models which are inadequate to model interesting indoor scenarios. Unfortunately, an experimenter is forced to make the tradeoff between accuracy and efficiency without any systematic means of validating the choice of abstraction [5].

Tedious experimental setup: Where live network experimentation is burdened by an experimenter's need to obtain remote accounts and configure nodes, mobile experimentation is further complicated by node placement and movement. A number of past [18] and proposed [19] mobile network testbeds, using automobiles, require expensive and non-scalable manual control. In fact, a lack of drivers proved a significant limitation [17]. Even static domains present configuration headaches. Kaba and Raichle interface wireless devices to a wired network with induced attenuation to curb the irreproducibility caused by multi-path effects from nearby people and objects [10]. This approach requires physically configuring the attenuators within the wired network.

Lack of realistic mobile scenarios: Simulation environments introduce randomized mobility models [3] that successfully stress network protocols, but make little attempt to capture real-world dynamics. Johansson *et al.* simulated a number of different scenarios including a conference, event coverage, and a disaster area [7]. However, these remain inaccessible to live experimentation, which is relegated to the artificial motion patterns of the above mobile testbeds. Another approach [25] skirts the issue of realistic mobility patterns by requiring that the experimenter specify a scenario describing node movement. This information is used to emulate movement on a wired network topology by affecting the "connectivity" of wired hosts. Interference effects are modeled off-line (e.g., by the Monarch project's ns extensions [9]) and are reflected in the scenario.

Lack of scale and availability: Unlike desktop machines, which abound throughout industry and academia, mobile and wireless devices are more exotic and hence available in lesser quantities and to much smaller communities. This limits the scale of existing testbeds, such as MIT's 30-node wireless "Grid" testbed [16].

Though mobile and wireless communication impose unique constraints on an experimental infrastructure, many of the above issues

First Workshop on Hot Topics in Networks (Hotnets-I), Princeton, NJ, USA, October 28–29, 2002.
This work was largely sponsored by NSF grants ANI-0082493 and ANI-0205702, Cisco Systems, and DARPA grant F30602-99-1-0503.

are not specific to these domains. We believe that many of the benefits Netbed has brought to emulation, simulation, and live network experimentation will be transferable to a mobile and wireless testbed. Section 2 describes how Netbed leverages shared abstractions despite diverse underlying hardware and implementations. It outlines how these techniques, originally designed for an emulation environment, have been successfully employed in the simulation and live network context and how they might similarly impact wireless and mobile experimentation.

The barriers to experimental evaluation in the wireless and mobile domains are much higher than in typical wired networks. This paper explores testbed designs that should dramatically lower these barriers. Section 3 discusses how interesting wireless scenarios can be realized by a dense mesh of wireless devices deployed indoors and outdoors. An automated mapping algorithm will select the subset of nodes that best match an experimenter's specified network topology and relieve the experimenter of strategically and painstakingly placing wireless devices.

Section 4 presents the notion of passive couriers: PDA- or PC-equipped mobile agents, such as students or busses, exhibiting predictable motion patterns. While their motion patterns are not completely reproducible, the dictates of their schedules mean that passive couriers capture realistic mobile scenarios with coarse-grain repeatability (e.g., within a few minutes and a few meters) at predictable times (e.g., every morning at 8 AM) without the manual tedium that is typical of mobile experimentation. Scenarios such as classroom interaction are exactly those that have motivated ad hoc networks. Finally, section 5 concludes.

2. NETBED

Netbed is a direct outgrowth of Emulab, a network experimentation platform that focused on efficient setup and control over *emulated* topologies. Key goals were to make the facility both universally available to any external researcher and extremely easy to use, without administrative, technical, or other obstacles. It pushes *automation* of testbed configuration and control to a qualitatively new level, allowing both interactive and programmatic exploration of a large space of experimental conditions.

Netbed extends the original platform by introducing simulated and distributed nodes and allowing their simultaneous use alongside emulated nodes in mixed, virtual topologies. Netbed's design generalizes resources and mechanisms into common abstractions applicable across the diverse realizations of emulation, simulation, and live network experimentation.

Netbed configures a set of distributed, emulated, or simulated nodes to realize a virtual topology specified either graphically or via an *ns* script. An experiment is defined by its configuration and any run-time dynamics (e.g., traffic generation) specified via the general-purpose *ns*/Tcl interface. Netbed's automated configuration includes managing experimenter accounts, setting emulated link characteristics, mapping the virtual nodes and links to physical resources, downloading clean disk images on emulated nodes, setting up network interfaces and IP addresses, and optionally configuring a primitive virtual machine to "jail" [11] each user of a shared machine. Once an experiment is configured, an experimenter may interactively log into emulated or distributed nodes.

A Netbed experiment may last from a few minutes to many weeks, giving researchers time to make multiple runs, change their software and parameters, or do long-term data gathering. Netbed's Web interface allows experimenters to create, pause, and terminate experiments remotely. All aspects of the experiment can be controlled via the web interface. When used in conjunction with Netbed's batch experiment system, a researcher is able to submit an *ns* file over the web and, when enough hardware resources become available to run the experiment, the user is notified that the experiment has started.

2.1 Supporting Heterogeneous Resources

Links in the user-specified, virtual topology may be emulated by traffic-shaping Dummynet nodes interposing experimental nodes, may be simulated via *ns*, or may be realized by wide-area links. *ns*'s emulation facility, *nse*, acts as transducer for packets crossing the simulation/live network boundary. An important feature of Netbed is the consistent interface it provides to control nodes and links regardless of their realization. For example, the same command uses a distributed event system to start and stop traffic generators on any type of node, be it distributed, emulated, or simulated.

The integration of heterogeneous resources is largely enabled by a database. The database serves as a level of indirection between front-end, general-purpose tools and interfaces and back-end, domain-specific implementations. It presents a consistent abstraction of heterogeneous resources to higher layers of Netbed and to experimenters. For example, the database representations of distributed and emulated nodes differ only in a type tag. Thus, in many cases, experimenters can interact with them using the same commands, tools, and naming conventions regardless of their implementation. As an example, nodes of any type can host traffic generators, despite the fact that traffic may flow over links simulated by *ns*, emulated by delay nodes, or provided by a distributed testbed.

2.2 Improving Wireless and Mobile Experimentation

Just as Netbed's notion of nodes and links has evolved to encompass distributed and simulated nodes and links, we believe the infrastructure is sufficiently flexible to incorporate wireless and mobile virtual node and link types. This will bring important practical benefits to experimentation in this domain, including: automated and efficient realization of virtual topologies, efficient use of resources through time- and space-sharing, increased fault-tolerance through resource virtualization, an ability to leverage existing tools, and easier validation across experimental techniques.

Ease of use and automation are not a mere convenience; they enable qualitatively new approaches. Our user experiments show that after learning and rehearsing the task of manually configuring a 6-node "dumbbell" network, a student with significant Linux system administration experience took 3.25 hours to accomplish what Netbed accomplished in less than 3 minutes. This factor of 70 improvement and the subsequent programmatic control over links and nodes encourage "what if" experiments that were previously too time- and labor-intensive to even consider. Experiment setup cost is even more acute in wireless and mobile domains, which require careful measuring of interference effects and "walk-through" experiments. Thus, the savings afforded by automated mapping of a virtual topology to physical devices removes a significant experimentation barrier.

Efficient use of scarce and expensive infrastructure is also important and a sophisticated testbed system can markedly improve utilization. For example, analysis of 12 months of the wired Netbed's historical logs gave quantitative estimates of the value of time-sharing (i.e., "swapping out" idle experiments) and space-sharing (i.e., isolating multiple active experiments). Although the behavior of both users and facility management would change without such features, the estimate is still revealing. Without Netbed's ability to time-share its 168 nodes, a testbed of 1064 nodes would have been required to provide equivalent service. Similarly, without

space-sharing, 19.1 years would be required. These are order-of-magnitude improvements. The importance of resource efficiency is heightened for wireless and mobile devices since they are less prevalent than commodity PCs.

Netbed virtualizes node names and IP addresses such that nodes and links form equivalence classes. For example, when an experiment is "swapped in" (i.e., reconstituted on physical resources from database state), it need not execute on the same set of physical nodes. Any nodes exhibiting the same properties and interconnection characteristics are suitable candidates. While virtual nodes may be explicitly bound to specific physical hosts, the flexibility to allocate from an equivalence class adds a measure of fault tolerance. If a node or link fails, an experimenter need not wait until the node or link partition is healed, but may instead re-map the experiment to an equivalent set of machines. This approach is valuable wherever node or link failures are anticipated: large-scale clusters, wide-area networks, or unstable wireless environments.

Incorporating wireless and mobile devices under the Netbed umbrella brings a mature set of tools and features to these domains. For example, once domain-specific resource mapping is provided, experimenters will use the existing *ns* or graphical interfaces to allocate wireless and mobile nodes. This ensures that any topology generator that produces *ns* syntax may be used to configure an experiment, while *ns* virtualization tools may be used to view it. The infrastructure will extend consistent control and specification of traffic generators across all experimental methodologies. The familiar user account management, including hierarchical authorization, key distribution, and account establishment will translate directly to these new domains. Applications running on wireless nodes will be controllable via the current event system and standard commands.

The common *ns* interface makes it easier to compare experimental environments, thereby facilitating and encouraging validation. Extending this capability to wireless and mobile nodes will provide an automatic way to compare simulated radio propagation models with real devices. The same *ns* script could then be used to instantiate either an *ns* simulation or a live experiment, leading to an approach in which the simulated model is iteratively refined according to empirical results.

We believe a new hybrid style of experimentation, incorporating resources from multiple experimental environments, can simultaneously leverage the particular strengths of each. More specifically, an experimenter can leverage the greater scalability of simulation without surrendering confidence in the accuracy of the results. A radio propagation model is sufficiently accurate *for a given application*, if the simulation it drives is indistinguishable from the interaction between live, wireless nodes. To this end, we suggest replacing a small "island" of simulated nodes with physical nodes. This subset of nodes will use live, wireless communication amongst themselves and simultaneously transmit packets within the simulated world. This allows an experimenter to apply an emulation "microscope" to achieve highly accurate introspection of an "island" of nodes. Moving or removing this "island" from the experiment will yield similar results if the analytic model of the simulator and the real instantiation of those models are equivalent. Further, if sets of nodes exhibit symmetrical interactions, the behavior of a cluster of interacting simulated nodes may be compared to a corresponding set of live wireless nodes.

Such close interaction between live traffic and simulation requires careful synchronization between simulated and real time. We intend to leverage the work of Ke *et al.* [12] to reduce timing discrepancies between the two worlds. The relative expense of wireless simulation exacerbates such concerns. A parallelizable

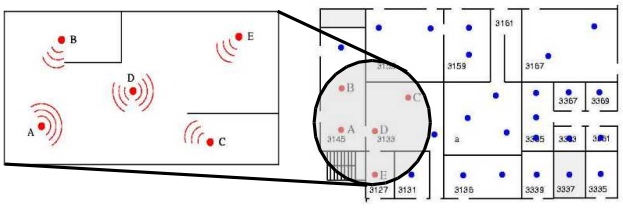


Figure 1: Wireless Virtual to Physical Mapping

simulation architecture, such as GloMoSim, would be more efficient in a multiprocessor environment, and is worth exploring. We have already done some work in this area, including cutting CPU usage in half by making some modest changes to *nse*'s scheduler.

3. WIRELESS TESTBED

Broadcast and interference effects are difficult or impossible to emulate on a wired testbed, although centralized simulation of all links [12] can provide some degree of realism. However, the lack of good propagation models, particularly for indoor environments, severely limits the accuracy of this approach. Trace modulation [20] finesses concerns over broadcast and interference issues as it captures their effect within a trace. However, the traces are bound to the sampled environment; if the scenario changes, new traces must be collected.

Our approach is to overcome these difficulties with a large pool of physically distributed test nodes that have real RF devices. When the number of nodes under test is significantly fewer than the total number of available nodes, it should be possible to provide a reasonable approximation of the intended virtual environment. Given the costs of the devices involved, the benefit to researchers, and the multi-institutional stake of the facility, we believe that sufficient device density will become available. As an example of the scale under consideration, we are considering 500 devices within a four-story building.

The wireless Netbed will include a dense deployment of wireless devices throughout one or more campus buildings. [1] Devices will typically be attached to control machines (e.g., low-end PCs or similar; UCLA is using PC-104 "Octopus" machines each of which controls 8 RF devices), enabling independent, reliable control by Netbed tools. Many devices will have permanent—and to the degree economical, controllable—sources of power and power monitoring hardware. This avoids the maintenance problem of battery replacement while facilitating studies of power-related issues.

For an indoor wireless mesh, 900 Mhz devices have the appropriate range and popularity. We will select devices with a programmable MAC layer, such as "motes" [6] from Crossbow/Intel/-Berkeley, giving experimenters flexibility in that important dimension. The pervasiveness of 802.11b makes it an obvious choice to populate the indoor mesh. The higher data rates and larger number of frequency channels also make 802.11a an attractive option. To allow closer packing, we will reduce the transmit power of the devices [10]. Bluetooth devices could be deployed very densely, but their long-term importance and popularity are uncertain. Sensors, including those for temperature, light, and probably motion, will be included on nodes due to their low marginal cost and importance to real-time applications.

[1] In a wild possibility for researchers or others, Aerie Networks acquired Metricom's Ricochet 900 Mhz system, but has "abandoned in place" huge numbers of operational transceivers and access points in many U.S. cities. [4].

The success of an automated, wireless testbed hinges on two primary challenges: providing a high fidelity mapping of a virtual topology to wireless nodes and ensuring that interference does not pollute experimental results. These considerations are examined in the following subsections.

3.1 Mapping Wireless Resources

Our experience mapping complex virtual requirements to physical resources in Netbed showed that approaches based on combinatorial optimization can be practical with sufficient attention to abstraction, data representation and, sometimes, local search. Currently, a randomized heuristic algorithm, simulated annealing [8], underlies allocation of Netbed's local resources (nodes, links, and switches), efficiently targeting the NP-hard problem of mapping virtual resources to physical ones. We improved its computational behavior by grouping physical resources into equivalence classes based on their physical attributes; it finds solutions in a few seconds.

Whereas the specification of a wired topology is fairly straightforward, a faithful mapping of an experimenter's intent to wireless resources is highly dependent on the level of detail provided by the configuration interface. Such an interface must avoid a circular dependency: if Netbed were to rely on an existing simulation model to map a virtual topology to physical resources, the system's reliance on models incorporates the potential inaccuracies that live experimentation seeks to avoid! It may be possible to use offline analysis that would be intolerably slow in a simulation. Other possibilities, outlined below, avoid analytic models in favor of more intuitive and efficient interfaces.

Configurable wireless experimentation will allow manual or automatic selection of a subset of the wireless nodes, chosen to match certain characteristics, as represented in Figure 1. We plan three different user interfaces. First, and simplest to develop, we will provide an annotated 3-D map of the building, with nodes colored green, yellow, or red, indicating whether they are available, assigned to an idle experiment ("swappable"), or busy. Experimenters simply select the nodes they prefer, inferring link characteristics from the map and its annotations.

Next, we will develop a more abstract interface that allows a user to specify a scenario graphically, based on spatial arrangement, as in Figure 1. The mapping code will select the set of physical nodes that best match. Our algorithmic approach is uncertain at this stage, but seems to have an intuitive mapping to subgraph isomorphism. Graph edges correspond to some abstracted characterization of the configuration that affects attenuation, such as distance, number of obstructing walls, their orientation—but not to any concrete metric such as attenuation itself. However, this still implies estimating a propagation model, with all its assumptions. A better approach may be simply to match the spatial layout as closely as possible.

Experimental results from these first two approaches are clearly dependent on the RF idiosyncrasies of the particular building and environment. However, this downside is outweighed by the building's becoming a "reference platform" across multiple experiments and between experimenters—which has heretofore not been available.

Lastly, the experimenter could supply a desired configuration of node radio connectivities or higher-level properties such as bit error rate and the system would choose the set of real nodes that most closely match that configuration. This will require prior measurements of the NxN inter-node signal strength or link characteristics, ideally while selected combinations of other traffic is flowing. Solving this problem in discrete optimization should be feasible. This approach offers better hope of precision, but has the drawback that experimenters typically don't know the connectivity characteristics of their target environment.

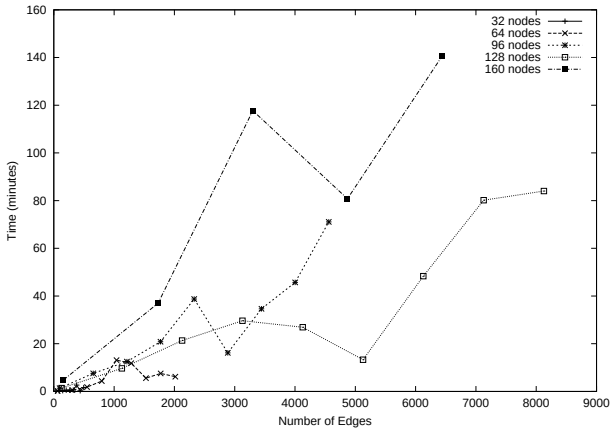


Figure 2: Time taken by genetic algorithm to map a variety of wide-area topologies

This last approach is analogous to the way we currently map wide-area resources. In our system, when an experimenter requests actual wide-area nodes and specifies desired inter-node path characteristics, we use a genetic algorithm to find the best-matching set of physical nodes and their corresponding Internet paths. Input to this algorithm consists of NxN latency and loss data, updated frequently. The wireless resource allocation problem is similar to this wide-area IP-path-matching problem, since the broadcast medium fully connects all nodes (in principle). Our experience with the wide-area algorithm suggests that the wireless problem will be challenging, but—with appropriate optimizations—tractable, well into the hundreds of nodes.

Our results on a simulated physical topology of 256 nodes are shown in Figure 2. All experiments with 32 nodes, as well as all sparse topologies, mapped in a few minutes. For larger and more dense topologies, up to 256 nodes and approximately 40 edges/node, mapping time ranged from 10 minutes to 2 hours.

We expect to improve that by an order of magnitude using the following techniques: less stringent and more clever termination conditions; standard optimization techniques, in particular memoizing; and parallelizing the algorithm, which is practical in either a shared memory multiprocessor or on a cluster [21]. In the "Island Model," mutations and crossovers can be done in parallel, sharing the best solutions periodically; we estimate synchronizing every few seconds to exchange 1–2KB of data. Other algorithmic approaches may also be relevant. For example, constraint programming [22], a method guaranteed to find a feasible solution if one exists or report otherwise, should produce better results for problems tightly constrained by heavy utilization of physical resources.

Finally, we expect major additional improvement to come from "binning" the nodes and links into groups with similar characteristics, which will dramatically reduce the search space. The result should be an algorithm that can map hundreds of nodes and links in a few minutes.

3.2 Interference

To retain Netbed's efficient use of space and resources, wireless experiments should be isolated from one another and from the environment to the greatest extent possible. Interference from unrelated "production" traffic or from devices such as microwave ovens, Bluetooth and cordless phones in the 2.4 GHz band may lead to

anomalous experimental behavior. Unfortunately, there is an inherent conflict in the unlicensed spectrum: popular technologies are the most interesting to study, but also the most likely to be incidentally present in the environment. There are three recourses: using buildings or areas that are not used for such traffic, negotiating for a subset of channels to be unused for production, or studying devices that are still on the upswing in popularity. All of these are reasonable approaches.

One of these techniques, the use of multiple channels, should also alleviate potential interference caused when multiple experiments occupy intersecting transmission ranges or collision domains. Wireless devices with overlapping collision domains can retain separate conversations by occupying *non-overlapping* frequencies. To enforce isolation, Netbed would take this additional constraint into account during the mapping phase.

In the wired emulated arena, Netbed leverages VLAN (Virtual LAN) switch technology to enforce isolation of experiments between separate network partitions. While each experiment may be readily mapped to a distinct VLAN, the number of non-overlapping frequencies within 802.11b's available spectrum is more limited. Given its channel width of 22 MHz, only 3 of the 11 channels available (in the United States) are non-overlapping [14]. Since 802.11a supports 8 independent, non-overlapping channels [13], it may be a more suitable technology to support a dense mesh.

An alternate approach to achieving greater density reduces transmission power; this effectively decreases both the range and attendant collision domain. In an analogy to graph coloring, because there are fewer overlapping domains, fewer "colors" or non-overlapping channels are required for isolation. This technique avoids a large physical footprint by simultaneously scaling down both the transmission power and inter-node distance. There is a caveat; reduced transmission power may not be indicative of real scenarios and may suffer some loss of realism.

An active channel may be assigned to a new experiment if the experiments in question do not share overlapping collision domains. To prevent an overly aggressive channel reuse strategy that would lead to co-channel interference, the wireless devices will be placed according to a network plan that aims to reduce interference while retaining high density. After placement, the ranges of each device will be carefully measured to be used later by the mapping algorithm to determine collision domains. Online wireless channel monitoring will be employed in all the nodes and made available to experimenters so that results from experimental runs with unacceptably high interference levels could be discarded or dealt with in a suitable manner by the experimenter.

4. MOBILITY

Configurable mobile environments are critical in the evaluation of many wireless systems, since coping with mobility is often the hardest part in the design of, for example, ad-hoc routing algorithms. To extend Netbed to provide actual mobile nodes, we will deploy a large, dense set of wireless devices via passive couriers that move predictably in time and space. We will use two types of couriers: students moving from class to class with radio-equipped PDAs, and city and campus busses with wireless PCs. Both exhibit predictable movement patterns. We expect that it is the general dynamics of a scenario rather than the tracking of individual nodes that make it interesting. Therefore, experimenters will specify a desired scenario (e.g., students wandering the halls, students eating in the cafeteria, etc.) and the density of its constituent nodes.

For those experiments where the precise relative motions of each node is significant, the Netbed software could manage automated couriers in the form of GPS-equipped radio-controlled cars that navigate a large open space. Differential GPS with RTK can localize to 1 cm when a correction signal is provided every second [17].

Couriers will present a previously unavailable source of live, realistic scenarios. They will offer the following advances over current approaches:

- Couriers remove the dependence on inaccurate simulation models.
- Couriers, by definition, will provide scenarios that are representative of real world behavior. An understanding of their behavior can guide ad hoc protocols in the same manner that file access characterizations [2] influence file system design. By tracking the location of the couriers we will be able to develop models or traces of their behavior to be used in simulation.
- Since they are governed by bus or class schedules, couriers will provide regular behavior, though not complete repeatability. Movement "noise" that deviates from the intended schedule will enable an accurate study of predictable mobility because of its consistency with real world behavior. Such coarse predictive capabilities were suggested as an extension to Grid's location service (GLS) [16].

Experimenters will have the option of selecting from a subset of students sitting in classrooms, wandering the halls, or eating lunch, which constitute a broad range of mobility. Experimental code will execute on the students' PDAs. Repeatability will be approximated by running experiments to coincide with defining periods in the schedule, for example when students disembark from the bus in the morning or every hour when they leave their classrooms. Such scenarios capture proposed ad hoc network activities, such as classroom interaction. Further, their breadth of movement patterns will provide an opportunity to study a protocol's steady-state behavior and to examine its agility in the face of transient topology changes.

The passive courier approach does not afford the perfect repeatability of simulation. Nevertheless, experience indicates that realism and ease of setup are invaluable, even without complete repeatability. In addition, experiments can be run many times, since the process is fully automated and can be run in batch mode. Simple statistical techniques (read "the average") can help compensate for lack of perfect repeatability.

City busses follow a unique mobility pattern that suggests alternate classes of protocols. Unlike classroom or random motion settings, the number of neighbors reachable by a particular node in a bus transit scenario will change dramatically over time. For example, there are typically 20 busses within a half mile radius of downtown Salt Lake City; more densely populated cities presumably have greater bus density. A cursory glance at bus schedules similarly indicates the potential for interesting multi-hop topologies, assuming antennas are deployed to reinforce a radio's nominal range. After leaving downtown, bus density decreases greatly.

This pattern is conducive to a cooperative long-term store and forward approach based on prediction. The resulting network is a hybrid since it exhibits the dynamism of an ad hoc network and the known (future) topology of a wired network. A bus node might selectively transmit data to a passing bus based on its destination. In such a scenario, sparingly transmitting data prevents pollution of the radio spectrum. For example, busses might engage in highway congestion monitoring [19], wherein an outbound bus only forwards downtown traffic updates to those busses likely to experience congestion.

A network composed of student couriers riding on school bus couriers provides *levels* of coordination, or an ad hoc hierarchy.

While riding on the bus, students are largely stationary and their motion is uninteresting. During this time, a bus node might act as a router for student traffic, perhaps performing aggregation. When students arrive at school and leave the bus, they once again assume the role of first-class peers. Such a scenario has military parallels in which mobile divisions may need to coordinate with upper echelons [15].

5. CONCLUSION

Shared wireless and mobile testbeds would dramatically lower the costs and barriers to emulation and live network experimentation. They will bring the ease of use, configurability, *ns*-compatibility, and transparency of the existing Netbed infrastructure to the wireless and mobile domains. By facilitating and automating large-scale mobile and wireless experimentation, we expect such testbeds to gain widespread adoption, leading to sets of community-accessible reference platforms. Such platforms promote comparable results, encourage validation, and will advance the state of the art in experimental design, setup, and execution.

6. REFERENCES

[1] S. Bajaj et al. Improving Simulation for Network Research. Technical Report 99-702b, USC, March 1999.

[2] M. G. Baker, J. H. Hartman, M. D. Kupfer, K. W. . Shirriff, and J. K. Ousterhout. Measurements of a distributed file system. In *Proc. of 13th ACM SOSP*, pages 198–212, 1991.

[3] J. Broch, D. A. Maltz, D. B. Johnson, Y.-C. Hu, and J. Jetcheva. A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols. In *Proc. of the Fourth Annual ACM/IEEE MOBICOM*, October 1998.

[4] S. M. Cherry. What Went Wrong at Ricochet. *IEEE Spectrum*, 29(3):60–61, Mar. 2002.

[5] J. Heidemann, N. Bulusu, J. Elson, C. Intanagonwiwat, K. chan Lan, Y. Xu, W. Ye, D. Estrin, and R. Govindan. Effects of Detail in Wireless Network Simulation. In *Proc. of the SCS Multiconference on Distributed Simulation*, pages 3–11. USC/ISI, January 2001.

[6] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister. System Architecture Directions for Networked Sensors. In *Proceedings of the 9th Intl. Conf. on Architectural Support for Programming Languages and Operating Systems*, pages 93–104, Nov. 2000.

[7] P. Johansson, T. Larsson, N. Hedman, B. Mielczarek, and M. Degermark. Scenario-based Performance Analysis of Routing Protocols for Mobile Ad-hoc Networks. In *Proc. of the Fifth Annual ACM/IEEE MOBICOM*, pages 195–206, August 1999.

[8] D. Johnson, C. Aragon, L. McGeoch, and C. Schevon. Optimization by Simulated Annealing: An Experimental Evaluation; Part I, Graph Partitioning. *Operations Research*, 37(6):865–893, 1989.

[9] D. B. Johnson et al. The CMU Monarch Project's Wireless and Mobility Extensions to ns. Technical report, Carnegie Mellon University, August 1999.

[10] J. Kaba and D. Raichle. Testbed on a Desktop: Strategies and Techniques to Support Multi-hop MANET Routing Protocol Development. In *Proc. of the 2001 ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc'01)*, Oct. 2001.

[11] P.-H. Kamp and R. N. M. Watson. Jails: Confining the Omnipotent Root. In *Proc. 2nd Intl. SANE Conference*, May 2000.

[12] Q. Ke, D. A. Maltz, and D. B. Johnson. Emulation of Multi-Hop Wireless Ad Hoc Networks. In *Proc. of the Seventh International Workshop on Mobile Multimedia Communications*, October 2000.

[13] LAN/MAN Standards Committee of the IEEE Computer Society. IEEE Std 802.11a-1999. Technical report, IEEE, 1999.

[14] LAN/MAN Standards Committee of the IEEE Computer Society. IEEE Std 802.11b-1999. Technical report, IEEE, September 1999.

[15] B. M. Leiner, R. J. Ruth, and A. R. Sastry. Goals and Challenges of the DARPA GloMo Program. *IEEE Personal Communications*, December 1996.

[16] J. Li, J. Jannotti, D. S. J. De Couto, D. R. Karger, and R. Morris. A Scalable Location Service for Geographic Ad Hoc Routing. In *Proc. of the 6th Annual Intl. Conf. on Mobile Computing and Networking*, Aug. 2000.

[17] D. A. Maltz, J. Broch, and D. B. Johnson. Experiences Designing and Building a Multi-Hop Wireless Ad Hoc Network Testbed. Technical Report CMU-CS-99-116, Carnegie Mellon University, March 1999.

[18] D. A. Maltz, J. Broch, and D. B. Johnson. Lessons from a Full-Scale Multi-Hop Wireless Ad Hoc Network Testbed. *IEEE Personal Communications*, Feb. 2001.

[19] R. Morris, J. Jannotti, F. Kaashoek, J. Li, and D. Decouto. CarNet: A Scalable Ad Hoc Wireless Network System. In *Proc. of the Ninth ACM SIGOPS European Workshop*, Kolding, Denmark, Sept. 2000.

[20] B. Noble, M. Satyanarayanan, G. T. Nguyen, and R. H. Katz. Trace-Based Mobile Network Emulation. In *Proc. of SIGCOMM '97*, September 1997.

[21] R. Tanese. The Distributed Genetic Algorithm. In *Third International Conference on Genetic Algorithms*, pages 434–439. Morgan Kaufmann, 1989.

[22] P. Van Hentenryck. *Constraint Satisfaction in Logic Programming*. Logic Programming Series, The MIT Press, Cambridge, MA, 1989.

[23] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar. An Integrated Experimental Environment for Distributed Systems and Networks. In *Proc. of the 5th Symposium on Operating Systems Design and Implementation*, Boston, MA, December 2002.

[24] X. Zeng, R. Bagrodia, and M. Gerla. GloMoSim: A Library for Parallel Simulation of Large-scale Wireless Networks. In *Proc. of the 12th Workshop on Parallel and Distributed Simulation*, pages 154–161, May 1998.

[25] Y. Zhang and W. Li. An Integrated Environment for Testing Mobile Ad-Hoc Networks. In *Proc. of the 2002 ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc'02)*, June 2002.

XORP: An Open Platform for Network Research

Mark Handley Orion Hodson Eddie Kohler

ICSI Center for Internet Research, Berkeley, California

{mjh, hodson, kohler}@icir.org

ABSTRACT

Network researchers face a significant problem when deploying software in routers, either for experimentation or for pilot deployment. Router platforms are generally not open systems, in either the open-source or the open-API sense. In this paper we discuss the problems this poses, and present an eXtensible Open Router Platform (XORP) that we are developing to address these issues. Key goals are extensibility, performance and robustness. We show that different parts of a router need to prioritize these differently, and examine techniques by which we can satisfy these often conflicting goals. We aim for XORP to be both a research tool and a stable deployment platform, thus easing the transition of new ideas from the lab to the real world.

1 VALIDATING INTERNET RESEARCH

A yawning gap exists between research and practice concerning Internet routing and forwarding disciplines. The savvy researcher has the tools of theory and simulation at hand, but validating results in the real world is hard. Why should this be so?

For network applications research, we have access to languages, APIs, and systems that make development and deployment easy. For end-to-end protocol research, we have access to open source operating systems, such as Linux and FreeBSD. End-to-end protocols can be simulated and implemented in these systems. And since these operating systems are used in both research and production environments, migration from the research to the production environment is feasible. TCP SACK provides an excellent example [8].

Unfortunately the same cannot be said of router software. Router vendors do not provide API's that allow third party applications to run on their hardware. Thus, even conducting a pilot study in a production network requires the router vendor to implement the protocol. Unless the router vendor perceives a reward in return for the effort, they are unlikely to invest resources in the protocol implementation. Similarly, customers are unlikely to request a feature unless they have faith in existing research results or can experiment in their own environment. A catch-22 situation exists of not being able to prototype and deploy new experimental protocols in any kind of realistic environment. Even when vendors can be convinced to implement, it is not uncommon for initial implementations of a protocol to be found wanting, and the path to improving the protocols is often difficult and slow. Finally, network operators are almost always reluctant to deploy experimental services in production networks for fear of destabilizing their existing (hopefully money-making) services.

Thus, we believe the difficulty in validating Internet research is largely attributable to the absence of open Internet routers for researchers to experiment with and deploy new work on. Routing toolkits exist, but typically they implement a subset of IP routing functionality and are rarely used in production environments—routing and forwarding research requires access to real production traffic and routing information to be validated. Similarly, open-source-based testbed networks such as CAIRN [1] provide valuable tools for the researcher, but they rarely provide a realistic test environment and are usually limited to a small number of sites due to cost. A recent spate of research in open, extensible forwarding paths is moving in the right direction [6, 11], but a truly extensible, production-quality router would need routing daemons, forwarding information bases, management interfaces, and so on in addition to a forwarding path.

How then can we enable a pathway that permits research and experimentation to be performed in production environments whilst minimally impacting existing network services? In part, this is the same problem that Active Networks attempted to solve, but we believe that a much more conservative approach is more likely to see real-world usage.

We envision an integrated open-source software router platform, running on commodity hardware, that is viable as a research and as a production platform. The software architecture should be designed with extensibility as a primary goal and should permit experimental protocol deployment with minimal risk to existing services using that router. Internet researchers needing access to router software could then share a common platform for experimentation deployed in places where real traffic conditions exist. Researchers working on novel router hardware could also use the mature software from this platform to test their hardware in real networks. In these ways, the loop between research and realistic real-world experimentation can be closed, and innovation can take place much more freely.

1.1 Alternatives

Having motivated the need for an open router on which network research can be deployed, we discuss the alternatives in more detail—simulations and network testbeds.

First, we note that it has not always been so difficult to deploy experimental work on the Internet. Prior to the advent of the World Wide Web, the Net was predominantly non-commercial. Most of its users came from universities and other research labs. Whilst there was a tension between conducting research and providing a networking service, researchers could have access to the network to run experiments, develop and test new protocols, and so forth. With the advent of the Web, the network grew rapidly and commercial ISPs emerged. Even the academic parts of the network became reluctant to perform networking experiments for fear of disrupting regular traffic. In the commercial parts of the network, where interesting scaling phenomena started to emerge, it was nearly impossible to do any form of experimentation. Growth was so rapid that it was all ISPs could do to keep up with provisioning. These problems were recognised, and two main solutions emerged: network testbeds and network simulators.

First Workshop on Hot Topics in Networks, Princeton, New Jersey, October 28–29, 2002

Network testbeds rarely resulted in good network research. One notable exception was DARTnet [3], which used programmable routers that network researchers had access to. Among its achievements, DARTnet demonstrated IP multicast and audio and video conferencing over IP. It worked well because the network users were also the network researchers and so there was less tension involved in running experiments.

Over recent years, the majority of network research that involved testing protocols has taken place in network simulators such as *ns*. Among the desirable properties of simulators is the complete control they provide over all the parameters of a system, and so a large range of scenarios can be examined. Within the research community the *ns* simulator has been particularly successful[1]. Many researchers have contributed to improve *ns* itself, but an even greater number have used it in their research. Many published results are supported by publicly available simulation code and scripts. This has allowed for the direct comparison of contemporary networking algorithms and allowed for independent verification of results. It could therefore be argued that *ns* has increased the rigor of network research.

Conversely, it could equally well be argued that simulators make it *too easy* to run experiments and are responsible for numerous papers that bear little, or no, relationship to real networks. Accordingly there is understandable doubt about any claims until they've been demonstrated in the real world.

Even in skilled hands, simulators have limits. When work requires access to real traffic patterns, or needs to interact with real routing protocols, or relates to deployed implementations warts-and-all, there is no substitute for real-world experimentation.

2 Architecture and Requirements

We've hopefully convinced you that Internet research would be better off if changes and extensions could be provisionally deployed. No simulation or testing environment can provide similarly useful lessons—or convince conservative router vendors that extensions work well enough to deserve real-world deployment. We want a routing infrastructure that is at least partially open to research extensions. The infrastructure must, further, meet Internet robustness standards to merit deployment even on research networks like Abilene. But how can we make this happen? There are several possibilities:

- Individual researchers could convince big router vendors to implement their extensions. To put it succinctly, this seems unlikely.
- Vendors could open their internal APIs for experimental use. This also seems unlikely—vendors probably consider their APIs to be trade secrets, and don't want to facilitate an open market in routing software. Furthermore, legacy code inside most routers isn't particularly easy to extend.
- Researchers could deploy currently available routing daemons such as Zebra [14], MRTd [13] or GateD [10], on a conventional PC running an operating system such as Linux or FreeBSD. The principle problems here are extensibility, performance, and robustness. The forwarding paths on these operating systems are not optimally designed for performance under heavy load, or for extensibility. The routing daemons are not as robust as we would like, and not generally designed with extensibility as a high priority. Finally, they don't present an *integrated* router user interface to the network operator, which reduces the likelihood of acceptance.
- Researchers could develop a new open-source router, designed from the ground up for extensibility, robustness, and performance. It would take time to build such a router, and then to convince the network operator community that it met stringent robustness standards, but the resulting router would make a more useful research platform than any of the other choices.

We have chosen this last path, which we call the Extensible Open Router Platform, or XORP. Our design addresses the four major challenges in building an open-source router: traditional router features, extensibility, performance, and robustness.

Features. Real-world routers must support a long feature list, including routing protocols, management interfaces, queue management, and multicast.

Extensibility. Every aspect of the router should be extensible, from routing protocols down to details of packet forwarding. The router must support multiple simulatenous extensions as long as those extensions don't conflict. APIs between router components should be both open and easy to use.

Performance. XORP isn't designed for core routers, at least not initially. However, forwarding performance is still important: that is the purpose of a router. Scalability in the face of routing table size or number of peers is also critical.

Robustness. Real-world routers must not crash or misroute packets. A fragile router faces an extremely difficult deployment path. Extensibility makes robustness even more difficult to achieve: extensions are inherently experimental, yet their use should not compromise the router's robustness (except perhaps for a subset of packets intended for the extension).

The next sections present XORP in general and describe how we're achieving each of these goals.

3 XORP Overview

XORP divides into two subsystems. The higher-level (so-called "user-level") subsystem consists of the routing protocols themselves, along with routing information bases and support processes. The lower-level subsystem, which initially runs inside an OS kernel, manages the forwarding path—anything that needs to touch every packet—and provides APIs for the higher level to access. The goal is for almost all of the higher-level code to be agnostic as to the details of the forwarding path.

For user-level XORP, we've developed a multi-process architecture with one process per routing protocol, plus extra processes for management, configuration, and coordination. To enable extensibility we designed a novel inter-process communication mechanism for communication between these modules. This mechanism is called XORP Resource Locators (XRLs), and is conceptually similar to URLs. URL mechanisms such as redirection aid reliability and extensibility, and their human-readable nature makes them easy to understand and embed in scripting languages.

The lower level uses the Click modular router [6], a modular, extensible toolkit for packet processing on conventional PCs. Further work will help this architecture span a large range of hardware forwarding platforms, from commodity PC hardware, through mid-range PC-based platforms enhanced with intelligent network interfaces (such as Intel's IXP1200 [5, 12]), to high-end hardware-based forwarding engines. We may also support alternative forwarding

[1] One could criticize the details of the *ns* architecture and some of the default settings, but that would miss the point.

paths, such as the FreeBSD forwarding path with AltQ queuing extensions [2] or an alternative extensible forwarding path such as Scout [11]. Some forwarding path choices may influence the functionality available for end users to extend or change. But for many aspects of research, such as routing protocol experiments that don't require access to the forwarding path, a conventional FreeBSD lower level would suffice.

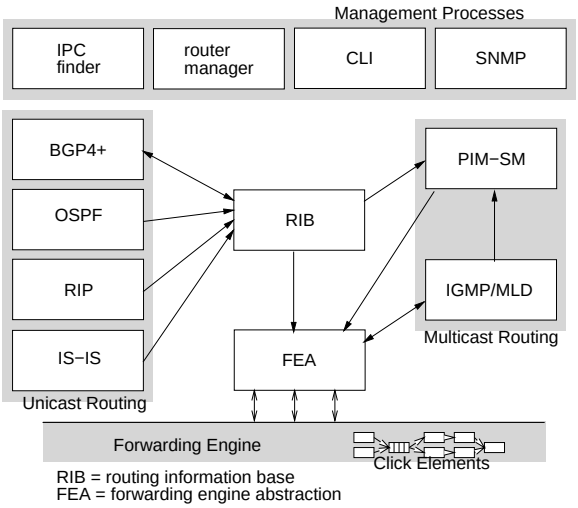


FIGURE 1—XORP High-level Processes

Figure 1 shows how a XORP router's user-level processes and Click forwarding path fit together. The shared user-level processes are the XORP architecture's most innovative feature. Four core processes are particularly worthy of comment: the *router manager*, the *finder*, the *routing information base*, and the *forwarding engine abstraction*.

The *router manager* process manages the router as a whole. It maintains configuration information; starts other processes, such as routing protocols, as required by the configuration; and restarts failed processes as necessary.

The *finder* process stores mappings between abstracted application requests, such as "How many interfaces does this router have?", and the particular IPC calls necessary to answer those requests. Think of it as an IPC redirector: when an application wishes to make an IPC call, it consults the finder to discover how to do it. The application typically caches this information so future calls circumvent the finder. Furthermore, the finder can instruct applications to update contact information. Thus it is easy to change how application requests are handled at run-time. We can trace XORP's communication pattern by asking the finder to map abstract requests to sequences of IPCs, some for tracing and some for doing the work. XORP processes can communicate without bootstrapping using the finder, but since XRLs are relatively low cost we have not found this necessary to date.

The *routing information base* process (RIB) receives routes from the routing processes, and arbitrates which routes should be propagated into the forwarding path, or redistributed to other routing processes. The forwarding path is managed by the *forwarding engine abstraction* process (FEA). The FEA abstracts the details of how the forwarding path of the router is implemented and as a result, the routing processes are agnostic to whether the forwarding plane is Click based, a conventional UNIX kernel, or an alternative method. The FEA manages the networking interfaces and forwarding table in the router, and provides information to routing processes about the interface properties and events occurring on interfaces, such as an interface being taken down. As with the finder, XORP processes can bypass the FEA when required.

4 SOLVING DESIGN CHALLENGES

This section shows how we address the four design challenges of traditional router features, extensibility, robustness, and performance. Space constraints prevent inclusion of much detail, but we do describe our IPC mechanism, XORP Resource Locators (XRLs), at length.

4.1 Features

To be successful, a router platform needs to be have good support for the routing and management protocols that are in widespread use today. The minimal list of routing and routing-support protocols is:

- **BGP4+** inter-domain routing.
- **OSPF** intra-domain routing.
- **RIPv2/RIPng** intra-domain routing.
- **Integrated IS-IS** intra-domain routing.
- **PIM-SM/SSM** multicast routing.
- **IGMPv3/MLD** local multicast membership.
- **PPP** for point-to-point links (as per RFC 1812).

With the exception of IS-IS, all of these are currently being worked upon within XORP. The aim is for both IPv4 and IPv6 support. MPLS is deliberately omitted at this stage. The multi-process architecture helps us to leverage existing code where appropriate; our OSPF and RIP implementations are derived from John Moy's *OSPFd* [9] and BSD's *routed* respectively.

For management interfaces, we are pursuing a command line interface resembling that of Juniper and intend to support SNMP.

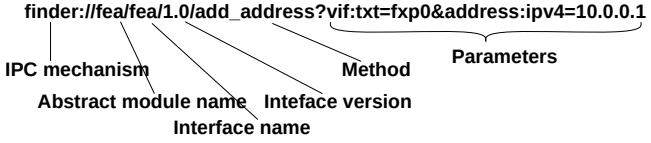
4.2 Extensibility

Open interfaces are the key to extensibility. Interfaces must exist at any point where an extension might plug in, and those interfaces must be relatively easy to use. XORP's design encourages the construction of useful interfaces through multi-process design. A routing protocol process, for example, must communicate with other processes to install routes and discover information about the router itself. Open inter-process interfaces, built in to the system from the beginning, form the basic source of user-level XORP's extensibility.

4.2.1 XRLs

Most inter-process communication within XORP takes place via XORP Resource Locators, or XRLs. XRLs resemble the Web's URLs. They specify in human-readable form the type of IPC transport desired (the "protocol"), the abstract name for the entity being communicated with, the method being invoked, and a list of named arguments. Unlike URLs, they can also specify the nature of the response expected.

As an example, the general form of one of the XRLs for the forwarding engine abstraction (FEA) process might be rendered in human readable form as:



The initial 'finder:' portion specifies the protocol; in this case the actual protocol has not yet been resolved. The first time this XRL is called, the client XRL library contacts the finder, which responds with a redirection to a new XRL containing the actual protocol to be used, together with all the parameters needed to contact the current FEA. Subsequent communication then goes directly between the client and the FEA process. If the FEA restarts, the client's XRL calls to the old FEA will fail, and it can consult the finder to update the redirection.

The XRL library, which all of our processes link against, takes an XRL and performs argument marshaling, then it invokes the specified transport mechanism, and handles the response or any failure that might occur. Unlike many RPC or remote method invocation mechanisms, XRLs don't try and hide from the programmer that off-process communication is taking place. While this makes the programmer's job harder at first, it is essential for robustness that the programmer is made aware that the failure modes of IPC are different from those of a local procedure call. To help the programmer and improve performance, an IDL and a stub generator exist, so most XRL clients never need to parse the human readable form.

The original motivation for XRLs was to encapsulate existing protocols within our consistent IPC framework. For example, we might wish to run third-party software that uses SNMPv3 for configuration. To integrate this software into our XRL-based management framework, we might write an SNMP 'protocol family' for the XRL client library. Then XORP processes could transparently interoperate with the third-party software via XRLs that start with 'snmp:'. XRLs are general enough to encompass simple communication with the kernel via `ioctl`s, and even signaling via `kill()`. At the present time, we have not had a need to add this functionality, but should the need arise, our architecture would support it. The current XRL library supports XORP-specific protocols for remote procedure call, one layered over TCP and the other over UDP, and a local procedure call mechanism for intra-process communication.

4.2.2 XRL Example: Command-line Interface

One of the biggest issues faced by an extensible router is the integration of separately maintained components into a coherent system. Consider the interaction between management mechanisms such as a command-line interface (CLI) and a new routing protocol. The author of each of the management processes has no knowledge of future routing protocols. At the same time, the author of each routing protocol has no knowledge of future management mechanisms. Our solution is for all management, including initial configuration, to take place using XRLs. To add support for a specific management mechanism, such as SNMP or a command-line interface, the protocol implementor writes simple text files that map management requests to XRL calls. These thin mapping files are easy enough to write that third parties might add them as new management interfaces become available.

To get more concrete, our configuration manager has a strict hierarchy of configuration parameters, which is directly reflected in our default CLI. A fragment of a router configuration file might look like:

```
protocols ospf {
    router-id: 128.16.64.1
    area 128.16.0.1 {
        interface xl0 {
            hello-interval: 30
        }
    }
}
```

The configuration manager takes a directory of template files, which define the possible configurable parameters for each XORP routing protocol, and generates mappings of configuration parameters to XRL dispatches. The designer of a new routing protocol can simply add a template file specifying the new functionality provided. Thus, the template entry for OSPF might contain the fragment:

```
hello-interval: uint {
    %set: xrl "ospf/ospf/1.0/set_hello_interval?
                    if:txt=$(IFNAME)&interval:i32=$(VALUE)";
    %get: xrl "ospf/ospf/1.0/hello_interval?if:txt
                    -> interval:i32";
}
```

The configuration manager can read the template file, discover the new functionality, and know how to communicate with the process to use it. The new functionality is then immediately available through the CLI.

4.3 Performance

Simultaneouly satisfying the three goals of extensibility, performance, and robustness without compromise is probably not possible. Different parts of XORP have different priorities. User-level XORP does not need particularly performance-conscious design. Route updates can arrive at high rates, but nowhere near the rates of packet arrival. It is well within the capabilities of a desktop PC to handle the volume of data and computation required for unicast routing. (The main issues here are of the computational complexity of the protocol implementations.) At user-level, our priorities are extensibility and robustness.

The forwarding path, however, must touch every packet and performance is paramount. Initially we have focused on a PC-based hardware platform as this allows for easy testing and early deployment, but great care is being taken to design a modular forwarding path architecture which can support some or all of the forwarding functions happening in hardware. For our own work, the preferred forwarding path is based on the Click modular router [6]. Click provides a high-performance forwarding path in the kernel, running on commodity hardware, and allows for run-time extensions and reconfiguration. Click's modularity allows for many divisions of work between software and hardware. If researchers or developers decide they want to augment XORP with network processors[2] or special-purpose hardware [7], this will be an advantage.

Click forwarding paths are extensible in two key ways:

- Existing defined elements can by interposed into a forwarding path to add new functionality.
- New Click elements can be created, loaded as kernel modules, and then plumbed in to the forwarding path.

In the context of XORP, this extensibility can aid performance. For example, a network administrator can ensure that only a small subset of traffic goes through a possibly-slow extension, by defining a new special-purpose classifier that matches extension traffic and inserting it into the relevant place in the forwarding path.

4.4 Robustness

The routing and coordination processes in XORP run in user space on a traditional UNIX operating system. Routing processes

[2]Network processors [5] are interface cards containing a number of high-speed network interfaces, typically together with a processor and some specialized programmable hardware for performing forwarding. Typically the network processor can perform simple forwarding very well, but more complex functionality needs to be off-loaded to be handled by software running on the host processor [12].

are protected from each other and can have their resources constrained according to administrative preference. Furthermore, routing processes can crash without affecting the kernel, forwarding plane, or each other. And if a routing protocol does crash, the RIB will remove its routes from the forwarding engine, and optionally inform the re-starting routing process of the routes it previously held.

Multiple processes are used in Zebra [14] and Cisco's proposed ENA router operating system, but not by some of the larger commercial vendors today.

A significant aspect of robustness is security. One benefit of being forwarding-path agnostic is that we can abstract privileged operations, such as sending on a raw socket, into the FEA via XRLs. This allows us to run many routing protocols in a sandbox. They have no interaction with the outside world except through XRLs and packets, and so an exploitable vulnerability in a routing protocol is far more difficult to escalate into full control of the router.

Robustness in the forwarding path is also important, but solutions such as memory protection that work well in user-space are not acceptable. In the Click forwarding path robustness comes from the granularity and simplicity of Click's components. Each element is small enough to be well understood and tested in isolation. And since many of Click's components can be run and debugged in user-space, confidence about their robustness can be attained before being used in the kernel.

5 SUMMARY

We believe that much good Internet research is being frustrated by an inability to deploy experimental router software at points in the network where it makes most sense. These problems affect a wide range of research, including routing protocols themselves, active queue management schemes, and so-called "middlebox" functionality such as our own traffic normalizer [4]. Many of these problems would not exist if the router software market more closely resembled the end-system software market, which has well defined APIs for application software, and high performance reliable open-source operating systems that permit kernel protocol experimentation. Our vision for XORP is to provide just such an open platform; one that is stable and fully featured enough for serious production use (initially on edge-routers), but designed from the very outset to support extensibility and experimentation without compromising the stability of the core platform.

There are many unanswered questions that we still have to resolve. For example, network managers simply wish to say *what* a router should do; they don't typically care about how this is implemented. Click starts from the point where you tell it precisely how to plumb together the forwarding path. Thus we need a forwarding path configuration manager that, given the network manager's high-level configuration, determines the combination of click elements and their plumbing needed to implement the configuration. The simplest solution is a set of static profiles, but we believe that an optimizing configuration manager might be able to do significantly better, especially when it comes to reasoning about the placement and interaction of elements such as filters.

Thus, while XORP is primarily intended to *enable* research, we also believe that it will also further knowledge about how to construct robust extensible networking systems.

Finally, while we do not expect to change the whole way the router software market functions, it is not impossible that the widespread use of an open software platform for routers might have this effect. The ultimate measure of our success would be if commercial router vendors either adopted XORP directly, or opened up their software platforms in such a way that a market for router application software is enabled.

ACKNOWLEDGMENTS

The XORP project is funded by grants from Intel and the National Science Foundation (Award ANI-0129541). Atanu Ghosh, Adam Greenhalgh, Mark Handley, Orion Hodson, Eddie Kohler, Pavlin Radoslavov and Luigi Rizzo have contributed significantly to XORP.

REFERENCES

[1] CAIRN. Collaborative Advanced Interagency Research Network (Web site). `http://www.cairn.net/`.

[2] K. Cho. A framework for alternate queueing: towards traffic management by PC-UNIX based routers. In *Proc. USENIX 1998 Annual Technical Conference*, pages 247–258, June 1998.

[3] D. D. Clark, S. Shenker, and L. Zhang. Supporting real-time applications in an integrated services packet network: Architecture and mechanism. *ACM Computer Communication Review*, 22(3):14–26, Oct. 1992.

[4] M. Handley, C. Kreibich, and V. Paxson. Network intrusion detection: Evasion, traffic normalization, and end-to-end protocol semantics. In *Proc. Usenix Security Symposium*, Aug. 2001.

[5] Intel Corporation. Intel IXP1200 network processor (Web site). `http://developer.intel.com/design/network/ixp1200.htm`.

[6] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek. The Click modular router. *ACM Trans. on Computer Systems*, 18(3):263–297, Aug. 2000.

[7] J. W. Lockwood, N. Naufel, J. S. Turner, and D. E. Taylor. Reprogrammable network packet processing on the field programmable port extender (fpx). In *Proc. ACM International Symposium on Field Programmable Gate Arrays (FPGA 2001)*, pages 87–93, Feb. 2001.

[8] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow. TCP selective acknowledgment options. RFC 2018, Internet Engineering Task Force, Apr. 1996. `ftp://ftp.ietf.org/rfc/rfc2018.txt`.

[9] J. Moy. *OSPF Complete Implementation*. Addison-Wesley, Dec. 2000.

[10] NextHop Technologies. GateD releases (Web site). `http://www.gated.org/`.

[11] L. L. Peterson, S. C. Karlin, and K. Li. OS support for general-purpose routers. In *Proc. 7th Workshop on Hot Topics in Operating Systems (HotOS-VII)*, pages 38–43. IEEE Computer Society Technical Committee on Operating Systems, Mar. 1999.

[12] T. Spalink, S. Karlin, L. Peterson, and Y. Gottlieb. Building a robust software-based router using network processors. In *Proc. 18th ACM Symposium on Operating Systems Principles (SOSP)*, pages 216–229, Oct. 2001.

[13] University of Michigan and Merit Network. MRT: Multi-threaded Routing Toolkit (Web site). `http://www.merit.edu/mrt/`.

[14] Zebra project. GNU Zebra routing software (Web site). `http://www.zebra.org/`.

A Blueprint for Introducing Disruptive Technology into the Internet*

Larry Peterson
Princeton University

Tom Anderson
University of Washington

David Culler, Timothy Roscoe
Intel Research – Berkeley

ABSTRACT

This paper argues that a new class of geographically distributed network services is emerging, and that the most effective way to design, evaluate, and deploy these services is by using an overlay-based testbed. Unlike conventional network testbeds, however, we advocate an approach that supports both researchers that want to develop new services, and clients that want to use them. This dual use, in turn, suggests four design principles that are not widely supported in existing testbeds: services should be able to run continuously and access a slice of the overlay's resources, control over resources should be distributed, overlay management services should be unbundled and run in their own slices, and APIs should be designed to promote application development. We believe a testbed that supports these design principles will facilitate the emergence of a new *service-oriented network architecture*. Towards this end, the paper also briefly describes PlanetLab, an overlay network being designed with these four principles in mind.

1. INTRODUCTION

The Internet was founded on a simple model in which the routers inside the network are responsible for forwarding packets from source to destination, and application programs run on the hosts connected to the edges of the network. However, the last few years have seen a blurring of the distinction between packet forwarding and application processing, as new widely-distributed applications are making their own forwarding decisions. This emerging class of applications includes network-embedded storage [12], peer-to-peer file sharing [16, 4], content distribution networks [20], robust routing overlays [17, 2], scalable object location [3, 15, 18, 14], and scalable event propagation [5]. At the same time, network elements such as layer-7 switches and transparent caches are performing application-specific processing.

These emerging services are the result of a convergence of two historically separate research communities. One is the distributed systems community, which traditionally viewed the network as merely providing bit-pipes between edge machines, but is increasingly embedding functionality at important crossroads and access points throughout the network. The other is the network community, which traditionally worried about forwarding packets without regard to application semantics, but is increasingly aware of the synergy that comes from unifying the computational and storage resources within the network with the requirements of the application.

We believe that the full integration of these two perspectives will result in a new, *service-oriented network architecture*. Unfortunately, this work is being done in an environment in which the Internet is less and less influenced by research, and increasingly shaped by commercial interests. In fact, a recent report from the National Research Council points to the ossification of the Internet [13]:

> ...successful and widely adopted technologies are subject to ossification, which makes it hard to introduce new capabilities or, if the current technology has run its course, to replace it with something better. Existing industry players are not generally motivated to develop or deploy disruptive technologies...

This paper offers a blueprint for introducing disruptive technologies into the Internet through the use of overlay networks. Overlays provide the right opportunity for innovation because they can be rapidly programmed to provide an innovative capability or feature, without having to compete with the existing infrastructure in terms of performance, reliability, administrative tools, and so on. Over time, we expect the "weight" such overlays place on the underlying Internet will result in the emergence of a new architecture, much in the same way the Internet (itself an overlay network) influenced the telephony network on which it was built. This paper concludes by speculating about what this new architecture might look like.

2. FROM DESIGN TO DEPLOYMENT

The example applications mentioned above are currently being designed and studied using a combination of simulation, network emulation, and small-scale testbeds. All of these systems would greatly benefit from a testbed that supports large-scale, real-world experiments. Overlays would play two roles in such a testbed: applications using the testbed will be structured as overlays, but the testbed itself is also an overlay from the perspective of controlling, managing, and deploying applications over it. We define such a testbed along three main dimensions.

First, the physical dimensions of the overlay network should

*This work is supported by Intel Corporation, NSF grant ANI-9906704, and DARPA contract F30602–00–2–0561.

be large—on the order of 1000 sites—to enable wide deployment of services and measurement tools. We envision the majority of these sites running a single overlay node that connects a large client population to the overlay. We can think of these nodes as providing a thousand viewpoints on the network. They should be selected to provide a rich diversity of link behavior, and wide-spread geographic coverage. We also envision the overlay including roughly a hundred sites with more substantial computing resources (e.g., a cluster of machines) located at network crossroads (e.g., peering points and co-location centers).

Second, the overlay consists of two main software components: (1) a *virtual machine monitor* (VMM) running on each node; and (2) a *management service* used to control the overlay. The VMM specifies the interface to which the services distributed over the testbed are written. This is a controlled interface to abstract resources (e.g., network ports and logical disks), rather than an interface that provides direct access to hardware. The management service is used to control the testbed; for example, to discover the set of nodes in the overlay, monitor their health, and to keep the software running on these nodes up-to-date.

The final dimension, which is the most distinguishing characteristic of the approach we advocate, is the overlay's mode of operation. Rather than view the overlay strictly as a testbed, we take the long-term view in which the overlay is both a research testbed and a deployment platform. In other words, the overlay should support the seamless migration of an application from early prototype, through multiple design iterations, to a popular service that continues to evolve.

Using an overlay as both a research testbed and a deployment platform is synergistic. As a testbed, the overlay's value is to give researchers access to (1) a large set of geographically distributed machines; (2) a realistic network substrate that experiences congestion, failures, and diverse link behaviors; and (3) the potential for a realistic client workload. Its value as a deployment platform is to provide (1) researchers with a direct technology transfer path for popular new services, and (2) users with access to those new services. We believe that supporting both roles is critical to the success of the system.

An important consequence of dual use is that the testbed must support the continuous operation of network services, as opposed to providing mechanisms for starting and stopping experiments. This leads to an obvious tension between the needs of "test & measure" researchers for reproducible results, and those interested in the system as a deployment platform. As an overlay, however, the bandwidth/latency/loss that can be achieved through the network is variable, and hence, unpredictable. The overlay's real value as a research platform is in providing realistic network conditions.

The dual use paradigm also implies that many experimental services will have to share nodes, since it is unreasonable to dedicate on the order of 1000 distributed nodes to a single experimental service for months at a time. This leads to strong security and resource control requirements on the virtual machine monitor.

Another way to understand the essential aspects of the approach we are proposing is to compare it to several current testbed efforts.

- Internet2, which includes the Abilene backbone, is a physical network that includes high-speed optical links connecting major research universities [10]. The network nodes are closed commercial routers, making it impossible to introduce new functionality into the middle of the network. In contrast, the main virtue of an overlay is that the nodes are fully programmable.
- Emulab is a network experimentation facility supported by the University of Utah [22]. Researchers are able to schedule a set of nodes for experiments, and dictate how the nodes are configured to emulate different network topologies. An overlay also has an experimental aspect, but its dual role as a deployment platform means that experimental systems run continuously, rather than for a limited period of time.
- The Grid is a collection of middleware, called Globus [6], that allows researchers to distribute large scientific applications across a distributed set of computational resources [9]. The main difference between the Grid and the proposed overlay is one of emphasis: the Grid is primarily interested in gluing together a modest number of large computing assets with high-bandwidth pipes, while the overlay is primarily concerned with scaling less bandwidth-intensive applications across a wider collection of nodes.
- The ABONE is an overlay testbed that grew out of the Active Networks initiative [1]. It allows service developers to dynamically load their applications onto the overlay's nodes. Although the high-level goals are similar, one important difference is that Active Networks is primarily focused on supporting extensibility of the network forwarding function, whereas we take a more inclusive view of the types of applications that will be deployed throughout the network, including those that involve a significant storage component.
- The XBONE is an overlay network with support for IP-in-IP tunneling [19]. It also includes a GUI-based toolset for establishing and monitoring specific overlay configurations. The XBONE and the proposed overlay share the goal of supporting multiple independent overlays on the same set of machines, but the XBONE is limited to IP tunnels, whereas we also hope to support higher-level overlays, such as those implemented by peer-to-peer systems.

An alternative to developing a new service on a traditional testbed is to package it as an application that can run on any desktop machine. If it proves to be a popular service, users will install it. File sharing systems like Napster and KaZaA have successfully adopted this approach, but it is not clear that it extends to other styles of services. More importantly, deploying services in the wild by viral dissemination has several shortcomings.

First, it does not work unless the service is immediately and widely popular. This makes it impossible, for example, to do

research studies into algorithms for managing overlay networks. Often, the technically superior solution will not be used if its applications or content happen to be less popular.

Second, it is difficult to modify such a system once it is deployed, making the process of learning from experience very cumbersome. The next version of the algorithms—or more generally, the next iteration of the design-evaluate-refine cycle—often requires a new set of compelling applications.

Third, such systems are not secure. The recent problem with KaZaA exposing all files on the local system is just one example of the potential dangers. We prefer a system that allows us to understand how to sandbox viral peer-to-peer applications so that users need not trust their entire systems to the coding standards of random peer-to-peer developers.

3. DESIGN PRINCIPLES

Our vision of an overlay that serves both service developers and service users has several implications for the architecture of the system. This section outlines the key design principles that shape such an overlay.

3.1 Slice-ability

Because services are expected to run continuously, rather than be globally scheduled to run one at a time, the overlay must support distributed virtualization. That is, each application acquires and runs in a *slice* of the overlay. Distributed virtualization, in turn, requires each node to multiplex multiple competing services. Thus, a key responsibility of the VMM running on each node is to allocate and schedule slices of the node's processor, link, and storage resources.

The node slicing mechanism must be *secure* in that it protects the node from faulty or malicious services. It must also use a *resource control mechanism* such as proportional share scheduling to enforce bounds on the resources consumed by any given service. Finally, it must be *scalable* in the sense that each node is able to efficiently multiplex resources among a large number of services.

Note that while each node is able to enforce slices of its local resources (including its outgoing link bandwidth), since the system is an overlay network, it is not possible to ensure that a given application receives predictable network performance, given that the Internet does not yet support bandwidth reservations.

Finally, in addition to viewing a slice as the collection of resources available on some set of nodes, a slice can also be characterized at the global level in terms of how those nodes (resources) are spread throughout the Internet. For example, one slice might contain resources that are uniformly distributed over as wide of area as possible, while another might wish to ensure that its resources are clusterd in autonomous systems with a high degree of connectivity.

3.2 Distributed Control of Resources

In its dual role as testbed and deployment platform, there will be two types of users: (1) researchers that want to install and evaluate new services, and (2) clients that want to access these services. Initially, the researchers are likely to be the only users (it is important that the researcher community develop applications that they themselves want to use), but in order to function as a deployment platform, the overlay must also provide explicit support for people that are willing to add nodes to the overlay for the sake of accessing its services.

These two user populations have different views of the nodes. Researchers want to dictate how their services are deployed. It may be as simple as "on as many nodes as possible" but they may also want to dictate certain node properties (e.g., at a crossroads site with sufficient storage capacity). Clients want to decide what services run on their nodes. They should be required to allocate slices of their machines to experimentation—thereby postponing future ossification—but they need to be able to set policy on how resources are allocated to different services.

This shared control of resources implies a highly-decentralized control structure. For example, a central authority may provide legitimate service developers with credentials that allow them to request a slice of a node, but each node will independently grant or deny such a request based on local policy. In essence, the node owner decides how many of the node's resources may be consumed by different services.

From a security perspective, applications have to trust both the central testbed authority and the physical security of the nodes at individual sites. Ultimately, this means service overlays need to be aware of where they cross administrative domain boundaries, and protect themselves from rogue elements.

3.3 Unbundled Management

Rather than view testbed management as a single, fixed service, overlay management should be unbundled into a set of largely independent sub-services, each running in their own slice of the overlay. For example, overlay management might be partitioned as follows:

- discover the set of nodes in the overlay and learn their capabilities;
- monitor the health and instrument the behavior of these nodes;
- establish a default topology among the nodes;
- manage user accounts and credentials;
- keep the software running on each node up-to-date; and
- extract tracing and debugging information from a running node.

Some of these sub-services are part of the core system (e.g., managing user accounts), and so there must exist a single, agreed-upon version. Others can be provided through a set of alternative services. The system will need to provide a default set of such services, more or less bundled, but we

expect them to be replaced by better alternatives over time. In other words, the management structure should be engineered for innovation and evolution.

To better appreciate the power of being able to run new management services in a slice of the overlay, imagine if we were able to go back to the days when IP was defined; we could then put in the instrumentation hooks we need today to measure Internet traffic. An overlay that ensures that some fraction of each node can be programmed will give us that ability.

The strategy of unbundling the management service requires that appropriate interfaces be defined. First, the individual services are likely to depend on hooks in the VMM that, for example, make it possible to retrieve the status of each node's resources. Second, the various sub-services may depend on each other; for example, the node monitoring service might provide input to a realtime database that is later queried by the resource discovery service. Allowing these two services to evolve independently will require a common representation for node attributes.

3.4 Application-Centric Interfaces

Perhaps the single greatest failure of testbeds, in general, is that they do not promote application development. One reason is that they are short-lived: experience teaches us that no one builds applications for pure testbeds since their lifetime is, by definition, limited. Related to this point, there is usually little motivation to integrate the testbed with desktop machines, making it nearly impossible for clients to access applications that might be available. The hope is that by designing the overlay to serve as both a research testbed and a deployment platform we will lower the hurdle for application development.

A more tangible problem is that it is difficult to simultaneously do the research needed to create an effective testbed, and use the testbed as a platform for writing applications. Users require a stable platform, which is at odds with the need to do research on the platform. To make matters worse, such research often results in new APIs, requiring that applications be written from scratch.

Thus, our final design principle is that the overlay must support an existing and widely adopted programming interface, with platform-related research changing the underlying implementation over time, while the API remains largely unchanged. Should an alternative API emerge from this effort, new applications can be written to it, but the original API is likely to be maintained for legacy applications.

4. PLANETLAB

We are currently building an overlay testbed, called PlanetLab, that adheres to these design principles. We envision PlanetLab achieving the goals outlined in this paper in three phases. Our strategy is to incrementally enhance the capabilities of PlatnetLab in accordance with the next user community we hope to attract.

Seed Phase: We have seeded PlanetLab with 100 machines and provided just enough functionality to meet the needs of a small, known set of researchers. These researchers are implementing and experimenting with many of the services mentioned in the introduction. We do not expect to support a client community during this phase, but instead PlanetLab will function as a pure testbed.

Researchers as Clients: We are now opening PlanetLab to the larger research community, which we expect to drive the size towards 1000 sites. We recognize, however, that adding clusters at strategic Internet crossroads will require broader government and industrial support. During this phase, the user community will still be primarily researchers experimenting with their services. We also expect these researchers will themselves begin to use primitive services provided by these applications.

Attracting Real Clients: Our thesis is that the research community is poised to develop innovative services, and that a true client base will follow. Accessing a service in this world is equivalent to joining an overlay network, and so we expect a growing client community to connect to PlanetLab. To the extent successful services are developed, we also expect to spin-off physically distinct copies of PlanetLab.

The hardware is dedicated PlanetLab nodes, as opposed to client-owned desktop machines. To minimize heterogeneity headaches, we prescribe the permitted hardware configurations. To ensure conformance with the common interfaces and policies, only the central PlanetLab authority (as opposed to node owners) have root-access to the machines. Moreover, while node owners will be able to establish policy on how their nodes are sliced, PlanetLab will retain the right to allocate some fraction of each node to experimental services. Finally, to ensure stability, PlanetLab will maintain a "core" of highly available nodes and sites.

The initial default management service is provided by a combination of the Ganglia resource monitoring tool [8], a boot and software update process based on Xenoboot [23], and an account/project management interface patterned after Emulab. We are currently evolving Ganglia into two separate components: a *resource monitor* that reports the resources available on a given node, and a *resource broker* that aggregates this information from a set of nodes, and responds to requests for slices.

Moreover, we are evolving account managment in a way that moves PlanetLab away simply providing a set of Unix accounts, towards a service-oriented architecture in which services dynamically create the slices in which they run. This is a three stage process: (1) a service manager first contacts a resource broker to select (discover) a set of candidate nodes that constitute a slice, (2) it then contacts the nodes in that set to initialize a network of virtual machines (this involves a per-node *admission control* decision), and (3) it launches the service in the resulting slice.

In terms of the VMM, our strategy is to evolve the kernel component of PlanetLab toward a strong notion of an *isolation kernel*, while maintaining an operational system that

is usable by researchers for both experimentation and long-term deployment. Towards this end, we are pursuing two complementary development efforts.

The first builds on a traditional Unix-like operating system that does not distinguish between the interface at which resource allocation and protection is applied (the *isolation* interface) and the system call interface used by program developers (the *application* interface). Speficically, we have settled on Linux since it is a popular platform for implementing network services. We then plan to augment Linux with functionality to provide better security and resource isolation between services running over it. The most attractive way of doing this includes virtualizing the kernel (rather than the hardware) in the style of Vservers [11], replacing privileged kernel functionality with safe alternatives (e.g., safe raw sockets), and adding support for slice-ability (e.g., resource containers plus proportional scheduling of the link and CPU).

The second effort revolves around isolation kernels like Denali [21] and Xenoservers [7], which provide a low-level isolation interface that closely resembles virtualization of the hardware. Operating systems such as Linux, BSD, and Windows XP can be ported to this virtual machine, an operation that is greatly simplified by the similarity between the virtual machine architecture and the "real" hardware the operating systems were originally developed for. The expectation is that it will be easier to assure the correctness of a minimal isolation kernel, thereby improving the overall security of the system.

5. CONCLUDING REMARKS

Just as the Internet was originally an experimental packet-switched network that evolved into a ubiquitous communication substrate over time, we believe it is possible to design a shared overlay infrastructure that can evolve from a modest research testbed into a planetary-scale service deployment platform. In fact, it is not an accident that our underlying design philosophy is similar to that of the Internet: we define the minimally required interfaces, and through the use of virtualization (slice-ability), the system is engineered to evolve.

There is a second interesting comparison point between the Internet and experimental testbeds like PlanetLab. The Internet was originally an overlay network that viewed the underlying telephony system as providing a collection of leased lines, but as it grew, the "weight" of the Internet eventually contributed to a complete re-design of the phone network. Likewise, overlay networks like PlanetLab initially view the Internet as providing a set of bit-pipes, but over time it is likely that the Internet architecture will evolve to better support service-level overlays. In other words, one of the most interesting questions emerging from this effort is how the interaction between the Internet and an overlay network like PlanetLab eventually results in a new service-oriented network architecture.

As an illustrative example of how this process might take place, it is already the case that multiple overlay services running on Planetlab independently probe the network as part of their topology-selection process. This is inefficient, and at the very least, there needs to be a shared "topology probing" mechanism. An additional step beyond such a mechanism would be to define an interface that allows overlays and the Internet to share topology information with each other. Eventually, one could imagine a few well-designed topology services evolving, with other services employing one of them rather than inventing a new one of their own. Whether a single "winner" emerges from this effort—and perhaps is subsumed into a new Internet architecture—or the very nature of a programmable overlay means that services will continue to define their own routing machinery, remains a subject of speculation.

6. REFERENCES

[1] ABONE. http://www.isi.edu/abone/.

[2] D. Andersen, H. Balakrishnan, M. F. Kaashoek, and R. Morris. Resilient Overlay Networks. In *Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP)*, pages 131–145, Chateau Lake Louise, Banff, Alberta, Canada, October 2001.

[3] M. Balazinska, H. Balakrishnan, and D. Karger. INS/Twine: A Scalable Peer-to-Peer Architecture for Intentional Resource Discovery. In *Proceedings of Pervasive 2002 - International Conference on Pervasive Computing*, Zurich, Switzerland, August 2002.

[4] F. Dabek, M. F. Kaashoek, D. Karger, R. Morris, and I. Stoica. Wide-area cooperative storage with CFS. In *Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP)*, Chateau Lake Louise, Banff, Alberta, Canada, October 2001.

[5] P. Druschel, M. Castro, A.-M. Kermarrec, and A. Rowstron. Scribe: A large-scale and decentralized application-level multicast infrastructure. *IEEE Journal on Selected Areas in Communications*, 20, 2002.

[6] I. Foster and C. Kesselman. Globus: A Metacomputing Infrastructure Toolkit. *Internationall Journal of Supercomputer Applications*, 11(4):115–128, 1997.

[7] K. Fraser, S. Hand, T. Harris, I. Leslie, and I. Pratt. The Xenoserver Computing Infrastructure, 2002. http://www.cl.cam.ac.uk/Research/SRG/netos/xeno/xeno-general.pdf.

[8] Ganglia. http://ganglia.sourceforge.net.

[9] Grid. http://www.globus.org.

[10] Internet2. http://www.internet2.edu.

[11] J. Gelinas. Virtual private servers and security contexts. http://www.solucorp.qc.ca/miscprj/s_context.hc.

[12] J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, and B. Zhao. OceanStore: An Architecture for Global-Scale Persistent Storage. In *Proceedings of the Ninth international Conference on Architectural Support for*

Programming Languages and Operating Systems (ASPLOS 2000), Nov. 2000.

[13] National Research Council. *Looking Over the Fence at Networks*. National Academy Press, Washington D.C., 2001.

[14] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker. Topologically-Aware Overlay Construction and Server Selection. In *Proceedings of the IEEE INFOCOM Conference*, New York, NY, June 2002.

[15] A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *Proceedings of the 18th IFIP/ACM International Conference on Distributed Systems Platforms (Middleware 2001)*, Heidelberg, Germany, November 2001.

[16] A. Rowstron and P. Druschel. Storage Management and Caching in PAST, A Large-Scale Persistent Peer-to-Peer Storage Utility. In *Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP)*, pages 188–201, Chateau Lake Louise, Banff, Alberta, Canada, October 2001.

[17] S. Savage, A. Collins, E. Hoffman, J. Snell, and T. Anderson. The End-to-end Effects of Internet Path Selection. In *Proceedings of the ACM SIGCOMM Conference*, Cambridge, MA, September 1999.

[18] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan. Chord: A Peer-to-Peer Lookup Service for Internet Applications. In *Proceedings of the ACM SIGCOMM Conference*, San Diego, CA, September 2001.

[19] J. Touch and S. Hotz. The X-Bone. In *Proceedings of the Third Global Internet Mini-Conference at Globecom '98*, pages 75–83, Sydney, Australia, November 1998.

[20] L. Wang, V. Pai, and L. Peterson. The Effectiveness of Request Redirection on CDN Robustness. In *Proceedings of the 5th Symposium on Operating System Design and Implementatio (OSDI)*, Boston, MA, December 2002.

[21] A. Whitaker, M. Shaw, and S. Gribble. Scale and Performance in the Denali Isolation Kernel. In *Proceedings of the 5th Symposium on Operating System Design and Implementatio (OSDI)*, Boston, MA, December 2002.

[22] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar. An Integrated Experimental Environment for Distributed Systems and Networks. In *Proceedings of the 5th Symposium on Operating System Design and Implementatio (OSDI)*, Boston, MA, December 2002.

[23] Xenoboot. http://www.cl.cam.ac.uk/Research/SRG/netos/xeno/boot/.

Predicate Routing: Enabling Controlled Networking

Timothy Roscoe
Intel Research at Berkeley
2150 Shattuck Avenue,
Berkeley, CA 94704, USA
troscoe@intel-research.net

Steve Hand
University of Cambridge
Computer Laboratory
Cambridge CB3 0FD, UK
steven.hand@cl.cam.ac.uk

Rebecca Isaacs
Microsoft Research
7 J.J. Thomson Avenue
Cambridge CB3 0FB, UK
risaacs@microsoft.com

Richard Mortier
Microsoft Research
7 J.J. Thomson Avenue
Cambridge CB3 0FB, UK
mort@microsoft.com

Paul Jardetzky
Sprint ATL
1 Adrian Court
Burlingame, CA 94010, USA
pjardetzky@sprintlabs.com

1. INTRODUCTION AND MOTIVATION

The Internet lacks a coherent model which unifies security (in terms of where packets are allowed to go) and routing (where packets should be sent), even in constrained environments. While automated configuration tools are appearing for parts of this problem, a general solution is still unavailable. Routing and firewalling are generally treated as separate problems, in spite of their clear connection. In particular, security policies in data hosting centers, enterprise networks, and backbones are still by and large installed manually, and are prone to problems from errors and misconfigurations. In this paper, we present *Predicate Routing* (PR) as a solution to this problem. We briefly describe our centralized implementation and then outline the extension of Internet routing protocols to support Predicate Routing.

In current IP networks, the state of the system is primarily represented in an imperative fashion: routing tables and firewall rulesets local to each node strictly specify the action to be performed on each arriving packet. In contrast, Predicate Routing represents the state of the network declaratively as a set of boolean expressions associated with links which assert which kinds of packet can appear where. From these expressions, routing tables and filter rules are *derived* automatically. Conversely, the consequences of a change in network state can be *calculated* for any point in the network (link, router, or end system), and predicates derived from known configuration state of routers and links. This subsumes notions of both routing and firewalling.

We use the phrase "controlled networking" to refer to environments where every packet flow in a network has been explicitly allowed or "white listed", possibly by an automated process. Controlled networking using Predicate Routing gives precise assurances about the presence of network packets, even when network elements cannot provide the filtering and packet discrimination required by a naive, manually configured approach.

Where ideal security is infeasible with the given infrastructure and topology, Predicate Routing can be used to guide risk assessments and security tradeoffs by providing complete information about what packets are allowed to traverse each link. Furthermore, Predicate Routing aids packet traceback, by allowing those properties of a packet (such as origin machine) which cannot be directly observed at most points in the network to be logically inferred from observable properties.

Note that this differs from the various forms of coordinated firewall control or distributed firewalling in that predicates do not specify traffic rules but rather capture both current and desired network state. From this state appropriate firewall rulesets may be synthesized.

Since our declarative view of network state is very different from traditional routing concepts, we first give an abstract view of how Predicate Routing represents a network, as a precursor to discussing its application. We then detail two scenarios where Predicate Routing can be applied: the control of flows and isolation of applications in a centralized environment such as a datacenter, and the use of distributed Predicate Routing algorithms to support controlled networking in the wide area.

2. PREDICATE ROUTING: FRAMEWORK

Predicate Routing is concerned as much with where particular packets in an IP network *can appear* as with where they should be sent. While the term "reachability" in conventional IP networks refers to the notion that some packets can reach a given point in the network, in Predicate Routing this notion is packet-specific, and subsumes both the notion of firewalling (ensuring that a particular destination is unreachable for that packet), and routing (attempting

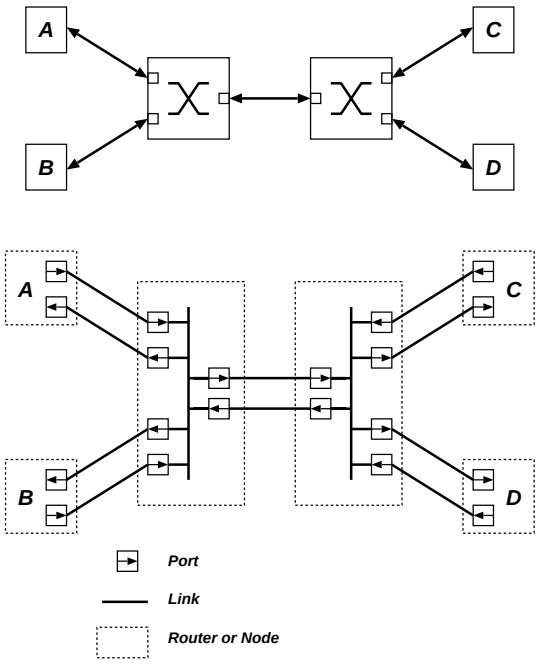


Figure 1: A traditional representation of a network, and the same network in Predicate Routing terms.

to ensure that the desired destination is reachable for the packet). Predicate Routing achieves this by employing a non-traditional abstraction of network properties. The upper half of Figure 1 shows a typical, simple IP network composed of 2 routers and 4 end nodes. Links are bidirectional, and connect ports on routers and nodes.

The lower half shows how the same network is represented in Predicate Routing. Some differences are immediately apparent. The most obvious is that the switch-centric representation has been replaced by one made up of ports and links. Indeed, in Predicate Routing the notion of a switch or router *per se* is important only from an implementation standpoint (as a collection of ports with a single control interface and shared resources). Ports are now *unidirectional*, and so there are twice as many. At the same time, links are regarded as broadcast media, and so are neither unidirectional nor bidirectional. Finally, the "inside" of a router or switch is equivalent to an external network link from the point of view of Predicate Routing.

While the network has become more complex (more links, more ports), the elements making it up have become much simpler, making it easier to automate reasoning about the network. Firstly, links are now passive media elements, and so it makes sense to talk about a packet being "present" on a link without needing to specify the direction in which it is traveling. Secondly, ports have a single input and a single output and have subsumed the role of switches and routers in the traditional representation, and so they can be viewed as "gates" which allow some packets through (possibly modifying them in the process) and disallow others. These two abstractions, (unidirectional) ports and (non-directional) links, form the basis for Predicate Routing.

2.1 Predicates

Predicate Routing views the state of an IP network as a set of logical expressions—predicates—that refer to properties of packets at each point in the network. In traditional routing, network state is represented as a forwarding table at each router, which can be viewed as a function from packet properties to outgoing router ports. In contrast, in Predicate Routing a packet can potentially appear on any router output port which does not explicitly disallow it; and the forwarding table is represented as a set of output filters. These two views of a router are equivalent (Predicate Routing is just as expressive), but the more declarative approach taken by Predicate Routing simplifies automated reasoning about network state.

The primitive terms of a predicate are packet attributes like source or destination IP address, port numbers, protocols, etc. A simple (and highly restrictive) link predicate might be:

```
Proto(TCP) AND DestPort(80) AND DestAddr(10.10.1.2)
```

While all the attributes in this example are directly observable from the packet header, one can define other attributes which are not immediately observable, such as the origin machine of the packet (in the presence of potential source address spoofing), a particular flow or path the packet is part of, etc. Predicate Routing can allow these non-observable attributes to be inferred from the network state. Routers generally operate only on observable properties of packets.

Four kinds of predicate are involved in representing network state: *link* or *network* predicates, *switch* predicates, *port* predicates, and *filter* predicates.

2.2 Link and network predicates

A *link predicate* is an assertion about the properties of packets that can be seen on a network link. Recall that links do not have a direction, so a single boolean expression in disjunctive normal form[1] suffices to describe everything that can be observed on the link.

While the idea generalizes to broadcast networks, where the predicate refers to then packets that can be present on a given segment, in this paper we restrict our discussion to switched point-to-point links.

2.3 Switch predicates

A *switch predicate* is an assertion about packets that may be seen "inside" a router or switch—packets which may potentially traverse the switching fabric. We treat the inside of a router as a "sea of packets", with no notion of which port a packet entered on, or which port or ports the packet is leaving on, hence there is a symmetry between the "insides" of switches and routers, and the "outsides" of links, with a corresponding symmetry between input and output ports.

This is a relatively simple model of a router. Modern IP routers are rather more complex than this: in particular

[1] i.e. an OR of a series of AND-connected compound terms.

many combine the functions of switching (based on MAC address) and routing (at the IP layer) using the concept of virtual LANs (VLANs). In this paper we use the terms switch and router interchangeably. We can capture this complexity of networking equipment in several ways. Firstly, if we treat VLANs as if they were real Ethernet broadcast domains, a Predicate Routing port now corresponds to the IP interface of the VLAN on the switch, as opposed to the physical ports. VLANs consequently have network predicates associated with them. A better approach integrates the VLAN notion into Predicate Routing's model of the switch, but that is beyond the scope of this paper.

2.4 Port predicates

A *port predicate* is an assertion about the properties of packets passing through a switch port. We view ports as unidirectional. A port predicate is identical in form to a link or network predicate.

2.5 Filter predicates

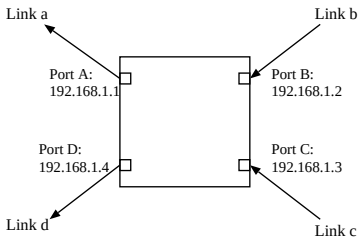
In the Predicate Routing model, input and output ports apply filters (which may be trivially simple). Thus, in addition to the port predicate (which asserts the properties of traffic flowing through the port), each port has an associated *filter predicate*, which asserts the properties of traffic which *can* flow through the port. The filter predicate for a port expresses the filter configuration which currently applies to the port. Most modern IP routers provide some facility for input port filtering, sometimes referred to as Access Control Lists, for which there is a natural mapping to input port filter predicates. Output port filter predicates are also naturally mapped onto real router configuration properties in the form of IP routing table entries. To understand this, consider the set of routing table entries in the router which cause packets to be forwarded to a given port. Each component term in the output filter predicate is the property that the packet destination address matches the address prefix of the table entry. The complete filter predicate is the OR of these terms.

Increasingly, all but low-end routers and high-performance core IP switches support policy routing, where a routing decision is made based not only on destination address, but also on source address, protocol, ports, etc. This additional router state information also maps naturally onto output port filters.

2.6 Relations between predicates

Figure 2 shows a very simple example of a network configuration, together with corresponding predicates which combine both the filtering and routing configuration of the network in a unified model of network state. It is clear that the four types of predicates are closely dependent on each other:

1. The port predicate for an input port on a switch is the AND of the network predicate of the attached network with the filter predicate for the port. This expresses what the port filter does: it constrains the traffic that enters the switch from the network through the port. Similarly, the port predicate for an output port on a switch is the AND of the switch predicate, and the filter predicate for the port.




```
Routing table:
 10.1.0.0/16 -> 192.168.1.1
 10.2.0.0/16 -> 192.168.1.4

Input filters:
 Port B:
  Allow Source = 10.3.0.0/16
  Deny all
 Port C:
  Allow Source = 10.4.0.0/16
  Deny all
```

Predicates:

Filter A:
  Dest = 10.1.0.0/16
Filter B:
  Source = 10.3.0.0/16
Filter C:
  Source = 10.4.0.0/16
Filter D:
  Dest = 10.2.0.0/16

Switch:
  Source = 10.3.0.0/16 OR Source = 10.4.0.0/16

Link A:
  Dest = 10.1.0.0/16
  AND (Source = 10.3.0.0/16 OR Source = 10.4.0.0/16)
Link B:
  None (anything)
Link C:
  None (anything)
Link D:
  Dest = 10.2.0.0/16
  AND (Source = 10.3.0.0/16 OR Source = 10.4.0.0/16)

Port A:
  Dest = 10.1.0.0/16
  AND (Source = 10.3.0.0/16 OR Source = 10.4.0.0/16)
Port B:
  Source = 10.3.0.0/16
Port C:
  Source = 10.4.0.0/16
Port D:
  Dest = 10.2.0.0/16
  AND (Source = 10.3.0.0/16 OR Source = 10.4.0.0/16)

Figure 2: A very simple network showing predicates

2. A link predicate is the OR of the port predicates for the switch output ports which are attached to the link. This simply expresses the fact that any packet enters a link through one switch output port.

3. Similarly, the switch predicate for a switch is the OR of the port predicates for all the input ports on the switch.

2.7 Discussion

Given these relations, we note that in a closed network, link, port, and switch predicates can be derived from knowing only the network topology and all the filter predicates.

Strictly speaking, the notion of a port predicate is redundant in this framework: port predicates are entirely determined by link and filter predicates and so don't convey any additional information about network state. However, they are

important as intermediate terms when applying the logical framework in a real implementation, as in the next section. Also, although Predicate Routing's logical framework treats networks and routers identically, in practice the difference clearly matters.

When the network is connected to other systems (like the rest of the Internet), these predicates can still be derived from a combination of the filter predicates and the link predicates at attachment points. Alternatively, predicates on links entering an administrative domain can be defined in accordance with service level agreements.

Note also that we can derive or "prove" additional properties of packets at given point in the network, not directly observable from the packet itself. For example, in Figure 2, if we see a packet on link **a** with a source address in 10.3.0.0/16, we can infer that the packet traversed link **b**. While this example is trivial, in more complex networks this can be a powerful tool for reducing human error in network configuration.

The use of declarative languages to detect network and system misconfiguration has been investigated in [11]. Predicate Routing provides the logical framework in which these techniques can be applied to network management, for example automatic detection of BGP misconfiguration, a well known problem [9].

The four types of predicates, together with the relations between them, faithfully model the packet forwarding behavior of an IP network, both filtering and routing, and together form a consistent logical system. However, this representation of network state is also highly amenable to manipulation by programs controlling network elements. Because it gives complete information of possible paths traversed by packets, it is highly appropriate for implementing a controlled networking environment. In the next section, we describe one approach to this, using a logically centralized approach for managing and controlling the network in a datacenter.

3. CENTRALIZED PREDICATE ROUTING

We applied Predicate Routing to the problem of controlled networking in a shared hosting platform or datacenter, where (possibly distributed) third-party applications are hosted on a shared cluster of servers. In this case it is reasonable to implement a centralized network "control plane" with out-of-band control of network elements. Here we sketch the algorithm we use and our prototype design; in Section 4 we outline how Predicate Routing can be applied in the distributed case to a larger network.

The system allows flows to be placed securely in the network quickly and efficiently. Flows can also trace multiple paths, allowing for fault tolerance in the face of link or router outages. This function must be performed online and incrementally, as the application mix is dynamic.

We define the meaning of "securely" in this context as follows. Every network link or computer in the datacenter may have an associated predicate which specifies which packets can be allowed on the link or arriving at the machine. This predicate represents an externally imposed, site-specific security policy; we can imagine that in many cases interior network links will have no restrictions on traffic, whereas vulnerable end-hosts might have strict limits on their exposure to potentially malicious traffic. Using predicate routing, we can guarantee that when a flow is placed in the network:

1. The connectivity of all existing flows is unaffected,
2. No policy predicates on links or nodes are invalidated.

If the system fails to place a flow, it provides precise information as to where the process failed, i.e. which security constraint would have been violated and the new link predicates that would result.

Our prototype at Sprint Labs is a network composed of two Enterasys SSR-8600 switch-routers and a front-end Cisco 11800 Layer-7 switch, which connect 36 servers (each with dual network interfaces) to each other and to the Internet. Both types of switch support policy routing and input port filters at the IP layer. The control plane runs externally on another machine.

Since we are dealing with real networking equipment, a key aspect of our system is the notion of a *switch driver*, which provides an encapsulation both of the capabilities of a particular piece of hardware, and a means to configure the router in real time. The control plane instantiates switch drivers objects for each switch in the cluster. The driver models a switch's capabilities (how many per-port or per-line card filters are supported, for instance), and establishes a control connection to the real hardware (through SNMP or a command line emulator). In addition, the driver exports an interface to the routing algorithm corresponding to Predicate Routing's "ports and links" representation.

For each physical port in a switch, the switch driver maintains state consisting of the current filter predicate and port predicate for the port, and the current *path list* for this port, i.e. the current set of paths which pass through the port.

For input ports to switches, the filter predicate is the current filtering configuration for the port. For output ports, it is the result of current static and policy routes configured on the switch which route traffic out through this port. The filter predicate is stored as a disjunction (logical OR) of tuple expressions, which corresponds well to the filtering functionality exposed by IP routers.

From the path list, a flow list can be constructed, and from the specification of each flow, a list of tuples, i.e. a predicate. Ideally, this predicate would precisely coincide with the filter predicate but in practice this doesn't always occur, either because the filtering functionality of the port is limited (for example, router output ports often have little filtering functionality other that provided by static routes), or because the router's capacity (in terms of filters) has been already exceeded.

The routing algorithm to place a new flow first calculates a candidate path for the flow, then operates a flooding algorithm starting at the flow origin. For each port encountered,

the switch driver is consulted to appropriately modify its filter predicate: either to let the flow through if the port is on the candidate path, or else to block packets from the flow (and any other unauthorized flows). The flooding stops at any port whose port predicate is unchanged as a result of the operation. The path is rejected if a link predicate at the edge of the cluster (i.e. at a server) violates an administrative security constraint, implying that placing the path would allow unacceptable packets to arrive at a host.

The switch driver abstraction allows great flexibility: a port is free to not apply a requested filter due to lack of functionality or the switch running out of resources, as long as the new flow is admitted along the candidate path. In this way the consequences are propagated "downstream", where even in a simple network other ports will often compensate and preserve the controlled environment.

Performance with our prototype is adequate, even though the control plane is implemented in the interpreted language Python. While the theoretical complexity of the algorithm is moderately high[2], in practice most loops terminate early with the reasonably powerful switch capabilities we have, resulting in much better scaling than might be expected, even with larger topologies. Communication latency with switches tends to dominate; this can in many cases be overlapped with the route computation.

4. DISTRIBUTED PREDICATE ROUTING

Controlled networking is also useful in the wider area Internet, although in this case a centralized scheme is not suitable. In this section we discuss how one might implement Predicate Routing in the Internet by modifying existing Internet routing protocols, specifically the IGP IS-IS and the EGP BGPv4.

4.1 Link-State Routing Protocols

IS-IS [3] is a link-state protocol adapted from the ISO CLNS protocol. Each router effectively broadcasts information about the other routers to which it is connected (its *link states*) in the form of *link state PDUs (LSPs)*. Routers store the LSPs they receive in a database, and then run a shortest path algorithm over this database to discover the interface on which they should transmit packets for destinations within the network. Much of this discussion also applies to OSPF, the other main link-state intra-domain routing protocol in use in the Internet today.

The link-state information is transmitted in variable length type-length-value fields appended to the LSP header information. As IS-IS was not originally intended for routing IP, it effectively distributes two forms of link-state information: the connectivity information, expressed in terms of CLNP nodes[3] and their adjacencies, and the IP information, expressed in terms of the IP prefixes available to a CLNP node.

We can extend IS-IS as follows. First, rather than simply advertise the destination IP prefixes available at a node, a set of predicates are advertised, potentially with associated resource usage information. Second, although LSP forwarding and database building takes place as normal, sets of predicates effectively form *views* of this database, defining the connectivity available to that set. The shortest path computation is run over each such view, producing a set of shortest path results, one for each collection of predicates. These can then be remerged, as allowed by the predicates in place, to create the forwarding tables to be used to actually route packets. This results in one (or more) forwarding tables that contain predicates to be applied to packets, and for each predicate, a corresponding output port on which packets can be transmitted.

4.2 Performance Implications

The performance impact of the above scheme can be separated into traffic and computation costs at both the data and control planes. The traffic impact is fairly easy to imagine: the network sees less user traffic (due to packets being filtered early), but more control traffic (since LSPs are now larger and potentially more frequent). If we expect predicates to be slowly varying (e.g. changing on the order of hours), the increased routing protocol bandwidth should not be significant.

Perhaps greater concerns are the additional computational overhead, and the risk of increased routing instability. In terms of the former, it is true that some additional overhead will occur due to the need to perform shortest path computations for every "view" of the network. However several factors mitigate this cost: firstly, we expect the number of views to be much smaller than the number of predicates, with many predicates mapping onto an empty or unconnected subgraph. Secondly, it is possible in some cases to infer shortest paths for smaller subgraphs; and thirdly, many subgraphs will be considerably smaller than the entire network (and may even be degenerate). Forwarding table performance should also not be an issue [6].

We don't expect our modifications to decrease routing stability, since the same topological information is communicated both cases. However, further investigation is a topic for future work.

4.3 External gateway protocols

Unlike OSPF and IS-IS, BGP is a path-vector routing protocol without an explicit notion of a link. Instead, each router advertises a cost to the destinations it can reach, and chooses e.g. the cheapest route to a particular destination. It then re-advertises its chosen routes to other routers, adding in a cost component to account for their own presence on the path to the destination.

BGP already has extensive support for filters, for routers to control the routes advertised to other routers and the route advertisements received from other routers. However, these filters are currently entered and managed manually. It would seem that the natural way to implement predicates in BGP is to extend BGP to allow automatic distribution and installation of filters, but the details of such an approach, in particular how to deal with transferring such information between administrative domains, are future work.

[2]A detailed analysis is beyond the scope of this paper.

[3]Each node in the network must be assigned a CLNP address, even if the network will only route IP traffic.

4.4 Discussion

Predicate Routing permits incremental deployment: as network routers are upgraded to support the routing protocol extensions described above, the inferences which may be made about the state of the network become stronger. Even with a small number of enhanced routers, however, useful information is available to operators. For example, access routers could be upgraded initially which suffices to provide automated management of ingress filtering. As incremental deployment proceeds, the ability of the system to infer the origin(s) of traffic generated by attacks (for instance) increases.

A practical deployment of Predicate Routing would benefit from the ability to compare the desired and actual network state. This requires a mechanism to accurately snapshot the current network configuration. This presents a challenge in a highly dynamic environment such as the Internet and is a matter for future work.

Enhancement of Predicate Routing as presented should include the ability to refer to predicates as first class entities, in particular across administrative boundaries. This naming of predicates enables scoping and modularization thereby allowing aggregation and transformation of predicates, late binding of policy and information hiding between networks.

5. RELATED WORK

Predicate Routing builds on several ideas from the areas of firewalling, virtual private networks and signaling.

Like distributed [1] or embedded firewalling, we aim to have an explicit notion of which packets may be transmitted where, and we attempt to automatically enforce this notion at multiple redundant locations. We do not rely upon a centralized security policy, but if end-users or end-user groups were to desire a shared security policy, we can envisage using the KeyNote trust management language [8] for example.

Another, more "overlaid" approach to the problems that Predicate Routing solves is Virtual Private Networks (VPNs). These may be constructed over IP by using tunneling; i.e. encapsulating packets prior to routing them [5]. Using Predicate Routing, a VPN is defined simply as a set of predicates, obviating the need for tunneling. Isolation from other network users is achieved "for free", and changes in VPN topology are supported by the modification of Predicate Routing paths. Similar arguments apply to IEEE VLANs [7] in the local area.

Predicate Routing also has much in common with the *hose model* [4] in that end-points are explicit (being described by predicates) while network paths are implicit.

The *network calculus* [2] provides a framework for reasoning about traffic queuing patterns in networks, based on the Min-Plus algebra. Network calculus provides a way to extend Predicate Routing with notions of Quality of Service and traffic engineering. By attaching network calculus expressions to flow terms in link predicates, link utilizations and latency bounds can be calculated as part of the predicate calculation. This is another promising area for future work.

6. CONCLUSION

We have presented Predicate Routing, a unified model of routing and firewalling in IP networks, and outlined both centralized and distributed implementations. Predicate Routing facilitates the *controlled networking* required to evolve the Internet toward a secure and robust infrastructure without the need for extensive protocol redesign. Our current work centers on deploying a Predicate Routing-based secure overlay network using the PlanetLab [10] testbed.

Acknowledgments

Christos Gkantsidis wrote the first implementation of Predicate Routing for the Sprint Labs cluster. We thank Bryan Lyles and Jon Crowcroft for their insights and discussions.

7. REFERENCES

[1] S. M. Bellovin. Distributed firewalls. *;login:*, pages 37–39, Nov. 1999.

[2] J. Y. L. Boudec and P. Thiran. *Network Calculus.* Springer Verlag LNCS 2050, June 2001.

[3] R. W. Callon. Use of OSI IS-IS for Routing in TCP-IP and Dual Environments. RFC 1195, December 1990.

[4] N. G. Duffield, P. Goyal, A. Greenberg, P. Mishra, K. K. Ramakrishnan, and J. E. V. der Merwe. A flexible model for resource management in virtual private networks. In *Proceedings of SIGCOMM*, volume 29 (4), pages 95–108, Sept. 1999.

[5] B. Gleeson, A. Lin, J. Heinanen, G. Armitage, and A. Malis. A framework for IP based virtual private networks. RFC 2764, Feb. 2000.

[6] P. Gupta and N. McKeown. Packet classification on multiple fields. In *Proceedings of SIGCOMM*, volume 29 (4), pages 147–160, Sept. 1999.

[7] IEEE. *IEEE Standards for Local and Metropolitan Area Networks: Virtual Bridged Local Area Networks (802.1Q)*, 1998.

[8] S. Ioannidis, A. D. Keromytis, S. M. Bellovin, and J. M. Smith. Implementing a Distributed Firewall. In *ACM Conference on Computer and Communications Security (CCS'00)*, pages 190–199, Nov. 2000.

[9] R. Mahajan, D. Wetherall, and T. Anderson. Understanding BGP Misconfiguration. In *Proceedings of SIGCOMM 2002*, pages 3–16, August 2002.

[10] L. Peterson, D. Culler, T. Anderson, and T. Roscoe. A Blueprint for Introducing Disruptive Technology into the Internet. In *Proceedings of the 1st Workshop on Hot Topics in Networks (HotNets-I), Princeton, New Jersey, USA*, October 2002.

[11] T. Roscoe, R. Mortier, P. Jardetzky, and S. Hand. InfoSpect: Using a Logic Language for System Health Monitoring in Distributed Systems. In *Proceedings of the 2002 ACM SIGOPS European Workshop, Saint-Emilion, France*, September 2002.

Feedback Based Routing *

Dapeng Zhu dapengz@cs.stanford.edu
Mark Gritter mgritter@cs.stanford.edu
David R. Cheriton cheriton@cs.stanford.edu

Department of Computer Science
Stanford University
Stanford, CA 94305

ABSTRACT

In this paper, we describe the problems that affect availability in BGP, such as vulnerability to attacks, slow convergence time, and lack of scalability. These problems arise from the basic assumption of BGP: every router has to cooperate to make routing work. We propose a new routing system, feedback based routing, which bifurcates structural information and dynamic information. Only structural information is propagated. Dynamic information is discovered by the routers based on feedback and probes. Routing decisions are made based on the dynamic information. We argue that this system is resilient to minority compromises in the infrastructure, provides higher availability than BGP, and can scale to the size of the Internet of the future.

1. INTRODUCTION

BGP is the current inter-domain routing system. In BGP, a router receives structural and dynamic information about the Internet from its peers, builds a model of the network from this information, then derives its routing table from this model [18]. Effective packet forwarding depends on this routing information being accurate and timely, and every router building a reasonably consistent model of the network connectivity and capacities.

This model worked great in the early days of the Internet, when the number of networks was small and the network operators trusted each other. The situation is different now. The networks connecting to the Internet are very diverse. It is not clear that a network operated by the U.S. government trusts a network operated by someone sympathetic to the terrorists. The number of networks, or in BGP terminology, prefixes that are visible in the BGP routing table is growing exponentially [11]. As a result, the amount of routing messages a router has to handle is growing exponentially [13]. Finally, the distributed nature of the BGP algorithm makes it extremely difficult to predict its behavior. For example, researchers have discovered that some routing policy combinations can cause BGP to diverge [23]. Many surprising discoveries like this([9], [13], [22], [21]) show the unpredictability and instability of BGP. These changes made three problems of BGP more prominent.

First, BGP is vulnerable to attacks from a single router. Such a concern is demonstrated to be valid by a failure [5] in which a misconfigured BGP router advertised to its peers that it had a direct route to every address in the Internet. This false information disrupted a large portion of the Internet for several hours. Recently, CERT warned of increasing focus on compromising routers by the hacker community [10], where there are active discussions on attacking routing protocols [6]. Going beyond the "prank" level of threat, there are concerns arising from recent terrorist events of the feasibility and effect of a serious attack on the Internet infrastructure.

There is a great deal of concern about the scalability of BGP, as the number of edge networks grow exponentially. This exponential growth caused some researchers to question whether BGP routers' CPUs can handle the amount of BGP messages that will be generated by tomorrow's Internet [14]. The correct functioning of BGP requires the cooperation of every router involved, and if some of the routers become overwhelmed with BGP updates, the whole routing system will take longer to converge.

BGP also suffers from slow convergence time, as a result of design decisions, the distributed decision making process, and the exponential growth of the Internet. Internet measurements have shown that BGP takes hundreds of seconds to converge after a routing failure [13]. Labovitz et al. also suggested that the convergence time will increase as the Internet grows [13]. As more mission critical communication systems, such as air traffic control and emergency communication, start to use the Internet as the underlying infrastructure, such service outages are simply unacceptable.

In this paper, we explore an alternative approach to inter-domain routing. In this system, we separate routing information into its structural and dynamic components. Structural information denotes the existence of links and is propagated to the edge of the Internet. Dynamic information denotes the quality of paths across the Internet. The routers

*This project is supported by the US Defense Advanced Research Projects Agency (DARPA) under contract number MDA972-99-C-0024.

at the core of the Internet only propagate structural information and forward packets. All routing decisions are done at the edge, based on structural information and end-to-end performance measurement.

This paper first describes the feedback based routing algorithm in section 2. We then analyze resistance to attacks, convergence time, and scalability in section 3. We discuss a mechanism to defend against Denial of Service attack, and a mechanism to implement fast fail-over virtual links in section 4, as applications of the routing system we propose. Related work is described in section 5.

2. ALGORITHM

2.1 Overview and Terminology

In our system, there are two types of inter-domain routers: transit and access. Figure 1 illustrates the location of these routers. Transit routers are usually located at the border of autonomous systems, while access routers are usually located at the border of an edge network. An edge network is the network of an organization. It is represented by one or more prefixes.

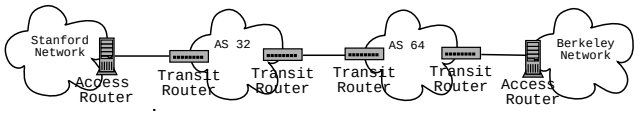


Figure 1: Access Routers and Transit Routers

Transit routers do not compute routing tables. They perform the following three functions: first, they forward packets according to routing information, which we call Internet Relay Tokens (IRT), in the packets. Second, they filter packets according to local access control list. Finally they propagate structural information. Access routers are responsible for inserting Internet Relay Tokens into IP packets. An IRT contains the list of autonomous systems a packet is going to travel. It is source routing at the granularity of autonomous systems. The packet format is specified in [7], part of the Wide-area Relay and Addressing Protocol (WRAP) specification.

In essence, our routing algorithm is composed of structural information propagation, route quality monitoring, and measurement based routing preference. An access router computes two routes to every advertised network prefix, based on the structural information it receives. The two routes are chosen to be as independent as possible. The access router monitors the quality of these two routes, and uses the better one to carry its traffic.

For the purposes of our algorithm, the Internet is modeled as a directed graph consisting of a vertex for each autonomous system and each edge network. The edges between vertices represent peering relationships; each edge may be either a single network link or many separate physical links. Since most Internet peering arrangements are bidirectional, the peering between autonomous systems A and B is represented by two directed edges $A \rightarrow B$ and $B \rightarrow A$. The peering between an autonomous system A and an edge network B is represented by one directed edge $A \rightarrow B$, since most edge networks do not carry traffic destined for other networks.

2.2 Structural Information Propagation

We associate a time stamp and an expiration time with every edge. An edge is removed from the structural graph after the expiration time is reached, and no new announcement is heard. The expiration timer should not be less than an hour. Routers announce the existence of edges periodically, but loss of edges are not explicitly advertised. Thus there are no link withdraw messages in our system.

For each edge $A \rightarrow B$ in the graph, there are three sets of advisory information associated with it, which determines whether a packet can be forwarded from A to B. The information is expressed in a packet matching rule. Therefore we also call the information rule sets. Here is an example: A packet with source address `136.152.197.176` and destination address `171.64.79.1` will match the rule `dst = 171.64.0.0/16` and the rule `src = 136.0.0.0/8`. In the expected case, the rules are simple matches on source and/or destination addresses. But they can be arbitrarily complex.

The first rule set is the positive rule set. If a packet does not match any rule in this set, then the edge does not exist with respect to this packet. The second rule set is called the negative rule set. If a packet matches any rule in this set, the edge does not exist with respect to this packet. Traffic engineering rule set is the third kind. Access routers that respect this rule set will not send packets that do not match any rule in this set. A transit router might have to enforce some rules (i.e. drop the packets that do not obey the rules), as a result of a contractual agreement. But they are free to ignore any rule if such an agreement does not exist, since it costs resource to filter packets. If we view BGP as a reachability maintenance protocol, then our protocol does not only maintain reachability, but also unreachability.

Neighboring vertices exchange information about edges they know about. Routers inside a vertex exchange connectivity information about their neighbors using an interior routing protocol.

Digital signatures can be used to increase the difficulty of attacks against the system. In a system without digital signatures, suppose there is a link from AS 32 to `128.12.0.0/16`, a malicious router might modify or generate a routing advertisement for this link which has a negative rule set of `0.0.0.0/0` — i.e., no packet is allowed to flow through the link. This modified rule can replace the valid link advertisement, causing a loss of connectivity to `128.12.0.0/16`. If both end points of an edge have to sign an edge, such an attack will not happen. However, digital signatures do not prevent other types of attacks. For example, a compromised router can announce an edge it has with a peer, but refuses to forward packets through this edge. If digital signature is used, an access router will not use an edge in its route computation unless its signature has been verified. The additional verification does not decrease the scalability of the

protocol because the computation is off the critical path. In other words, even if an access router can not keep up with edge verification, it can still function correctly.

2.3 Algorithm for Access Routers

Upon receiving structural information, access routers build a graph representation of the Internet. This representation incorporates the advisory information for each link. Then for each destination, the access router tries to find two routes. The two routes should differ as much as possible. In the ideal case, when both the source and destination are multihomed, the two routes should be vertex disjoint. One route will be used as the primary route, and the other serves as the backup route.

When the access router first boots, it chooses the route with the shortest AS hop count as the primary route. In subsequent operations, the cost of a route is the round trip time to the destination. The access router chooses the route with the lowest cost as the primary route.

An access router can sample the round trip time (RTT) to a destination D by measuring the time between a TCP SYN packet and the corresponding SYN ACK packet. (Here, we focus on TCP traffic, the dominant type of Internet traffic, especially in the wide area.) It keeps a running average of the round trip time, similar to TCP [17]. An access router is the ideal place to monitor the feedback because all traffic between the organization and Internet flows through it. We assume that there is one access router per organization, since here is no reason for more than one. Standard techniques can be applied to increase the availability of the access router.

Occasionally, an access router generates probes to test the quality of a route, such as that of a newly calculated backup route. The SYN-SYN-ACK sampling of TCP traffic above allows the router to record a promising address in the destination address range to probe with. For example, it may observe a TCP SYN to port 80, which suggests that a web server is present. In this case, it can send a TCP SYN to the same specific address and destination port over this route and time the round trip time to the SYN ACK. In the absence of an identified server, the router uses an ICMP echo to attempt to measure the round trip time. As another alternative, the router can try to use the backup route to forward packets and measure the actual delay in the same way as for the primary route.

Figure 2 illustrates the operation of an access router at Stanford. Through feedback and probes, it learns that RTT to Berkeley through the upper route is 20ms, while the RTT through the lower route is 100ms. Therefore it uses the upper route to forward packets to Berkeley.

An access router periodically computes a primary route and a backup route based on its current view of the Internet. The computation also takes current routes into consideration. If both routes to a destination have an infinite RTT in the routing table, the edges in the two routes are excluded when computing the two new routes (except the edges that connect the source and destination to the Internet. We either know or have to assume that they are working.) The

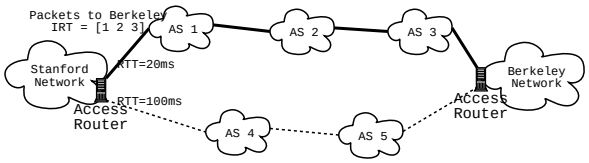


Figure 2: Primary and Backup Routes

results from the computation are used as the primary and backup routes. The computation can fail with very small probability. In that case, we randomly exclude some old edge from the two routes and redo the computation.

If only the backup route has an infinite RTT, The vertices of the primary route, and the edges of the backup routes are excluded from the computation. The result is used as the backup route.

If neither route has an infinite RTT, only the vertices of the primary route are excluded from the computation to find a new route. If the result is different from the current backup route, it is probed. If the RTT is less than the RTT of the current backup route, it is used as the backup route.

3. ANALYSIS

3.1 Attack Resistance

One of the attacks against routing protocols is fooling other routers into using a link, while the link either does not exist, or does not function well. If digital signatures are used, then all the edges an access router knows are real. But one end point of the edge can drop all the packets that are supposed to go through the edge, thus creating a black hole.

Our routing system is robust with respect to such attacks. An access router constantly monitor the quality of a route it is currently using. If it can not reach the destination, it would try a different (at least edge disjoint) route, according to the algorithm described in section 2.3.

However, our scheme fails when a compromised router generates a bogus SYN ACK packet in response to a SYN packet from a client. This can fool the access routers into forwarding packets through a compromised router. Because a misbehaving router can intercept and respond to any packets that go though it, we argue that this problem can only be addressed by user level verification and/or authentication. As an illustration of the problem, a compromised router can act as a web server. All user HTTP requests to `CNN.com` that go through this router will be intercepted and answered with bogus pages, such that only an intelligent being can detect the difference. Our system allows an access router's clients to inform it of bad routes. Our future research will address the problem of bogus complaints.

3.2 Scalability

There are three reasons why our system scales better than BGP. Route computation and propagation are removed from the critical path. The availability of the routing system does not depend on in-time computation of shortest paths any

more. Although computation is still needed to incorporate updated information, it can be delayed. Because there are two routes to any destination, if one of them fails, packets can be forwarded using the other one. Since the two routes are as independent as possible, the probability of both of them failing is much lower than that of one.

The amount of routing messages is significantly lowered in our system, as a result of the separation of structural and dynamic information. Because the majority of them are not caused by structural changes on the Internet. Rather, they are the result of traffic fluctuation, routing policy changes and traffic engineering attempts [9, 22, 11]. These messages will not appear in our system.

Finally, the requirements upon transit routers are substantially reduced, since their resource requirement is no longer tied to the size of the Internet. Other than a small per-edge cost for storing structural information, transit router's resource requirement is independent of the number of distinct address prefixes and autonomous systems. In BGP, every router has to perform route computation, and possibly update its forwarding state, in response to routing updates. Our system requires this calculation only at the access routers, which can be incrementally updated, and have less demanding performance requirements. This keeps the core of the network independent from the costs of Internet growth such as more address prefixes, and from the requirement to provide aggregatable addressing by matching IP address structure to topology.

In order to understand the amount of information that an access router has to deal with, we processed the April 1st, 2002 BGP routing table dumps from the Route View Project [15]. We found that there are 13041 autonomous systems, 124330 prefixes, and 184557 links. Among the links, 57702 are inter-autonomous links. While these numbers might not accurately describe the topology of the Internet, they do give us a sense of the amount of computation and storage we are dealing with. Suppose we need 200 bytes to store information about a link, and 100 bytes for each autonomous system and prefixes, the total memory requirement is less than 50 megabytes.

The routing information we put into each packet is also minimal. Our analysis of the BGP routing tables showed that there are less than 5 autonomous systems between most prefixes. Therefore less than 24 bytes are needed for most packets. In comparison, IPv6 needs 32 additional bytes in the IP header.

4. APPLICATIONS

4.1 Defend Against Denial of Service Attacks

Denial of service attacks are becoming a serious threat to the availability of Internet connected networks and services. Although the attacks might not specifically target the routing infrastructure, we believe that the inter-domain routing system can be leveraged to help edge networks defend against DoS attacks.

The current defense against Denial of Service attacks involves manually contacting the upstream providers to install filters in their border routers to stop the flow of attack packets. This process is slow because sometimes several ISPs have to be contacted, and subject to social engineering (hackers impersonating edge network administrators). In this section, we propose an automated mechanism to install filters.

Once a denial of service attack has been detected, it is often feasible to recognize a pattern that differentiates the attack from normal traffic. The victim can then originate an update of the routing information for the link between his network and the upstream provider. The negative rules of this link would include the pattern that matches these DoS packets. A responsible upstream provider would then incorporate the rule into the access control list of its border routers, thus stopping the DoS packets from reaching the victim. Information about the link can be propagated further, and other autonomous systems on the way from the victim to the attacker can incorporate the rules to stop the attack traffic even earlier on.

4.2 Virtual Links with Zero Failover Time

Many Internet based applications require a reliable link between two remote locations. For example, if a surgery is performed at San Francisco, while the experts are located in New York City, then it is unacceptable if there is an connection outage. Although the current inter-domain routing system is designed to handle link failures, it can not deliver convergence time in the sub-second time scale [11]. Labovitz et al. [13] has shown that even with multi-homing, BGP's convergence time after a link failure is 3 minutes on average. And this number is increasing as the Internet grows bigger. Such a long period of no connectivity is unacceptable for mission critical real time applications. Therefore, the traditional solution to providing a link with fast fail-over property is by using leased lines. However, the cost of doing so is much higher than using a pure IP based approach.

In this section, we propose a mechanism to implement highly available virtual links with zero fail-over time based on the routing system we proposed. Obviously, if a site is connected to the Internet through only one provider, then it will lose connectivity with all other sites when that provider goes down. Therefore, a site wishing to have a reliable virtual link is best served with multiple upstream providers.

Suppose networks A and B want to establish such a virtual link, each of them should have a reliable gateway. In most cases, this would be the access router. We further assume that both A and B are multi-homed. Our research show that for most prefixes that are multi-homed, there exist two vertex disjoint routes between them, given the current Internet topology. Therefore, it can be assumed that the primary route and backup route are vertex disjoint. We further assume that the failure of these two routes are independent.

The gateways at A and B duplicates every packet that is sent to each other. Each packet is assigned a unique sequence number for duplicate detection. Or if there is an underlying IP level security protocol such as IPsec that can detect and eliminate duplicate packets, such a sequence number is not needed. The gateways are responsible for eliminating the duplicate packets.

If one of the routes fails, the virtual link continues to function. Since we assume independent failures of the two routes, the probability of a virtual link failure between A and B is π^2, where π is the probability of one route failing. After one route fails, the access router should perform a route computation, and select another route as the backup route. Therefore the period in which there is only one usable route from A to B is minimal.

This seems to be a waste of bandwidth. But bandwidth is virtually free and the network should be engineered such that if one link goes down, the surviving one can still fulfill the communication needs between A and B.

The assumption of failure independence of vertex disjoint routes can be invalid since there might be hidden dependencies among autonomous systems. For example, two autonomous systems might be operated by the same company. If the company files Chapter 7 bankruptcy, both AS's will go down. It is also possible that disjoint routes go through the same physical wire, or the same wiring closet. Our future research will try to address this problem.

5. RELATED WORK

Previous work on "byzantine robustness" [16] addressed many of the same security problems we discuss in this paper. There are three major differences between Perlman's proposal and ours. First we realized that the transit routers do not need to know about the network topology. This optimization makes the transit routers' resource requirement almost independent of the growth of the Internet. Second, we made the observation that most network traffic is TCP. Instead of inventing a separate network layer ACK, as proposed by Perlman, we used TCP SYN and SYN ACK packets as our end-to-end performance measurement. Finally, we are concerned about the scalability the routing system. Therefore all expensive computation are moved out of the critical path of the algorithm. In fact most computation can be done in parallel to packet forwarding.

Secure BGP [12] tries to address the routing security issues by adding signatures to route advertisements, so that reachability information can not be forged. However, BGP already suffers from slow response to network changes. Burdening it with further validation of peer-provided routing information threatens to slow it down further, if meaningful validation is in fact possible. Even if it can be done in a scalable manner, the existence of an authenticated route does not ensure that the route is actually operational. We believe the security problem in routing is a robustness problem, and can not be solved by authentication alone.

The Resilient Overlay Network [4] is a project that creates an overlay network that can get around routing failures. We believe overlay networks are not the final solution for reliable packet forwarding. The reason is simple. Overlay networks only increases the probability that the communication does not fail when there are only isolated routing failures in the network. No overlay network is going to function when the underlying routing infrastructure complete fails, for example, as the result of a black hole attack.

There are several companies, such as RouteScience [2], SockEye Networks [3] and netVmg [1], which are building products that can be used in an edge network to dynamically choose better paths, based on the end-to-end performance measurement. These products provide possible performance enhancement, and provide resilience to the failure of upstream providers. However, because they are edge-only solutions, they do not shield the customers from a large scale failure in the network. Also because they do not control the paths a packet travel after the packet enters the upstream ISP's network, they can not provide true redundancy.

6. CONCLUDING REMARKS

There are three key components of the routing system we propose. First, our system separates determination of dynamic performance information from structural information propagation. The former is determined locally with feedback based routing instead of receiving both from supposedly trusted peers, as in BGP.

Second, our system reduces routing in the backbone to purely structural information propagation. By removing route computation from the transit routers, and by removing route computation from the critical path of access routers, our system scales better than BGP.

Third, based on the structural information, an access router maintains more than one route to every destination. These redundant routes decrease the response time after a router or link failure.

The Internet has changed a lot from its early days. The trust among network operators no longer exists; the size of the Internet is growing exponentially; and many people have realized BGP is neither predictable nor stable. We believe a routing algorithm that depends on everyone behaving to function will no longer work well in such an environment. We also believe the solution to the scalability problem lies in giving the control of route selection to the edge networks and freeing the core routers from maintaining a complete model of the Internet. Our results to date suggest that feedback-based routing is a promising direction to consider. We plan to explore ways to incorporate this mechanism into the Internet as it evolves.

7. ACKNOWLEDGMENTS

The paper greatly benefited from the comments of the anonymous reviewers. Shelley Zhuang of UC Berkeley reviewed an early draft of the paper and provided much valuable feedback.

8. REFERENCES

[1] netvmg. http://www.netvmg.com.

[2] Routescience. http://www.routescience.com.

[3] Sockeye networks. http://www.sockeye.com.

[4] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris. Resilient overlay networks. In *Proceedings of ACM SOSP*, October 2001.

[5] R. Barrett, S. Haar, and R. Whitestone. Routing snafu causes internet outage. *Interactive Week*, April 25 1997.

[6] batz. Security issues affecting internet transit points and backbone providers. Blackhat Briefings, 1999.

[7] D. R. Cheriton and M. Gritter. Triad: A new next-generation internet architecture. http://www.dsg.stanford.edu/triad/triad.ps.gz, July, 2000.

[8] A. Collins. The detour framework for packet rerouting. http://www.cs.washington.edu/research/networking/detour/, November 1998.

[9] J. Cowie, A. Ogielski, B. Premore, and Y. Yuan. Global routing instabilities during code red ii and nimda worm propagation. http://www.renesys.com/projects/bgp_instability, September 2001.

[10] K. Houle and G. Weaver. Trends in denial of service attack technology. http://www.cert.org/archive/pdf/DoS_trends.pdf, October 2001.

[11] G. Huston. Commentary on inter-domain routing in the internet. RFC 3221, December 2001.

[12] S. Kent, C. Lynn, and K. Seo. Secure border gateway protocol, April 2000.

[13] C. Labovitz, A. Ahuja, A. Bose, and F. Jahanian. Delayed internet routing convergence. In *Proceedings of ACM SIGCOMM*, pages 175–187, 2000.

[14] C. D. Marsan. Faster 'net growth rate raises fears about routers. *Network World Fusion*, April 2 2001.

[15] D. Meyer. University of oregon route views project. http://antc.uoregon.edu/route-views/.

[16] R. Perlman. Network layer protocols with byzantine robustness. Technical Report 429, MIT Laboratory for Computer Science, 1988.

[17] J. Postel. Transmission control protocol, September 1981.

[18] Y. Rekhter and T. Li. A border gateway protocol (bgp-4), March 1995.

[19] S. Savage, T. Anderson, A. Aggarwal, D. Becker, N. Cardwell, A. Collins, E. Hoffman, J. Snell, A. Vahdat, G. Voelker, and J. Zahorjan. Detour: a case for informed internet routing and transport. *IEEE Micro*, 19(1):50–59, January 1999.

[20] S. Savage, N. Cardwell, and T. Anderson. The case for informed transport protocols. In *Proceedings of HotOS*, March 1999.

[21] S. Savage, A. Collins, E. Hoffman, J. Snell, and T. Anderson. The end-to-end effects of internet path selection. In *Proceedings of SIGCOMM*, pages 289–299, September 1999.

[22] A. Shaikh, L. Kalampoukas, R. Dube, and A. Varma. Routing stability in congested networks: Experimentation and analysis. In *Proceedings of SIGCOMM*, pages 163–174, 2000.

[23] K. Varadhan, R. Govindan, and D. Estrin. Persistent route oscillations in inter-domain routing. *Computer Networks*, 32(1):1–16, 2000.

Secure Traceroute to Detect Faulty or Malicious Routing

Venkata N. Padmanabhan* and Daniel R. Simon†
Microsoft Research

ABSTRACT

Network routing is vulnerable to disruptions caused by malfunctioning or malicious routers that draw traffic towards themselves but fail to correctly forward the traffic. The existing approach to addressing this problem is to secure the routing protocol by having it validate routing updates, i.e., verify their authenticity, accuracy, and/or consistency. In this paper, we argue that it is also important to ensure the robustness of packet forwarding itself. To this end, we propose a different approach, the central idea of which is a *secure traceroute* protocol that enables end hosts or routers to detect and locate the source of (arbitrarily severe) routing misbehaviors, so that appropriate action can be taken.

1. INTRODUCTION

Although a great deal of attention has been paid to securing network communication, the security of network routing has been neglected by comparison. For example, the predominant inter-domain routing protocol in the Internet, BGP, includes no mechanism for verifying either the authenticity (correct origin) or the accuracy of the routing information it distributes. As a result, traffic can be severely disrupted by routers refusing to serve their advertised routes, announcing nonexistent routes, or simply failing to withdraw failed routes, as a result of either malfunction or malice. A particularly problematic case is that of sophisticated malicious routers (e.g., routers that have been compromised) that attempt to hide or disguise their misbehavior.

Two approaches have been suggested to solving this problem. One, typified by Secure BGP (S-BGP) [7], requires routing information to be authenticated—say, by digital signature—so that routers advertising false routing information can be held accountable when detected. However, the overhead of digital signature is large and can be prohibitive—for example, when bringing a failed Internet router back on line, at which time all BGP routing advertisements for that router must be digitally signed at once. Moreover, authentication of routing information does little to help detect or diagnose faulty routing information emanating from a router (for example, a compromised one); it only ensures reliable identification of the information's origin (for after-the-fact, out-of-band blame assignment) should the misbehavior somehow be detected.

The other approach is to maintain a centralized registry of "plausibility" information about routing advertisements (akin to the Routing Arbiter [15]), so that blatantly invalid routing information can be discounted when received. This approach can prevent the most egregious routing problems arising from router misconfigurations, but it is still vulnerable to a wide range of both inadvertent and malicious false advertisements for routes that a particular router may be "entitled" to advertise, but cannot (or simply will not) in fact serve. Thus, beyond ensuring the security of the routing protocol, it is also important to deal directly with packet forwarding misbehavior.

To this end, we propose a third approach, in which routers, assisted by end hosts, adaptively detect poorly performing routes that indicate routing problems, and use a *secure traceroute* protocol to attempt to identify an offending router. Once the offending router has been identified, it can be routed around, the detection can be publicized, or out-of-band action can be taken to attempt to resolve the problem.

The key idea behind secure traceroute is to securely trace the path of existing traffic, rather than that of special traceroute packets, to prevent adversaries from misleading the tracer by treating traceroute and normal traffic differently. Secure traceroute responses are also authenticated, to verify their origin and prevent spoofing or tampering.

Our approach is orthogonal to and can complement schemes such as S-BGP that secure the routing protocol itself. Secure traceroute can be incrementally deployed in the Internet; moreover, it is a general technique that can be used to diagnose routing problems in other settings such as single-ISP networks, ad hoc networks, and peer-to-peer networks. In the latter two contexts, especially, where routers are generally uncontrolled and untrusted, methods for detecting and reacting to malicious behavior are particularly useful.

2. ASSUMPTIONS AND THREAT MODEL

We assume a network in which nodes are individually globally addressable (which is often, though not always, true in the Internet), and in which each (non-faulty) node, on receiving a packet with an origin and destination address, chooses a single next node to which to forward it. Our goal is to deal with fairly limited, localized routing problems in this setting; if the routing disruption is too widespread, involving too many nodes, then it can interfere with any attempt to investigate it. A typical example of our target problem might be a single faulty router somewhere in the network (although our approach certainly does not assume that there is only one faulty router). Note that it is not necessary to distinguish between a merely faulty (e.g., mis-

*http://www.research.microsoft.com/~padmanab/

†http://research.microsoft.com/crypto/dansimon/me.htm

configured) router and a malicious one; in practice, after all, sufficiently erroneous behavior can be indistinguishable from malice. (An adversary may, for instance, attempt to disguise its malicious behavior as mere error in order to avoid later retribution.) Hence we assume faulty routers to be capable of arbitrary Byzantine behavior.

3. SECURE TRACEROUTE

The normal traceroute protocol involves the sender simply sending packets with increasing TTL (time to live) values, and waiting for an ICMP time-exceeded response from the node that receives the packet when the TTL expires. Normally, this protocol easily generates a sequence of addresses of nodes on the path to the destination, or at least up to the point where packets are being lost on a faulty link. However, a malicious router could intercept and alter traceroute traffic to give an arbitrary misleading impression—say, by letting only traceroute packets through, or by constructing fake responses to them so as to give the impression that they are getting through and demonstrating a fully functioning path (or a path with a problem elsewhere). Secure traceroute is intended to prevent this type of disruption of the traceroute process by verifying the origin of responses, and preventing traceroute packets from being handled differently from ordinary traffic.

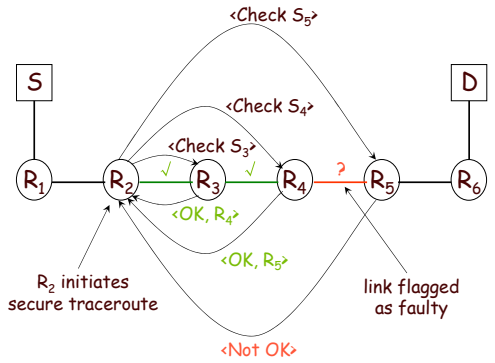


Figure 1: An illustration of secure traceroute in operation. R_2 initiates secure traceroute and concludes eventually that the link between R_4 and R_5 is faulty. S_i denotes the "signature" of the packets that router R_i is asked to treat as traceroute packets.

As in normal traceroute, secure traceroute proceeds hop by hop, with each node on the path being asked to respond to traceroute traffic (Figure 1). However, there are several fundamental differences between them, including traceroute packet contents and the responses they elicit from routers. We outline the specifics of secure traceroute below:

1. In addition to their own address (which is included implicitly in the response packet), hosts responding to secure traceroute packets provide a next-hop address for the packet. Thus the node performing the traceroute always knows the (expected) next node on the path.

2. Prior to sending the traceroute packets, the tracing node establishes a shared key for encrypted, authenticated communication to and from the expected next node. (How this is done is described in Section 4.) Using this key, identifying information, which specifies the "signature" of the packets to be treated as traceroute packets, is securely passed from the tracing node to the expected next node. This information could include the source and destination addresses (or address ranges) of the selected packets, and the values (or constraints on the values) of certain fields on the packets. For example, it could be specified that all packets between certain source and destination addresses for which a certain field contains a particular value modulo a specific prime number should be examined. Alternatively, a random value ("nonce") could be inserted into some field in the packet by the tracing node, and certain such values would be designated as signifying a traceroute packet. Thus, traceroute packets will look indistinguishable from ordinary traffic to intermediate nodes, and will therefore constitute a representative sample of traffic between the specified source and destination ranges.

3. In replying to a packet identified as a secure traceroute packet, a node sends some agreed-upon identifying marker for the packet back to the tracing node, to "prove" that the packet has been received. Alternatively, some accumulated value based on multiple received packets may be returned. For example, if the tracing host can place data on the traceroute packets, then it can use a threshold secret-sharing scheme [12] to send a secret that can only be extracted when some minimum number of shares (and thus packets) has been received. (In a threshold secret-sharing scheme, a secret is "divided up" into a number of shares such that any subset of the shares of a certain "threshold" size suffices to reconstruct the secret, but any subset smaller than the threshold reveals no information about the secret. Threshold secret-sharing is an efficient operation, comparable in cost to symmetric-key cryptography.) By inserting one share per traceroute packet, and measuring how many shares must be sent before a reply is received, the tracing host can estimate what fraction of traceroute packets are getting through.

4. In addition to packet-identifying information, the secure traceroute reply contains a strongly secure Message Authentication Code (MAC) that ensures its authentic origin. The MAC is based on the original key shared between the tracing node and expected next node, and covers the entire response, including the address of the node to which the expected next node forwarded the traceroute packet, based on its destination. This new node becomes the "new" expected next node for the next step of the traceroute.

This iterative process produces, as does a normal traceroute, one of two possible results: either a complete route is determined, or a faulty link is found, such that one end of the link claims to be sending packets through the link, but the other end claims not to be receiving them (or simply doesn't respond to the traceroute). However, the identification of this link is much more reliable than for ordinary traceroute, since return packets are (with greater or lesser certainty, as described below) established to be coming from the correct node, and the use of normal packets

in place of identifiable traceroute packets ensures that the route (whether functioning or faulty) accurately represents the route taken by normal packets.

Because secure traceroute is more expensive than the normal variety, it may make sense to limit its use whenever possible. One way to do so is to initiate it starting at a point on the route just before where the preceding normal traceroute process indicates that the problem appears; thus, if that insecure traceroute turns out to be accurate, then the secure traceroute will demonstrate said accuracy at a relatively small cost. For example, in the (probably very common) case that the normal traceroute indicates that the path functions well until the destination host (and hence, that the problem is probably with the destination host rather than the route), then a secure traceroute can be initiated starting close to the destination host (if not at the destination host itself) on the presumed route. If the secure traceroute results in correct responses where it is initiated, then it can be assumed that the "skipped-over" portion of the route generated by the standard traceroute is accurate (or, in any event, that the "skipped-over" portion is not causing the problem).

4. AUTHENTICATING SECURE TRACEROUTE

The above protocol assumes that a secure (that is, encrypted, authenticated) key exchange can be performed between the tracing host and the expected next host. Ideally, a public-key infrastructure (PKI) would be available to allow such key exchange to occur using standard protocols (e.g., IPSEC [6]). Such a PKI for infrastructure operators (as opposed to end users) is not unthinkable, as the widely deployed "SSL" PKI for Web servers demonstrates; however, it cannot necessarily be assumed. In its absence, various ad hoc mechanisms can be used. For example, PGP-style "Web of trust" techniques [16, 2] can be used to propagate routers' public keys from node to (trusting) node; using Web- or email-based protocols, nodes can distribute public keys of nodes they have established confidence in the sources of, and receive such keys in turn from others whose judgments they trust. Similarly, certain widely trusted hosts could voluntarily act as "key servers", collecting and verifying public keys of nodes and offering them for authenticated download to nodes that trust the key server. Finally, the redundancy of the network can be used to attempt to determine the valid public keys of other hosts at authentication time. By requesting the public key of a host multiple times, via multiple paths—possibly with the help of a set of widely distributed, trusted routing intermediaries (an overlay network of sorts)—a tracing host can increase the likelihood that the public key being returned has not been tampered with en route by a malicious host. Once the expected next host's public key has been obtained, a (one-way) authenticated key exchange is easy, and the secure traceroute can proceed.

5. USING SECURE TRACEROUTE

We propose a general five-stage process for using secure traceroute to detect, identify and mitigate routing problems, consisting of the following steps:

1. **Complaint:** If an end host detects an end-to-end performance problem to a particular destination, it sets a "complaint bit" in its subsequent traffic to that destination, indicating that a problem is occurring. Complaints emanating from sources found to be generating false complaints are discounted or ignored. (Obviously, this mechanism is vulnerable to source address spoofing; we assume some other solution to the spoofing problem—ingress filtering, traceback, authentication—has been implemented.)

2. **Complaint Evaluation:** If a router's "complaint level" get high enough—say, if most or all traffic destined for a particular set of addresses complains—then the router may choose to investigate it. Note that a sufficiently severe and sustained congestion problem is indistinguishable from a routing problem, and can be treated as such.

   In principle, of course, an end host could investigate its own complaints. However, it would be best for a router that is as far downstream as possible (i.e., closest to the problem) to investigate, since its resolution of the problem (say, by rerouting traffic) is likely to benefit the maximum number of sources. We propose an adaptive approach where each router waits for a random interval based how far downstream it thinks it is (say, based on the TTL value of packets it receives) before initiating the investigation.

3. **Normal Traceroute:** The first step in the investigation is to perform a traceroute to attempt to identify the offending node or link in the downstream path. The (possibly partial) path returned by the traceroute may be completely misleading; a malicious router along the route could easily intercept and respond to traceroute packets so as to suggest an arbitrary failed or successful subsequent path. However, the path returned by the traceroute may be useful in locating a faulty node in the presumably more common cases of router misconfiguration or failure, or end-host failure. So this information is used as the starting point for the secure traceroute step described next.

4. **Secure Traceroute:** To verify the results of the traceroute, the investigating router then performs a secure traceroute, described in Section 3. Since secure traceroute is a more "heavyweight" protocol, it is initially only performed to confirm the results of the standard traceroute. That is, if the normal traceroute was successful, then secure traceroute is used to check that packets are in fact getting to the destination node; if the normal traceroute terminated prematurely, then a secure traceroute is initiated starting with the successful node closest to the point of failure. Thus, secure traceroute is a relatively cheap procedure if the normal traceroute is in fact valid. However, if this procedure reveals that the normal traceroute was misleading, then secure traceroute is performed starting with the very next downstream hop, all the way to the destination.

   If secure traceroute is supported only by a subset of the routers in the network, we proceed as follows: for each router on the path reported by the normal traceroute, a secure traceroute step is attempted. If it succeeds for a particular router, we deduce that the path up to that router is good; otherwise, we simply ignore that router. The subset of secure traceroute-capable routers thus

divides the end-to-end path into a sequence of subsegments. Eventually, the secure traceroute will have identified a subsegment (possibly the last one) that contains a faulty link. Hence, even when only incrementally deployed, secure traceroute still provides partial information about the location of faulty links.

Note that the path information used here is obtained from a normal traceroute and hence is not authenticated; it need not be, since is verified by the secure traceroute. A bad link found by the secure traceroute is in this case either a faulty link on the true end-to-end path, or a point at which the normal traceroute was led astray.

5. **Problem Correction:** A Secure traceroute can at best identify a faulty link on a route—a link that one node claims to be sending packets through, while the node on the other end claims not to be receiving them. The router that determines such a faulty link can try to route around it; notify downstream routers of the problem, expecting them to make the appropriate routing adjustments; or even pursue the matter through human intervention (such as contacting the administrator of the suspected offending router). We will not examine possible correction strategies in detail here.

6. ROUTING ASYMMETRY

Internet routing is, in general, asymmetric. The path from node A to node B may be different from that from B to A. This asymmetry can create two problems. First, an end host cannot be sure whether its inability to communicate with a peer host is the result of a network problem in one direction, the opposite direction, or both. Second, the same ambiguity can also affect a node that is performing a secure traceroute. We discuss both these issues in turn.

6.1 Impact on the End-host Complaint Process

Consider an end host A that is trying to communicate with end-host B. If A does not receive any response (e.g., TCP ACKs) to the packets it has sent to B, A cannot be sure whether there is a network problem in the A→B direction, in the B→A direction, or in both directions. The question then becomes when end host A should start "complaining" by setting the appropriate bit in its packets.

If A receives a steady stream of packets from B but none that indicates that A's packets have been received by B, then A can be quite sure that the problem is in the A→B direction. Certain applications, such as conferencing, may generate a steady, bidirectional flow of traffic that enables such disambiguation. Certain protocols as well, such as TCP, generate sustained traffic in at least one direction (e.g., TCP retransmissions). In cases where no such traffic is normally generated, we propose that peer hosts that are in the midst of communicating with each other transmit "keep-alive" packets (at a low frequency) during times when they do not have regular packets to send. Receipt of traffic (with little or no loss) from host B would indicate to A that the problem is in the A→B direction. A can then start setting the "complaint bit" in its packet to B. On the other hand, if the problem is only in the B→A direction, then B would hear A's traffic, deduce that the problem is in the B→A direction, and initiate the complaint process.

There are, however, two problem cases: (a) there may be a failure in both the A→B and B→A directions, or (b) the network failure may precede any communication between A and B, preventing the hosts from exchanging any traffic at all. In both cases, since neither A nor B may receive traffic from its peer, neither would be in a position to determine definitively the direction of failure. In such cases, the deadlock can be broken by having the host(s) initiate the complaint process anyway, after having waited for a random extra duration to give its peer the opportunity to initiate the complaint process and possibly resolve the problem.

6.2 Impact on Secure Traceroute

Asymmetry in network routing can make it difficult for an investigating router, R, to check whether a downstream router, D, is in fact receiving packets. First, even if D is receiving all packets forwarded by R (and hence there is no network problem in the R→D direction), the two routers may not be able to establish a secure communication channel simply because of a network problem in the D→R direction. Second, even if a secure channel has been established, R may not receive the appropriate response from D due to a (new) network problem in the D→R direction. Thus if R's secure traceroute attempt does not elicit the appropriate response from D, R cannot be sure whether there is a problem in the R→D direction or in the D→R direction.

The solution we suggest is as follows: The investigating router, R, first initiates the secure traceroute process as described earlier. If it does not receive the appropriate response from a certain downstream router, D, router R repeats the secure traceroute process with one difference: it includes the reverse route that D could use to communicate back to R using a mechanism such as IP source routing. The reverse route consists of the sequence of intermediate routers along the path from R to D in reverse order. (Note that due to routing asymmetry, the reverse route may not be the same as the route from D to R.) The underlying assumption is that if in fact there is no network problem in the R→D direction, it is quite likely (modulo the presence of unidirectional links) that D will be able to communicate back to R via the reverse route.

In the worst case, the inability of D to communicate back to R would just mean that R would incorrectly deduce that the problem is at or around D and would proceed with unnecessary rerouting or other corrective action, an undesirable but hardly disastrous outcome.

7. ATTACKS

There are a number of potential attacks against the approach presented here; we outline a few below, along with some potential countermeasures.

1. Because the most frequent cause of failed connections will be unresponsive end hosts—a problem which cannot be fixed by routing adjustments—a malicious router can avoid detection via secure traceroute by simulating a "dead" end host, simply by advertising a (non-responsive) direct link to the targeted end host. If the claimed last-hop router is very far away from the targeted end host, then routing metrics (such as hop counts) can suggest a problem; also, an attempt to find an alternate route should yield one that avoids the offending router. On the other hand, if the malicious

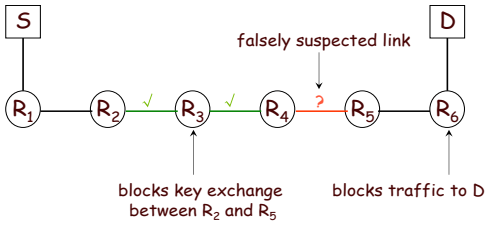


Figure 2: Misbehaving routers R_3 and R_6 work in conjunction to deceive R_2 into concluding that the link between R_4 and R_5 is faulty.

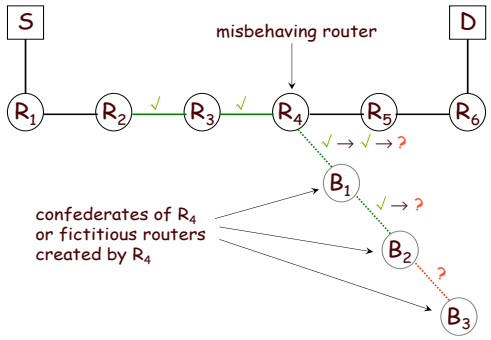


Figure 3: Misbehaving router R_4 leads the secure traceroute initiated by R_2 astray, down a path of bogus routers. However, R_2 can still identify and eliminate the bogus links through repeated application of secure traceroute.

router is very close to the targeted end host, then these measures are likely to be less successful; of course, in the worst case, where the misbehaving router is really the (sole) last-hop router, then it will obviously be impossible to distinguish its "blackholing" activity from a truly dead end host.

2. A malicious router may adjust its disruptive behavior so as to avoid detection. For example, it may confine its attacks to periods of time where it does not detect any secure traceroute initiation attempts (i.e., key exchange packets from upstream routers). A simple solution is to give occasional indications of traceroute activity (such as sending key exchange packets—whether real or bogus) whenever there is any hint of a problem. Since the malicious router cannot distinguish real secure traceroute attempts from fictitious ones (beyond detecting the presence or absence of key exchanges), the presence of such simulations should ensure that misbehavior occurs either at such times when it can be detected by secure traceroute, or else not at all.

   Alternatively, the malicious router may attempt to interfere with the secure traceroute by selectively blackholing the packets used in the key exchange phase, so as to give the impression that a router further downstream is not accepting key exchanges (and hence either malfunctioning or malicious). This attack cannot be used by a single misbehaving router to frame a router further downstream: if the misbehavior affects normal traffic, then the secure traceroute will correctly detect a misbehaving link when the (honest) router immediately downstream of the adversary on the path reports the anomalous traffic pattern. However, two misbehaving routers could collude to frame a router between them on a path; the downstream confederate disrupts traffic, while the upstream one disrupts key exchanges to the victim router so as to implicate it (Figure 2). A simple countermeasure to this attack (if multiple colluding routers are deemed a threat, and if redundant routes are not being used to effect the key exchange) is to use "onion routing"-style encryption of key exchange messages [14]. Since each step of the traceroute involves a key exchange, the key exchange traffic can be encrypted hop by hop, so that each router along the route does not know the final destination of the message (and therefore cannot consistently frame a single router).

3. If an attacker can create numerous fictitious nodes in the network, then both identifying a bad link and attempting to route around it can become much more difficult and time-consuming. A single attacker on a path, for instance, could divert a secure traceroute by returning a bogus next-hop router (Figure 3). This may still remain undiscovered if the bogus router is either a confederate of the attacker or the attacker itself under the garb of a different address (i.e., a "dummy" node). In fact, the attacker could lead the secure traceroute into a thicket of bogus routers before it peters out. However, the secure traceroute will eventually identify a bad link, at least one end of which is the attacker or its confederate. Thus, this link deserves to be avoided. There may still be other misbehaving nodes in the path, but persistent application of a succession of secure traceroutes can eliminate them, one by one, from the path until they have all been purged. Hence, if adding confederates or dummy nodes to the network is sufficiently costly, or if owners or administrators of misbehaving nodes are investigated and penalized sufficiently harshly if found to be guilty, then the application of secure traceroute would still help identify misbehaving nodes, so that they can be eliminated and/or punished.

4. Finally, the need for computationally expensive key exchanges creates an opportunity for powerful denial-of-service attacks using bogus key exchange messages that require significant computational resources to detect and discard. A simple solution is just to throttle the number of secure traceroute key exchanges honored to keep it below an acceptable threshold. This raises the possibility of combining malicious routing with denial-of-service attacks to foil secure traceroute attempts. One possible solution to this problem is to respond to a key exchange message with a "client puzzle" (as in [1, 5]). Such puzzles are easy for the responding router to generate and check, without maintaining state; the requesting router (or the attacker), in order to have its traceroute request attended to, would have to solve the puzzle—which would require at least the same order of magnitude of computation as the responding router has to perform in order to handle the secure traceroute—and return the solution along with a resend of its key exchange message.

Of course, the attacker could always simply muster the computational resources to mount the attack (say, by harnessing thousands of hacked "zombie" computers to the task). But anyone with that level of resources is likely able to mount a more conventional denial-of-service attack against their intended targets in any event—probably without ever needing to subvert the routing system.

8. RELATED WORK

There have been several pieces of work on making Internet routing protocols robust against attacks by malicious entities. Perlman [9, 10] presents an approach for "sabotage-proof" routing. The key idea is to use "robust flooding" to distribute link-state packets (LSPs) and the public keys of all nodes throughout the network. Robust data routing is then accomplished by having end hosts construct digitally signed source routes using the link-state information they have gathered. There are a couple of issues that limit the practicality of this approach. First, flooding LSP information on a global scale is likely to be infeasible; indeed this is the rationale for using BGP, a path-vector protocol, for inter-domain routing. Second, allowing each end host to do source routing severely weakens the ability of ISPs to engineer traffic in their networks.

Other proposals [7, 13] have considered less disruptive approaches to securing the routing protocol. In particular, Secure BGP (S-BGP) [7] proposes using public key infrastructures (PKIs) and IPSec to enable a BGP speaker to validate the authenticity and data integrity of BGP UPDATE messages that it receives and to verify the identity and authorization of the senders. As noted in Section 1, S-BGP could impose an unacceptable overhead, and more importantly does not offer protection against a misconfigured or failed router that is authorized to advertise routes for an address prefix but fails to deliver packets anyway.

There has been considerable interest recently in securing routing on mobile ad hoc networks. In [8], a "watchdog" technique is proposed to enable a node to check that a neighboring node did in fact forward a packet onward without tampering with it. This technique makes the strong assumption that nodes can hear the onward transmissions of their neighbors, something that may not be true even in wireless ad hoc networks (for instance, due to directional antennae). SEAD [3] focusses on a lightweight scheme to enable nodes to authenticate routing updates from other nodes for the specific case of a distance-vector routing protocol. As with S-BGP, authentication does not solve the problem of a faulty node that fails to forward packets. [4] proposes a self-organized PKI suitable for mobile ad hoc networks. In the absence of a centralized PKI, we could use a similar approach to support secure traceroute.

Finally, recent work on IP traceback (e.g., [11]) tries to solve a different problem related to the one addressed by secure traceroute. The goal of IP traceback is to determine which routers a specified subset of traffic (typically the "attack traffic" during a DDoS attack) traverses. In IP traceback, the information provided by routers is trusted. Secure traceroute, on the other hand, is used to determine whether traffic did in fact traverse a router.

9. CONCLUSION

In this paper, we have argued that robust routing requires not only a secure routing protocol but also well-behaved packet forwarding. To this end, we have proposed an approach to robust routing in which routers, assisted by end hosts, adaptively detect poorly performing routes that appear suspicious, and use a *secure traceroute* protocol to attempt to detect an offending router. This approach complements efforts that focus on securing the routing protocol itself. We view secure traceroute as a general technique with wide applicability, and are presently investigating it in the context of multi-hop wireless networks.

Acknowledgements

We would like to thank John Douceur, Jitu Padhye, Lili Qiu, Stefan Saroiu, and the anonymous HotNets reviewers for their useful feedback on an earlier version of this paper.

10. REFERENCES

[1] C. Dwork and M. Naor. "Pricing Via Processing or Combatting Junk Mail", In Ernest F. Brickell, editor, *Advances in Cryptology – CRYPTO '92*, volume 740 of *Lecture Notes in Computer Science*, pages 139-147, 16-20 August 1992. Springer-Verlag, 1993.

[2] C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas and T. Ylonen. "SPKI Certificate Theory", RFC 2693, September 1999.

[3] Y. C. Hu, D. B. Johnson, and A. Perrig. "SEAD: Secure Efficient Distance Vector Routing for Mobile Wireless Ad Hoc Networks", *IEEE WMCSA 2002*, June 2002.

[4] J. P. Hubaux, L. Buttyan, and S. Capkun. "The Quest for Security in Mobile Ad Hoc Networks", *ACM MobiHoc 2001*, October 2001.

[5] A. Juels and J. Brainard. "Client Puzzles: A Cryptographic Defense Against Connection Depletion Attacks", *NDSS '99 (Networks and Distributed Security Systems)*, February 1999.

[6] S. Kent and R. Atkinson. "Security Architecture for the Internet Protocol", RFC 2401, November 1998

[7] S. Kent, C. Lynn, and K. Seo. "Secure Border Gateway Protocol (Secure-BGP)", *IEEE Journal on Selected Areas in Communications*, Vol. 18, No. 4, April 2000.

[8] S. Marti, T. J. Giuli, K. Lai, and M. Baker. "Mitigating Routing Misbehavior in Mobile Ad Hoc Networks", *ACM Mobicom 2000*, August 2000.

[9] R. Perlman. "Network Layer Protocols with Byzantine Robustness", Ph.D. Thesis, Department of Electrical Engineering and Computer Science, MIT, August 1988.

[10] R. Perlman, "Interconnections: Bridges, Routers, Switches, and Internetworking Protocols", *Addison-Wesley Professional Computing Series*, Second edition, 1999.

[11] S. Savage, D. Wetherall, A. Karlin, and T. Anderson. "Practical Network Support for IP Traceback", *ACM SIGCOMM 2000*, August 2000.

[12] A. Shamir. "How to share a secret." Communications of the ACM, 22:612–613, 1979.

[13] B. R. Smith and J. J. Garcia-Luna-Aceves. "Securing the Border Gateway Routing Protocol", *Global Internet 1996*, November 1996.

[14] P.F. Syverson, G. Tsudik, M. G. Reed, and C. E. Landwehr. "Towards an Analysis of Onion Routing Security". In H. Federrath, editor, *Designing Privacy Enhancing Technologies*, vol. 2009 of LNCS, pp. 96–114. Springer-Verlag, 2001.

[15] The Routing Arbiter Project, *http://www.isi.edu/div7/ra/*

[16] P. Zimmermann, "The Official PGP User's Guide", MIT Press, 1995.

Performance of Multihop Wireless Networks: Shortest Path is Not Enough

Douglas S. J. De Couto Daniel Aguayo Benjamin A. Chambers Robert Morris
M.I.T. Laboratory for Computer Science
{decouto, aguayo, bac, rtm}@lcs.mit.edu

Abstract

Existing wireless ad hoc routing protocols typically find routes with the minimum hop-count. This paper presents experimental evidence from two wireless test-beds which shows that there are usually multiple minimum hop-count paths, many of which have poor throughput. As a result, minimum-hop-count routing often chooses routes that have significantly less capacity than the best paths that exist in the network. Much of the reason for this is that many of the radio links between nodes have loss rates low enough that the routing protocol is willing to use them, but high enough that much of the capacity is consumed by retransmissions. These observations suggest that more attention be paid to link quality when choosing ad hoc routes; the paper presents measured link characteristics likely to be useful in devising a better path quality metric.

1. Introduction

Ad hoc networking has grown into a large and diverse field of research, which spans topics from power control to privacy and security. There has been much work on routing in ad hoc networks, and protocols such as DSR [12], AODV [19], Grid [15], and DSDV [20] have been shown in simulation to work very well on small to medium networks [15, 3]. However, our experience with two wireless networks leads us to believe that there are significant challenges left in finding and choosing usable routes, even in small ad hoc networks which are static.

To explore how ad hoc protocols work when implemented as part of a complete system, we built two experimental wireless networks. The first, called the "indoor" net, has 18 small PCs as nodes on the fifth and sixth floors of our building, as shown in Figure 1. We chose node locations that would keep the network connected, while also providing spatial diversity. Each indoor PC has a Cisco Aironet 340 wireless adapter [1]; these adapters implement the IEEE 802.11b Direct Sequence Spread-Spectrum protocol [7], and have 30 mW of output power.

The second "rooftop" network [4] has seven nodes spread over one square kilometer near our lab, as shown in Figure 4. This is a dense residential area with primarily two- and three-story buildings. Nodes have external omni-directional antennas attached to chimney-tops, except for Node 30, which is on the ninth floor of our lab building and has a directional antenna aimed roughly at Node 7. This network uses Cisco 350 wireless adapters, which are like the 340s, but with 100 mW of output power.

This research was supported by grants from NTT Corporation under the NTT-MIT collaboration. More details about the Grid project are available at http://www.pdos.lcs.mit.edu/grid.

Both networks run our implementation of the DSDV protocol. We thought that DSDV would be a good choice because the networks are static, and DSDV seemed to be one of the simpler protocols to implement. Indeed, in simulation scenarios with little to no mobility, like our static networks, DSDV has been shown to deliver the same number of packets as protocols such as DSR and AODV [3]. Unfortunately, our daily experience with the indoor network has been disappointing: tasks like interactive logins and file transfers are often unusably slow.

Our first hypothesis was that there might be no high-quality paths connecting some parts of the network. Although there is a wide range of link qualities in the network, it turns out that the network is still well connected with links that deliver more than 90% of the packets. For example, Figure 2 shows the subset of inter-node radio links that had loss rates less than 10% at a particular time; with the exception of one node, these links form a connected graph.

To find an approximate lower bound on how well a routing protocol should be able to do, we tried to find the path with the highest throughput between each pair of nodes in the network. For each pair of nodes we generated all possible paths of length less than or equal to four hops. We pruned the large number of resulting paths by eliminating paths containing links with low delivery rates, as determined by earlier link measurements. We then randomly chose among the remaining paths so that each pair of nodes had a few paths of each length in each direction. For each path, the source node sent source-routed packets as fast as possible over that path to the destination node for 30 seconds; the destination measured the arrival rate for each path. The black points in Figure 3 indicate the best throughput for each pair of nodes. We ran a similar experiment using DSDV to find routes and forward packets between each pair of nodes. The grey points in Figure 3 show the throughput of traffic routed by DSDV for each node pair. For almost all of the node pairs, DSDV routed packets over paths that were considerably slower than the best path. DSDV sometimes performed better than the "best" route because the two experiments were necessarily run at different times, so network conditions were not identical; in addition, since we only tested some of the possible paths between each pair, we may have missed the actual best path in some cases.

This result was surprising: all else being equal, multi-hop path capacity is determined by hop count [14], and DSDV finds shortest paths. But of course, all else is not equal. Figure 6 shows the packet transmission rates of three-hop paths from node 10 to 18, from an earlier experiment. These are often the shortest paths that a routing algorithm could find from 10 to 18. It is clear from the graph that if a routing protocol made a random choice among these paths, it

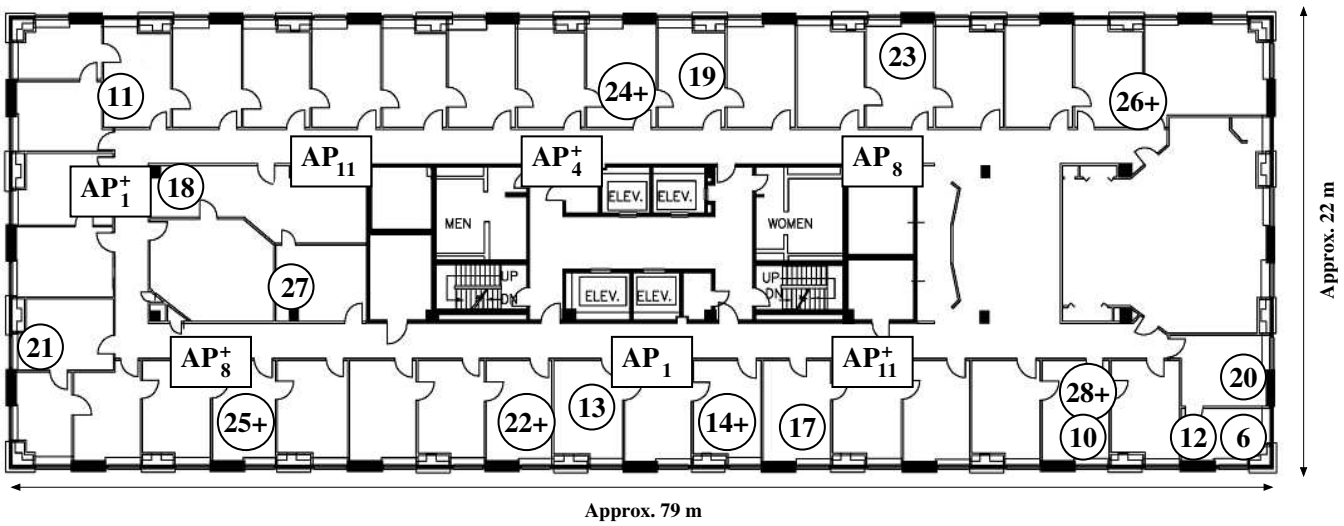


Figure 1: A map of the indoor network. The circles are nodes; the squares marked 'AP' are access points, labeled with their channel. Our experiments do not use the APs, but they may have affected the results. 6th floor nodes and APs are marked with '+'.

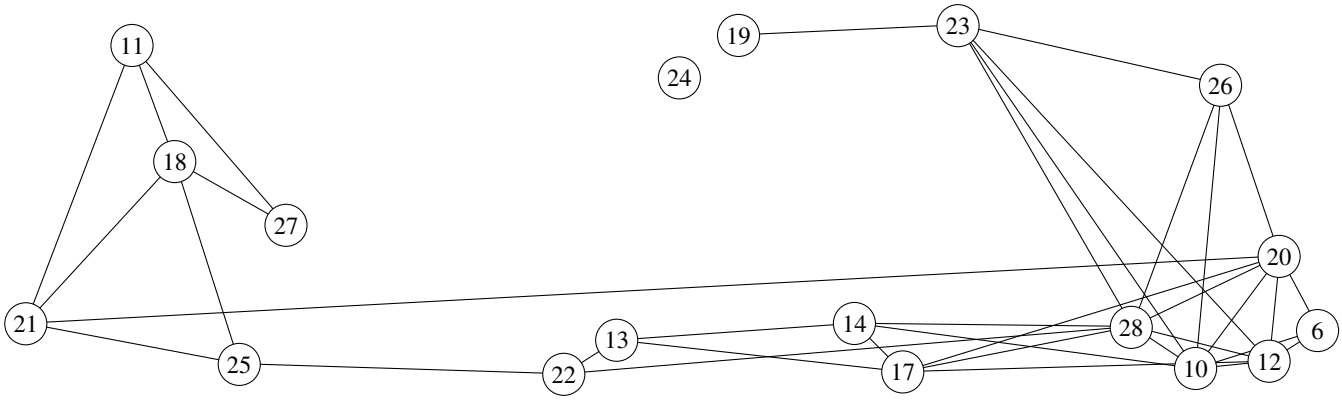


Figure 2: Indoor network links with greater than 90% delivery rate in both directions show that the network is well connected (8-Feb-13:30-50-byte). Node 24 was not functioning during this experiment. For clarity, node positions have been slightly changed from Figure 1.

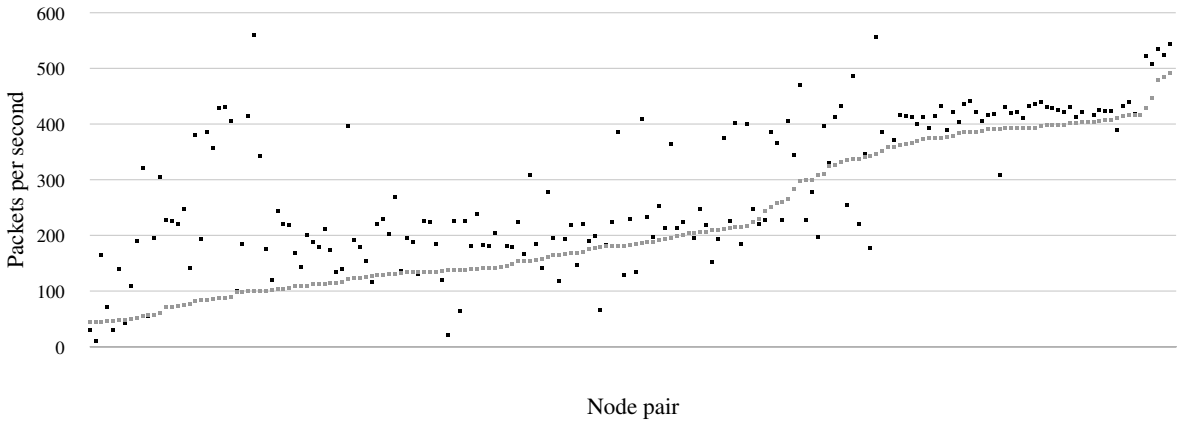


Figure 3: DSDV and best static route throughput for pairs of nodes in the indoor network, sorted by the DSDV throughput of each pair. DSDV does worse on 152 out of the 185 pairs. DSDV throughput is shown in gray, static route throughput in black (DSDV data from 30-Sep-22:10; static route data from 19-Aug-28-Aug). Packets had 124 bytes of 802.11 payload.

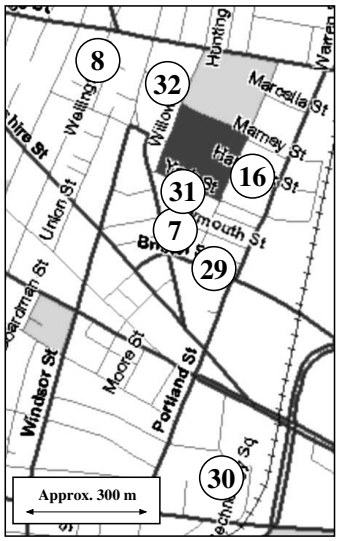


Figure 4: A map of the outdoor rooftop network.

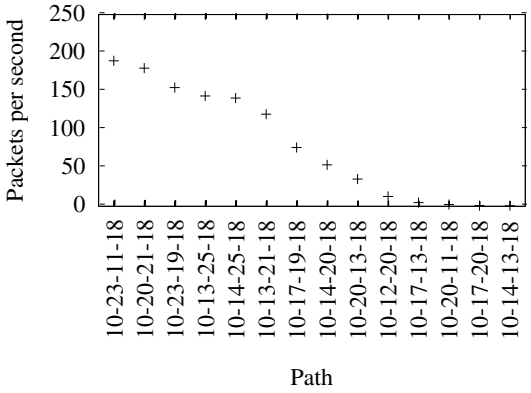


Figure 6: Measured capacity on various 3-hop paths from node 10 to node 18 (27-Jun-17:00). The units are packets per second; each packet contains 124 bytes of 802.11 payload.

would be unlikely to choose the path with the best packet throughput. In fact, in this case a randomly chosen path would achieve less than half of the maximum path throughput more than half of the time. Although we do not have as detailed data for all other node pairs, initial results imply that we can expect to see similar behavior for other node pairs: a wide range in packet throughput among the shortest paths between each pair.

The rest of the paper uses link quality measurements to explain the observed differences in quality of the paths between given node pairs. It also presents detailed wireless link measurements which outline some of the difficulties that would be involved in finding good paths. The measurements include the distribution of link-by-link loss statistics, to see how accurately we need to distinguish links; the extent to which link quality differs depending on link direction; the rate at which link quality changes with time, which determines the usefulness of averaging techniques; and the relationship between signal strength reported by 802.11 cards and loss rate. We conclude the paper by outlining some potentially fruitful avenues of research.

2. Related Work

One solution to low link quality is to improve the apparent quality with some form of redundancy. Forward error correction, MAC-level acknowledgment and retransmission, and solutions such as Snoop-TCP [2] and Tulip [18] all take this approach. Even with these techniques it is preferable to use high-quality rather than low-quality links: retransmissions (or other redundancy) reduce useful link capacity and generate interference. For these reasons, error correction should be combined with route selection that avoids low-quality links when possible.

A number of existing ad hoc wireless routing algorithms collect per-link signal strength information and apply a threshold to eliminate low-quality links [10, 9, 21, 8, 13]. This approach has two drawbacks. First, thresholding may eliminate links that are necessary for connectivity, or include links of dubious quality; both of these are likely to be issues if many links are of intermediate quality. Second, Section 3.3 shows that, in the case of some 802.11 cards, reported signal strength is a poor predictor of loss rate.

Wireless Quality-of-Service (QoS) algorithms approach route selection from the top down. Some techniques explicitly schedule transmission slots in time or frequency division MAC layers to provide bandwidth guarantees [11, 6, 16, 17, 24], while others treat the MAC as opaque, and rely upon it for bandwidth and delay information and constraints [5, 23, 22]. These approaches are only successful if the lower layers can provide accurate information about the actual links, such as the average number of usable transmission slots, or the achievable throughput. However, none of these approaches consider the case of lossy links.

3. Link Behavior

We carried out several experiments to evaluate the link characteristics, particularly loss rate, between every pair of nodes. In each complete experiment, each node in turn broadcasts a series of equally-sized packets at a constant rate; the other nodes record which packets they receive. Because we wanted to discover the underlying link behavior, we used broadcasts to avoid the 802.11 ACK and RTS/CTS mechanisms, which hide the real loss rates.

No routing protocol is running during these experiments: only experiment packets are sent or received on each node's wireless interface. The interfaces are running in 802.11 ad hoc mode. Packets were sent at about 0.4 Mbps, which is well below the minimum 802.11 capacity of 1 Mbps. However, on some occasions nodes were not able to broadcast at the desired rate, perhaps because of 802.11 traffic outside our control, or interference appearing to the card as carrier.

3.1 Distribution of Link Quality

The link quality distribution will affect the way we distinguish good links from bad. Most current ad hoc routing protocols assume that link quality follows a bi-modal distribution, where links are either very good or very bad. Protocols assume that if a link can successfully deliver a routing control packet, then the link is useful for delivering data. In this case, protocols don't need a very accurate link classifier because if a link seems at all usable, it is likely to be very good. On the other hand, the link quality distribution may be very spread out. In this case, protocols will need to accurately differentiate good links from links that aren't suitable for data, but still deliver some control packets. Our experiments indicate that the distribution is spread out.

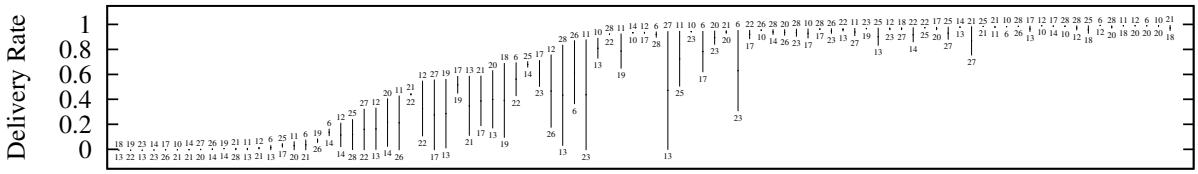


Figure 5: Delivery rates for each link pair in the indoor network (8-Feb-13:30-50-byte). The y values of the two ends of each line indicate the delivery rate in each direction; the numeric labels indicate the sending node for that delivery rate. Links with zero delivery rate in both directions are omitted. 91 links are shown.

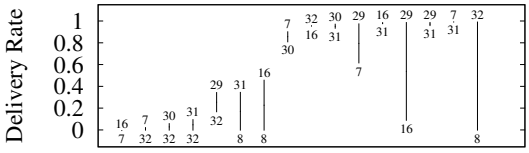


Figure 7: Delivery rates on each link pair for the outdoor rooftop network (6-Mar-18:30-50-byte), as in Figure 5. 16 links are shown.

We conducted a set of experiments with the indoor testbed using 50-byte UDP packets (8-Feb-13:30-50-byte). Each node transmitted 1024 packets per second for 300 seconds. Figure 5 shows the results for each link pair, excluding link pairs which were not able to communicate at all. About 30% of the link pairs shown are completely unusable, although they might deliver a few routing packets. The best 40% of link pairs deliver about 90% or more of their packets; these are the links we would like to use in routes. The delivery rates of the remaining links are spread out. Other experiments on different days, at different times, and with different parameters confirm that in general the links in the network exhibit a wide range of delivery rates.

Link pairs that are very good in one direction tend to be good in both directions, and pairs that are very bad in one direction tend to be bad in both directions. However, at least 30% of the link pairs shown have asymmetric delivery rates, defined as a difference of more than 20% between the rates in each direction.

Figure 7 summarizes an identical set of experiments carried out on our rooftop network (6-Mar-18:30-50-byte). Like the indoor network, the rooftop network has widely varying delivery rates, with noticeable asymmetry. Experiments over several days exhibited similar distributions of delivery rates. The wide variation in delivery rates for both networks suggests that routing protocols may often choose links that are high enough quality to pass routing protocol packets, but which still have substantial loss rates.

3.2 Link Variation over Time

One way to determine link quality is to measure it by counting the number of packets received over a period of time. However, the accuracy of this approach is sensitive to length of time over which the delivery rate is measured. For example, Figure 8 shows the second-by-second delivery rates for two links (8-Feb-13:30-50-byte). The

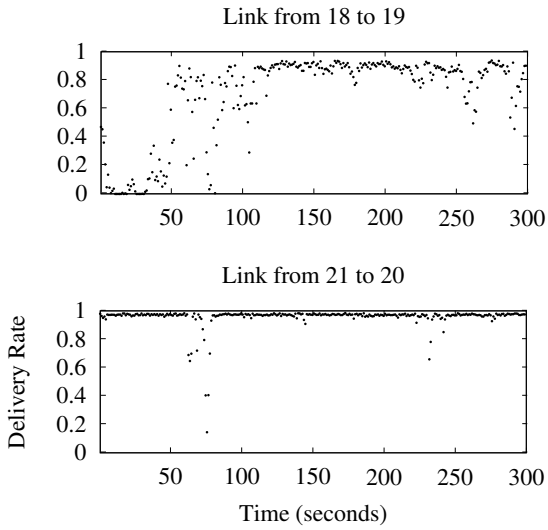


Figure 8: Example per-second variation in link delivery rates (8-Feb-13:30-50-byte). Each point is the delivery rate over one second. The delivery rate of the link from 18 to 19 fluctuates quickly, while the link from 21 to 20 is comparatively stable.

graphs show that while delivery rates are generally stable, they can sometimes change very quickly. Averaging may work well on the link from node 21 to 20, but it is likely to hide much of the detailed behavior of the link from node 18 to 19.

Figure 9 summarizes variation in loss rate over time for all links. For each link, we calculated the mean and standard deviation of the 1- and 10-second loss rates over the whole experiment. The graph shows the cumulative distribution of these standard deviations, normalized by the respective means. We use loss rates rather than delivery rates for this analysis to emphasize the changes in the delivery rate on links with low loss, since very lossy links are useless for data traffic regardless of their variation.

Results for 1- and 10-second windows show that quite a few links vary greatly on these times scales. For example, half of the links had standard deviations in their 1-second loss rates that exceeded half of the mean 1-second loss rate. This suggests that wireless routing protocols should use agile predictors of link loss rates.

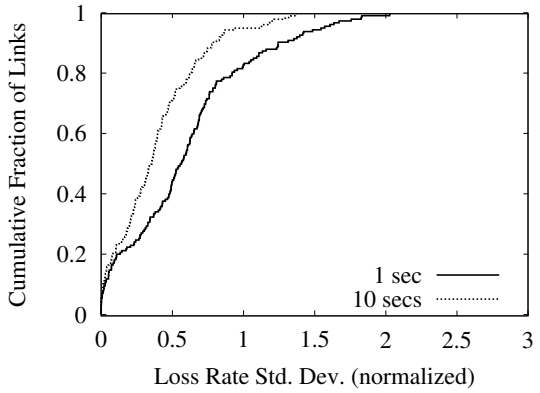


Figure 9: The indoor network's cumulative distribution of the standard deviation of short-term link *loss* rates. For each link, the loss rate is calculated over 1- and 10-second intervals, and the standard deviation is normalized by the link's mean loss rate (8-Feb-13:30-50-byte). Many links show significant variation in short-term loss rates over time.

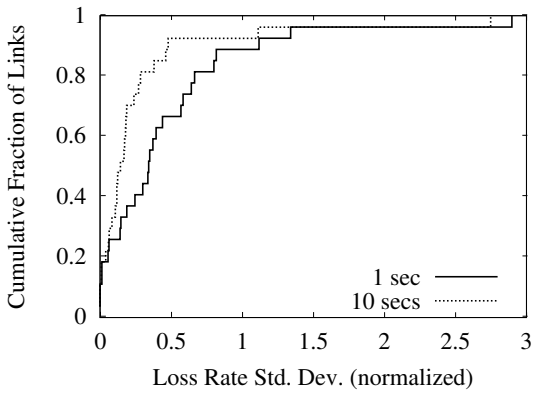


Figure 10: The rooftop network's cumulative distribution of the normalized standard deviation of short-term link loss rates over 1- and 10-second intervals, calculated as in Figure 9 (6-Mar-18:30-50-byte).

Figure 10 shows the variation in short-term loss rates from the same experiment as in Figure 9, but carried out on the rooftop network (6-Mar-18:30-50-byte). This figure shows that short-term loss rates in the rooftop network vary much like those in the indoor network.

In addition to looking at short-term loss rates, we measured how link delivery rates change throughout the day. Figure 11 shows delivery rates for two links over a 24-hour weekday period in January. Every half-hour, each node tried to broadcast 100 1024-byte packets per second for 30 seconds. The results for the link from node 6 to node 23 are particularly interesting; the fact that the quality increases dramatically at 8 am suggests that opening office doors in the morning increases link quality.

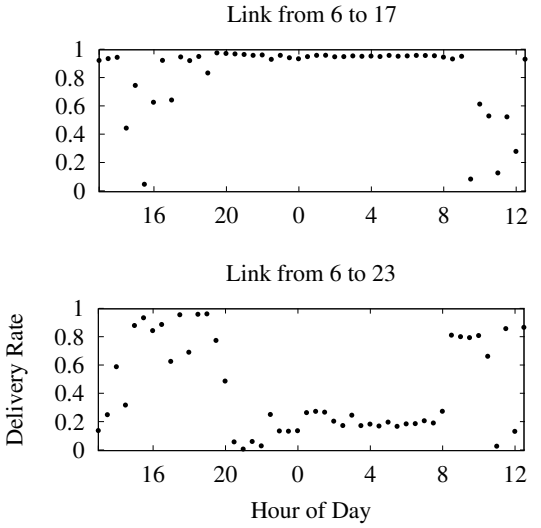


Figure 11: Example variations in link delivery rates throughout the day (10-Jan-24h-1024-byte).

3.3 Link Signal Strength

Signal strength could potentially be helpful in predicting link quality. To explore this possibility, we recorded signal strength (dBm) from the radio interface for each received packet during the link experiments. Figure 12 shows how the short-term delivery rate varies with these values for a few example links. Unfortunately there is no good correlation between delivery rate and the radio's measurements. Instead, the data reflect the physical fact that received signal strength is mostly a function of the distance between nodes. The link from 18 to 19 is a long link with low signal strength, and as a result is very susceptible to noise. Since the successful reception of a packet depends on the signal to noise ratio (SNR) at the receiver, this link's delivery rate varies significantly. In contrast, the link from 18 to 11 is a short link. Because it has a very high received signal strength, it is robust to noise and has high delivery rates. The links from 23 to 19 and 27 to 11 are medium range links which still deliver most packets. Since our radios don't provide a noise estimate with which to compute the SNR, we cannot determine much about the links from the signal strength estimate.

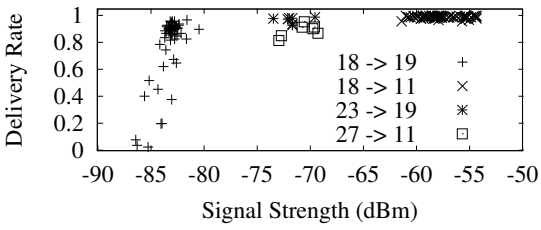


Figure 12: Delivery rates over 100 msecs versus average signal strength of received packets for four example links on the indoor network (6-Mar-18:30-50-byte). For clarity, only 1 in 50 of the data points are shown for each link. Signal strength more accurately reflects link distance than delivery rate.

4. Research Agenda

Based on the measurements presented here, we intend to develop techniques to help ad hoc routing protocols choose high-quality routes. The main challenges involve practical estimates of link quality and techniques to combine link metrics into useful path metrics.

One obstacle to using delivery rate as a link metric is that it requires many packet transmissions to measure; this is a problem if nodes move or if the environment changes rapidly. Signal-to-noise ratio might be useful as a fast predictor of delivery rates, if it is available from the radio hardware.

Combining route metrics to form a path metric is not straightforward. For example, the product of the delivery rates of the links in a path does not predict anything useful when 802.11 link-layer retransmissions are used. Self-interference by successive nodes in a route [14] may sometimes make long routes with good links less attractive than short routes with less good links. We are currently evaluating use of the expected total number of transmissions of a packet along a path (including forwarding and retransmission) as a path metric. This metric has a number of advantages: it captures the route's impact on the spectrum, it can be computed incrementally from link delivery rates, and it penalizes longer routes.

Finally, we plan to explore how protocols such as DSR and AODV handle the link quality distribution seen on our testbeds. These protocols are not simply shortest-path protocols: they notice when a route is using a very low-quality link and try to find a different route. We intend to compare the quality of the routes they find with the quality of "best" routes found by exhaustive search.

5. Acknowledgments

The authors would to thank Jinyang Li for her insightful comments, and for her work on the ad hoc capacity problem.

6. References

[1] *Using the Cisco Aironet 340 Series PC Card Client Adapters*. Cisco Systems Inc., March 2000. Part number: OL00379-01.

[2] Hari Balakrishnan, Venkata N. Padmanabhan, Srinivasan Seshan, and Randy H. Katz. A comparison of mechanisms for improving TCP performance of wireless links. *IEEE/ACM Transactions on Networking*, 6(5), December 1997.

[3] Josh Broch, David A. Maltz, David B. Johnson, Yih-Chun Hu, and Jorjeta Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *Proc. ACM/IEEE MobiCom*, pages 85–97, October 1998.

[4] Benjamin A. Chambers. The Grid roofnet: a rooftop ad hoc wireless network. Master's thesis, Massachusetts Institue of Technology, May 2002.

[5] Shigang Chen and Klara Nahrstedt. Distributed quality-of-service routing in ad hoc networks. *IEEE Journal on Selected Areas in Communcations*, 17(8), August 1999.

[6] T.-W. Chen, J.T. Tsai, and M. Gerla. QoS routing performancs in multihop, multimedia, wireless networks. In *Proceedings of IEEE ICUPC '97*, 1997.

[7] IEEE Computer Society LAN MAN Standards Committee. *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*. New York, New York, 1997. IEEE Std. 802.11–1997.

[8] Brian H. Davies and T. R. Davies. The application of packet switching techniques to combat net radio. *Proceedings of the IEEE*, 75(1), January 1987.

[9] Rohit Dube, Cynthia D. Rais, Kuang-Yeh Wang, and Satish K. Tripathi. Signal stability-based adaptive routing (SSA) for ad hoc mobile networks. *IEEE Personal Communications*, February 1997.

[10] Tom Goff, Nael B. Abu-Ghazaleh, Dhananjay S. Phatak, and Ridvan Kahvecioglu. Preemptive routing in ad hoc networks. In *Proc. ACM/IEEE MobiCom*, July 2001.

[11] Yu-Ching Hsu, Tzu-Chieh Tsai, Ying-Dar Lin, and Mario Gerla. Bandwidth routing in multi-hop packet radio environment. In *Proceedings of the 3rd International Mobile Computing Workshop*, 1997.

[12] David B. Johnson. Routing in ad hoc networks of mobile hosts. In *Proc. of the IEEE Workshop on Mobile Computing Systems and Applications*, pages 158–163, December 1994.

[13] John Jubin and Janet D. Tornow. The DARPA packet radio network protocols. *Proceedings of the IEEE*, 75(1), January 1987.

[14] Jinyang Li, Charles Blake, Douglas S. J. De Couto, Hu Imm Lee, and Robert Morris. Capacity of ad hoc wireless networks. In *Proceedings of the 7th ACM International Conference on Mobile Computing and Networking*, pages 61–69, Rome, Italy, July 2001.

[15] Jinyang Li, John Jannotti, Douglas S. J. De Couto, David R. Karger, and Robert Morris. A scalable location service for geographic ad hoc routing. In *Proc. ACM/IEEE MobiCom*, August 2000.

[16] Chunhung Richard Lin. On-demand QoS routing in multihop mobile networks. In *Proc. IEEE Infocom*, April 2001.

[17] Anastassios Michail and Anthony Ephremides. Algorithms for routing session traffic in wireless ad-hoc networks with energy and bandwidth limitations. In *Proceedings of 12th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, 2001.

[18] Christina Parsa and J. J. Garcia-Luna-Aceves. TULIP: A link-level protocol for improving TCP over wireless links. In *Proc. IEEE Wireless Communications and Networking Conference 1999 (WCNC 99)*, September 1999.

[19] Charles Perkins, Elizabeth Royer, and Samir R. Das. Ad hoc On-demand Distance Vector (AODV) routing. Internet draft (work in progress), Internet Engineering Task Force, January 2002. `http://www.cs.ucsb.edu/~ebelding/txt/aodvid.txt`.

[20] Charles E. Perkins and Pravin Bhagwat. Highly dynamic Destination-Sequenced Distance-Vector routing (DSDV) for mobile computers. In *Proc. ACM SIGCOMM Conference (SIGCOMM '94)*, pages 234–244, August 1993.

[21] Ratish J. Punnoose, Pavel V. Nitkin, Josh Broch, and Daniel D. Stancil. Optimizing wireless network protocols using real-time predictive propagation modeling. In *Radio and Wireless Conference (RAWCON)*, August 1999.

[22] Samarth H. Shah and Klara Nahrstedt. Predictive location-based QoS routing in mobile ad hoc networks. In *Proceedings of IEEE International Conference on Communications*, 2002.

[23] Prasum Sinha, Raghupathy Sivakumar, and Vaduvur Bharghavan. CEDAR: A core-extraction distributed ad hoc routing algorithm. In *Proc. IEEE Infocom*, March 1999.

[24] Chenxi Zhu and M. Scott Corson. QoS routing for mobile ad hoc networks. In *Proc. IEEE Infocom*, June 2001.

pSearch: Information Retrieval in Structured Overlays

Chunqiang Tang[†]
Dept. of Computer Science
Univ. of Rochester
Rochester, NY 14627-0226
sarrmor@cs.rochester.edu

Zhichen Xu
HP Laboratories
1501 Page Mill Rd., MLS 1177
Palo Alto, CA 94304
zhichen@hpl.hp.com

Mallik Mahalingam
HP Laboratories
1501 Page Mill Rd., MLS 1177
Palo Alto, CA 94304
mmallik@hpl.hp.com

ABSTRACT

We describe an efficient peer-to-peer information retrieval system, pSearch, that supports state-of-the-art content- and semantic-based full-text searches. pSearch avoids the scalability problem of existing systems that employ centralized indexing, or index/query flooding. It also avoids the nondeterminism that is exhibited by heuristic-based approaches. In pSearch, documents in the network are organized around their vector representations (based on modern document ranking algorithms) such that the search space for a given query is organized around related documents, achieving both efficiency and accuracy.

1. INTRODUCTION

The sheer quantity of Internet content is beyond the capability of any centralized search engine. A study [1] conducted by BrightPlanet Corporation in March 2000 estimated that the deep Web might contain 550 billion documents, far more than the 1.2 billion pages that Google had identified, not to mention the 600 million pages that Google was able to search at that time. Moreover, this data volume continues to grow at an astonishing rate, doubling each year. This calls for an infrastructure that is able to scale at a comparable rate.

Peer-to-peer (P2P) systems [8], on the other hand, are gaining popularity due to their scalability, fault-tolerance, and self-organizing nature, raising hope for building completely decentralized information retrieval (IR) systems.

Current P2P searching systems, however, are either unscalable or unable to provide deterministic guarantees. Usually they are based on one of the following techniques: centralized indexing, query flooding, index flooding, or heuristics. Centralized indexing systems such as Napster suffer from the single point of failure and performance bottleneck at the index server. Flooding-based techniques such as Gnutella send a query or index to every node in the system, consuming huge amount of network bandwidth and CPU cycles. Heuristics-based techniques try to improve performance by directing searches to only a fraction of the population, but they may fail to retrieve important documents.

Distributed hash table (DHT) systems such as CAN [9] do provide good scalability, but they only offer a simple interface for storing and retrieving `(key, value)` pairs, and hence are not suitable for full-text searches.

Besides the performance inefficiency, a common problem with most existing P2P systems is that they usually ignore the advanced ranking algorithms devised by the IR community through decades of refinement and evaluation, and thereby rely on naive keyword-based searches. Examples of successful IR algorithms include the vector space model (VSM) and latent semantic indexing (LSI) [2]. These algorithms represent documents and queries as vectors, and measure the similarity between a query and a document as the cosine of the angle between their vector representations.

Our goal is to build a scalable P2P IR system, *pSearch*, that has efficiency of DHT systems and accuracy of state-of-the-art IR algorithms. To achieve this goal, in pSearch, documents in the network are organized around their vector representations such that the search space for a given query is organized around related documents, achieving both efficiency and accuracy.

In this paper we describe two algorithms: pVSM that is based on the vector space model, and pLSI that uses latent semantic indexing. For pLSI, our simulation shows that it is able to achieve accuracy comparable to the centralized algorithm while visiting only 0.4-1% nodes in the system. For pVSM, the number of visited nodes is bounded by the number of terms in a query, which is usually small.

In pSearch, queries can be processed concurrently in different regions of the overlay, it hence has high throughput. It also has short response time, because nodes collectively search the partitioned document space to resolve a query. The performance of IR systems is usually bounded by disk I/O. In pSearch, indices are massively partitioned such that the indices held by a node may fit in its main memory.

Several features distinguish pSearch from other systems.

- It works in a completely decentralized manner, with neither single point of failure nor complex hierarchy.
- It supports content and semantic searches, as opposed to naive keyword matches.
- It is scalable, efficient, and effective. Index/query flooding are avoided. Its accuracy is comparable to that of the state-of-the-art centralized IR algorithms.

[†]This work was done when the author worked at HP Labs in summer 2002. He is supported in part by NSF grants CCR-9988361, CCR-0219848, ECS-0225413, and EIA-0080124.

HotNets-I '02 Princeton, New Jersey, USA

The remainder of the paper is organized as follows: Section 2 provides background information about DHT and IR. Section 3 describes pVSM and pLSI. We discuss pSearch applications in Section 4, present related work in Section 5, and conclude in Section 6.

2. BACKGROUND

pSearch is built on eCAN [12] (a hierarchical version of CAN) and uses extensions to VSM and LSI [2]. The Cartesian space abstraction of CAN makes it particularly attractive when used to store vector representations of documents generated by IR algorithms such as VSM and LSI. We describe these basic components below.

DHT systems and eCAN. Recent DHT systems, represented by CAN, Chord, and Pastry, offer an administration-free and fault-tolerant application-level overlay network. CAN stands for content-addressable network. It organizes the logical space as a d-dimensional Cartesian space (a d-torus) and partitions it into *zones*. One or more nodes serve(s) as owner(s) of a zone. An object `key` is a point in the space, and the object is stored at the node whose zone contains the point. Locating an object is reduced to the problem of routing to the node that hosts the object. Routing from a source node to a destination node is equivalent to routing from one zone to another in the Cartesian space. A node join corresponds to picking a random point in the Cartesian space, routing to the zone that contains the point, and splitting the zone with its current owner(s). eCAN improves CAN's logical routing cost to $O(\log(n))$, and takes only routes that closely approximate the underlying Internet topology.

Vector space model. VSM represents documents and queries as *term vectors*. Each element of the vector represents the importance of a word (*term*) in the document or query. The weight of an element is often computed using the *term frequency * inverse document frequency* (TF*IDF) scheme [2]. The intuition behind it is that two factors decide the importance of a term in a document: the frequency of the term in the document and the frequency of the term in other documents. If a term appears in a document with a high frequency, there is a good chance that the term could be used to differentiate the document from others. However, if the term also appears in a lot of other documents, e.g., `computer`, the importance of the term should be penalized. VSM usually normalizes vectors to unit Euclidean norm to compensate for differences in document length. During a retrieval operation, the query vector is compared to document vectors. Those closest to the query vector are considered to be similar and are returned. A common measure of similarity is the cosine of the angle between vectors.

Latent semantic indexing. Literal matching schemes such as VSM suffer from synonymy, polysemy, and noise in documents. LSI has been proposed to address these problems. It uses singular value decomposition (SVD) to transform and truncate a matrix of term vectors computed from VSM to discover the semantics of terms and documents. For instance, although `car`, `vehicle`, and `automobile` are different terms, LSI may be able to discover that they are related in semantics. Intuitively, LSI transforms a high-dimensional term vector into a medium-dimensional *semantic vector* by projecting the former into a medium-dimensional semantic subspace. The basis of the semantic subspace is computed using SVD. Semantic vectors are normalized and their similarity is measured as in VSM.

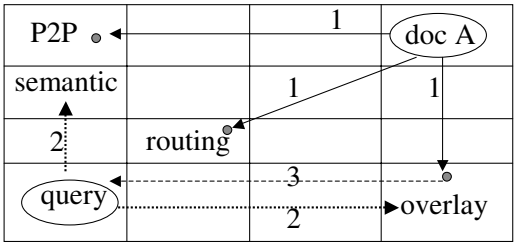


Figure 1: pVSM in a 2-dimensional CAN.

3. PSEARCH ALGORITHMS

The fundamental idea in pSearch is to store information of documents in DHT-based overlay networks around their vector representations such that the search space for a given query is organized around related documents. We present two algorithms here: pVSM and pLSI. Each is described in terms of the basic ideas, challenges, and solutions.

3.1 Peer-to-Peer VSM (pVSM)

As is pointed out by Cuenca-Acuna and Nguyen [4], a straightforward implementation of VSM in DHT needs to store `(term, index)` pairs for each term appearing in each document, a formidable task simply due to the large number of terms in each document. Recent studies, however, show that the frequency of terms in a document usually follows a *Zipf* distribution, meaning that a small number of keywords can categorize a document's content.

Basic ideas. In pVSM, nodes are organized using CAN and each node is responsible for storing indices containing certain keywords. Given a document, a term vector is computed using VSM. The m most heavily-weighted elements (terms) t_i, $i = 1, ..., m$ are identified, and the corresponding `(`$h(t_i)$`, index)` pairs are stored in DHT. (m can vary on a per-document basis, Section 3.4.) Here h is a hash function mapping strings into points in CAN, and `index` is the actual document or a pointer to the actual document. During a retrieval operation, each term in the query is hashed into a point using h and the query is routed to nodes whose zones contain the points. Each of the nodes retrieves the best matching indices locally using VSM, and sends them back to the node initiating the query. The query initiator ranks them globally, discards those with low ranks, and presents the rest to the user.

Figure 1 illustrates this process. Each zone is owned by a node. The little dots represent indices. Given the document A, its important keywords—`P2P`, `overlay`, and `routing`—are identified using VSM and the index is published to corresponding nodes (*step 1*). Given a query of `"semantic overlay"`, it is forwarded to nodes responsible for keyword `semantic` and `overlay`, respectively (*step 2*). The two nodes then search and return matching indices using VSM (*step 3*).

Challenges and solutions. The first problem pVSM faces is synonymy. An index may be stored under one term but retrieved using its synonyms. This problem can be fixed using a thesaurus, by also routing queries to points corresponding to synonyms of the terms in the query.

Second, VSM relies on some global information, such as

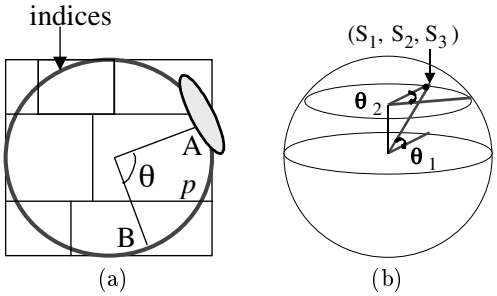


Figure 2: Transforming the sphere space. (a) *In a 2-dimensional CAN, the naive pLSI only places indices on the circle. The similarity ($\cos\theta$) between the two indices A and B is proportional to their distance (p) on the circle: $\cos\theta = \cos p$. The gray region is the flooding area for searching indices close to A in semantics.* **(b)** *Using Equation 1 to transform a 3-dimentional semantic vector.*

the dictionary of the terms that TF*IDF counts, and the *inverse document frequency* (IDF). We call this global information *statistics*. It has been demonstrated that VSM does not need precise statistics to work well, i.e., a good approximation is sufficient.

In pVSM, the initial copy of the statistics are precomputed using samples that are representative of the potential document set. Over time, a combining tree that approximates the underlying network topology and includes randomly chosen nodes is used to sample documents and merge statistics. The size of the statistics grows slowly with the size of the document population. We expect the statistics to change slowly, at the rate of weeks or even months, because statistics are more stable than the documents themselves, especially for a large document population. The root of the combining tree periodically disseminates the statistics to other nodes. The root is a node occupying a well-known zone in CAN. If it fails, one of its neighbors will take over.

3.2 Peer-to-Peer LSI (pLSI)

In the original proposal of CAN, document IDs (keys) are randomly generated with no meanings other than their role in routing. If we use the semantic vectors of documents as keys to place indices in such a way that indices, that are semantically close to each other, are stored logically close to each other in CAN, then searching in the semantic space is reduced to routing in CAN. In the following sections, we first outline a naive pLSI algorithm to highlight the challenges that need to be overcome, and then describe the solutions.

3.2.1 A Naive pLSI algorithm

Let's use $\mathcal{L}$ and $\mathcal{K}$ to denote the LSI semantic space and CAN Cartesian space, respectively, with l and k as the dimensionality of the two spaces. Because l and k are freely tunable, we can map each document to a point in $\mathcal{K}$, by setting $l = k$ and treating its semantic vector as its coordinates in $\mathcal{K}$. Given a document, its semantic vector S is computed using LSI, and the `(S, index)` pair is stored in DHT using eCAN routing. During a retrieval operation, the semantic vector Q of the query is computed and the query is routed using Q as the DHT key. Upon arriving at the destination, it floods the query *only* to nodes within a radius r based on the similarity threshold or the number of wanted documents specified by the user. All nodes that receive the query do a local search using LSI and merge the results back to the user as in pVSM. Because indices of documents similar to the query above the threshold can be stored only within this radius r and we do an exhaustive search within this area, pLSI achieves the same performance as LSI. Ideally, this radius r is small and the involved nodes are only a small fraction of the entire population.

3.2.2 Challenges for pLSI

There are four major problems that need to be solved.

- *Sphere distribution of semantic vectors.* Recall that semantic vectors are normalized and reside on the surface of the unit sphere $\mathcal{U}$ in $\mathcal{L}$, leading to an unbalanced load as depicted in Figure 2(a).
- *Uneven distribution of semantic vectors.* Even if the first problem is ignored and nodes are uniformly distributed on $\mathcal{U}$, it still suffers from hot spots because semantic vectors are not uniformly distributed on $\mathcal{U}$.
- *The curse of dimensionality.* Due to $\mathcal{L}$'s medium dimensionality (100-300), a straightforward nearest neighbor search in $\mathcal{L}$ will not be effective until a large fraction of the nodes are visited [11].
- *The global state problem.* Similar to pVSM, pLSI also needs some global information to work.

3.2.3 Transforming the Sphere Space

To solve the first problem, we transform a semantic vector $S = (s_1, s_2, ..., s_l),\ ||S||_2 = 1$ in $\mathcal{L}$ into a *parameter vector* $(\theta_1, \theta_2, ..., \theta_{l-1})$ in the $(l-1)$-dimensional polar subspace $\mathcal{P}$. Accordingly, we set $l - 1 = k$. Given a document or a query, we use the parameter vector, instead of the semantic vector, as the DHT key for eCAN routing. (To reiterate, we deal with three kinds of vectors: the *term vector* generated by VSM, the *semantic vector* generated by LSI, and the *parameter vector* generated by this transformation.)

This transformation does exist because points on $\mathcal{U}$ only have $l - 1$ degrees of freedom. Equation 1 achieves the exact goal. An example of this transformation is shown in Figure 2(b). Note that even after the transformation, parameter vectors are still not uniformly distributed in $\mathcal{P}$.

$$\theta_j = \arctan(\frac{s_{j+1}}{\sqrt{\sum_{i=1}^{j} s_i^2}}) \quad j = 1, ..., l-1 \qquad (1)$$

3.2.4 Balancing the Load

We use a modified node bootstrap process to solve the second problem. At node join, we randomly pick a document that the node is going to publish and use the parameter vector of the document as the random point towards which the join request is routed. This bootstrap process has three effects: (1) *Load balancing.* Each node stores roughly the same number of indices because the node distribution in $\mathcal{K}$ approximates the index distribution, given that a large number of nodes exist. [1] (2) *Index locality.* Assuming that a node's contents have some locality, the indices of its contents are likely to be published at itself or neighboring zones. (3) *Query locality.* Suppose that documents owned by a user

[1] Without the transformation in Equation 1, it can still achieve load balancing using a similar bootstrap process, but routing would be less efficient.

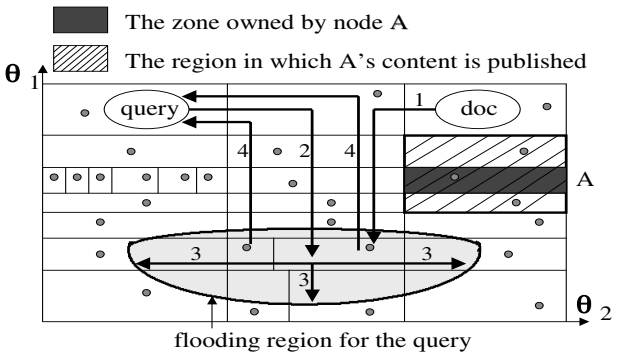


Figure 3: pLSI in a 2-dimensional CAN.

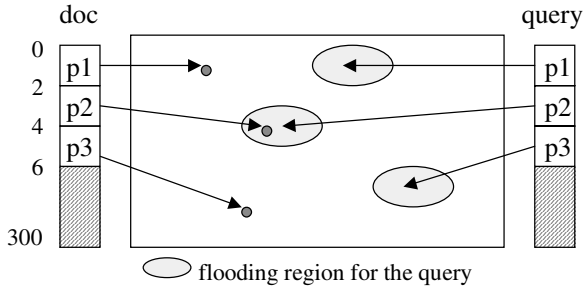


Figure 4: Multi-plane in a 2-dimensional CAN.

are good indications of the user's interests. Then queries submitted by the user would usually be searched in neighboring zones of where the query is submitted. Both the index and query locality lead to efficient eCAN routing.

Figure 3 depicts how pLSI works in a 2-dimentional CAN with the first two modifications. Given a document, its index is stored in DHT using its parameter vector as the key for eCAN routing (*step 1*). Given a query, it is first routed using its parameter vector as the DHT key (*step 2*) and then flooded to a small region computed from the given similarity threshold (*step 3*). Nodes in the flooding region search and return matching indices using LSI (*step 4*). Note that indices are not uniformly distributed but the number of indices per zone is roughly the same. Because of the transformation in Equation 1, the flooding radius r is not uniform in different directions. For node A, it is likely that its content is published in neighboring zones because of the *index locality* induced by the bootstrap process.

3.2.5 Dispelling the Curse of Dimensionality

Existing techniques [11] to relieve the curse of dimensionality usually do not work well at hundreds of dimensions, and require changing the DHT abstraction. In pLSI, domain-specific knowledge allows us to attack this problem at a lower cost. As a result of the SVD decomposition, the elements appearing earlier in a semantic vector are more important than those appearing later. This property also holds for parameter vectors. By aggressive cutoff, a parameter vector can be reduced to only a few dimensions to ease the nearest neighbor search, at the cost of precision.

We use a multi-plane scheme to reduce the dimensionality while keeping good precision. We partition lower elements of a parameter vector into multiple low-dimensional subvectors, with one subvector on each *plane*. The dimensionality of CAN is set to that of an individual plane. Each of the subvectors is used as the DHT key for routing. A document index is published to a node on each plane. A query is also routed and flooded on each plane. Given a query, each plane independently returns matching documents to the query initiator, based on subvectors on that plane. These returned documents form a *pre-selection set*. The query initiator then uses the *full* semantic vector to re-rank documents in this set. To increase the chance that relevant documents are included in the pre-selection set, we increase and vary the number of documents returned on each plane. For instance, when searching the top 15 documents for a query, the first three planes return 30, 20, and 10 documents, respectively, with a total of 60 documents in the pre-selection set.

A multi-plane example is shown in Figure 4. The first 6 elements of the vectors are partitioned into 2-dimensional subvectors on 3 planes. Both the document and query are routed on every plane. In this example, they match on the second plane. Note that only a single eCAN overlay is needed to support multiple planes.

3.2.6 Distributing the Global State

Similarly, pLSI also needs some global statistics: the dictionary, the IDF, and the basis of the semantic space. (Distributing the basis to each node allows the nodes to compute the projections from term vectors to semantic vectors independently.) The solution is also similar: precompute the statistics and update them using samples. When the basis shifts, the semantic vector of a document also changes. This may require redistribution of the index when the difference between two consecutive versions of a semantic vector is so significant that the old one and new one no longer reside in the same zone. Fortunately, since the statistics change slowly, this should happen fairly infrequently.

Computing SVD for a high-dimensional matrix is an intensive process. To avoid starting from scratch at every sample update, we incrementally update the matrix using *SVD-updating*. An alternative is to replace LSI with low-computation approximations such as *Concept Index*, and parallelize them, given abundant nodes in P2P networks.

3.3 pSearch Prototype and Initial Results

The SMART system, developed at Cornell, implements VSM. We extend it with a LSI module and link it with our eCAN simulator [12]. With the basic functionality of pVSM and pLSI in place, we experiment with two corpus widely used in IR research: the small MEDLINE corpus with 1033 documents and the large Text Retrieval Conference (TREC) corpus with 528,543 documents.

Our pVSM experiments running over the MEDLINE corpus demonstrate that storing a document index under the 30 most important terms in the document achieves a result as good as VSM. That is, if VSM returns a document for a query, with high probability, at least one of the terms in the query is among the top 30 terms in the document. The number of visited nodes is bounded by the number of terms in the query, which is usually small.

Since pLSI is more complex, we experiment it with the large TREC corpus, running a large number of configurations in search of the right system parameters. We find that using 4-6 planes with about 10-12 elements on each plane

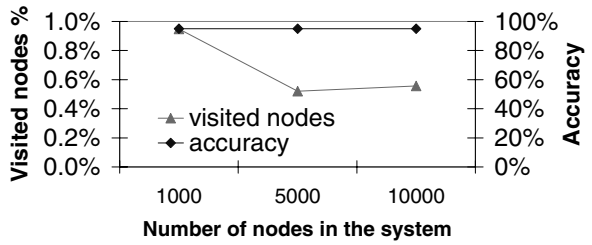


Figure 5: Performance of pLSI, using TREC corpus.

can be very effective. The result [2] is presented in Figure 5, where the X axis is the number of nodes in the system, the left Y axis is the percentage of visited nodes, and the right Y axis is the percentage of overlap between the documents returned by pLSI and LSI. We can see that pLSI is efficient as well as accurate. It only needs to visit about 0.4-1.0% nodes to achieve an accuracy of 95%. The accuracy is even better when more nodes are allowed to be visited. Due to the space limit, more results and our suggestions on configuring the system will be presented elsewhere.

Some differences between pVSM and pLSI are worth noticing. pVSM usually publishes more copies of an index than pLSI does, but it usually sends a query to less nodes during retrieval operations. The flexibility of pLSI allows it to be used in advanced searches such as video and audio retrieval (Section 4). In addition, pLSI can benefit from a modified bootstrap process for load balancing (Section 3.2.4).

3.4 Advanced Issues

We leave some advanced issues for future work.

Indexing at coarse granularity. For nodes that own a large collection of homogeneous documents, it is not necessary to maintain an index for each individual document. It has been shown that indexing at a per-database basis can work well [5]. In pSearch, we use *hierarchical k-means* to cluster documents at a node into collections until the variance inside a collection falls below a given threshold. Each collection is treated as a single document and its index is published in DHT. If a retrieval hits in a collection, the query is forwarded to the publisher for an in-depth search.

Relevance feedback. In addition to the conventional use of relevance feedback to build new queries, we also exploit user retrieval patterns to improve the index distribution. When answering a query, pSearch only returns a list of the best matching documents. It is the user who decides which document to download. Every node remembers the number of times that each document is downloaded and uses it as an estimation of document popularity. For a popular document, pVSM increases the number of terms under which the index is stored, to increase the chance of retrieving it with other queries. In pLSI, upon receiving a downloading request that originates from a successful query, the node records that the document index should move closer to the query vector. The move actually happens when this information is accumulated over a threshold.

Intelligent index replication. pLSI tries to balance load by approximating the index distribution with the node distribution. However, nodes in the system usually have different processing power, storage capacity, and network connectivity. To take advantage of the heterogeneity in system, when the load at a node is low comparing with its capacity, it can selectively replicate indices stored on its direct and indirect neighbors, and process queries on their behalf. The criteria of this selection process may include the popularity of the documents, the load of those neighboring nodes, and user access patterns. The goals of the selective replication are to relieve hot spots and to maximize query performance.

Hierarchical refinement. Using a single semantic space to organize all documents in the universe will inevitably result in coarse-grained classification. There will be dense clusters in the semantic space, within which documents can not easily be distinguished from one other. This cannot be refined by simply increasing the dimensionality of the semantic space because the added dimensions may be helpful to some clusters, but introduce noise into others.

Our solution is to use the global statistics only as an initial step, and then compute localized semantic subspaces using localized information (e.g., only using documents whose semantic vectors fall in a particular region in this global semantic space) to refine the search. Intuitively, the global semantic space clusters documents into coarse areas such as computer science, geology and business, and local semantic subspaces further differentiate documents in each area. The challenge is perhaps to identify the dense clusters and coordinate the refinement process in a decentralized fashion.

Advanced searches. Our focus has been on extensions to the basic VSM and LSI algorithms. Many other searching techniques can also be adopted to our framework.

For instance, instead of querying the entire universe, one might only be interested in documents that reside in a particular domain (e.g., hp.com) or under a certain subdirectory, produced before or after a certain date, etc. Often, it is convenient if one can perform certain *join* functionality to query documents that related to an existing set of documents in a certain way. To provide these features, we suggest including other metadata such as domain, path, date, etc. in the index, to enable functionalities that are analogous to database selection and join.

Some IR systems exploit context information in documents for document ranking. For instance, given a query of `"computer network"`, documents with the two words closeby are ranked higher than documents with the two words far apart. We propose to use pVSM or pLSI to find a medium number of related documents, and then apply other ranking algorithms locally to re-rank them, if desirable.

Application-aware overlay. In Section 3.2.4, we balance the load by fitting the node distribution to the document distribution. This unfortunately may result in inefficient routing and high cost in overlay maintenance. Non-uniform node distribution can also be a result of topology-aware overlay construction and the discriminate use of nodes in terms of forwarding and storage capacity. An interesting problem is to allow the node distribution to fit the application's demand, while not sacrificing the performance of the overlay. Extending the ideas in eCAN [12], we propose to maintain only the basic routing states to make the overlay connected, and employ intelligent route caching based on application behavior to improve performance.

[2] The transformation in Equation 1 is not used in this experiment. It is tested seperately but not incorporated into the simulator yet.

4. OTHER APPLICATIONS

The pSearch technologies should be applicable to many other applications. We give only a few examples here.

Video/Audio. pLSI works by representing media contents as vectors and organizing contents around the vectors. This method can be applied to any media that can be abstracted as vectors and have its object similarity measured as some kind of distance in the vector space. A lot of pattern recognition problems fall into this category. For instance, people also employ SVD to extract algebraic features from images, and use various extractors to derive frequency, amplitude, and tempo feature vectors from music data.

Semantic-based Publish/Subscribe. Going one step further, pSearch provides a decentralized infrastructure for semantic-based Publish/Subscribe. The nodes are natural places for keeping document subscriptions and for document availability detection. The subscription can be described not only in topics and contents, but also in semantics, allowing users to subscribe to unstructured documents that they do not know how to describe precisely. pSearch users may simply describe their needs as *"notify me when documents similar to my collection show up"*.

Deep search in grid. Like the P2P model, the Grid also deals with resource sharing and cooperation among a large number of heterogeneous systems. To provide a uniform resource discovery and IR interface over existing heterogeneous services in Grid, we propose to use an overlay as the pSearch infrastructure to connect existing services. Nodes in the overlay maintain service indices and route queries, using the coarse-granularity indexing technique described in Section 3.4. For services that cannot provide the indices, pSearch either crawls their Web pages or uses *query-based sampling* to extract a good summary of the service contents.

Semantic-based resource discovery. pSearch, in essence, provides a decentralized resource discovery service, in which providers publish summaries of their resources and consumers use DHT routing to discover the resources. Both publishing and discovery can be expressed in either object IDs, contents, or semantics. A lot of applications can be implemented with this paradigm: P2P cooperative caching, interest-based online bidding, interest-based chat room, resource discovery in *ad hoc* networks, and so forth.

5. RELATED WORK

Both routing indices [3] and attenuated bloom filter [10] use heuristics to selectively forward queries to a subset of neighbors that are likely to contribute in resolving the query. A study by Lv et al. [7] shows that expanding-ring search and random walk are better than Gnutella's query flooding. All these systems try to improve performance by limiting searches to a fraction of the population. Due to the lack of control over the contents placement, they may fail to retrieve important documents.

Distributed IR systems such as GlOSS [5] usually employ a centralized or hierarchical index to direct queries. Cuenca-Acuna and Nguyen [4] follow the conventional database selection approach but use a bloom filter to summarize each node's contents and flood the network. JXTA search [6] is a query broker system built around centralized hubs. Currently, it does not address the problem of routing queries among hubs at a large scale.

6. CONCLUSION

We propose two algorithms, pVSM and pLSI, that combine the efficiency of DHT routing with the accuracy of state-of-the-art IR algorithms to offer advanced content- and semantic-based full-text searches. Our techniques avoid the scalability problem of systems that employ centralized indexing, or index/query flooding. It also avoids the nondeterminism that is exhibited by heuristic-based approaches. To our knowledge, pSearch is the first system that allows decentralized, deterministic, and non-flooding P2P information retrieval based on contents and semantics.

To handle non-uniform distribution of documents in the semantic space, we propose a new node bootstrap process that achieves load balancing, index locality, and query locality. Furthermore, a multi-plane scheme is introduced to avoid inefficiency that can result from high dimensionality of the semantic space. We propose several uses of semantic-based indexing including semantic Publish/Subscribe, deep search in Grid, and so forth. We suggest several directions for future improvement, such as application-aware overlay.

Acknowledgements

We thank Ira Greenberg, Amy Dalal, Deqing Chen, Magnus Karlsson, Artur Andrzejak, Dejan Milojicic, Sandhya Dwarkadas, and the anonymous reviewers for their valuable comments and feedback on earlier drafts of the paper.

References

[1] M. K. Bergman. The deep web: Surfacing hidden value. http://www.brightplanet.com/deepcontent.

[2] M. Berry, Z. Drmac, and E. Jessup. Matrices, vector spaces, and information retrieval. *SIAM Review*, 41(2):335–362, 1999.

[3] A. Crespo and H. García-Molina. Routing indices for peer-to-peer systems. In *ICDCS*, July 2002.

[4] F. M. Cuenca-Acuna and T. D. Nguyen. Text-based content search and retrieval in ad hoc p2p communities. In *Proceedings of the International Workshop on Peer-to-Peer Computing*, May 2002.

[5] L. Gravano, H. García-Molina, and A. Tomasic. GlOSS: text-source discovery over the Internet. *ACM Transactions on Database Systems*, 24(2), 1999.

[6] JXTA Search. http://search.jxta.org.

[7] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker. Search and replication in unstructured peer-to-peer networks. In *ICS'02*, New York, USA, June 2002.

[8] D. S. Milojicic, V. Kalogeraki, R. Lukose, K. Nagaraja, J. Pruyne, B. Richard, S. Rollins, and Z. Xu. Peer-to-peer computing. Technical Report HPL-2002-57, HP Lab, 2002.

[9] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. In *ACM SIGCOMM'01*, August 2001.

[10] S. Rhea and J. Kubiatowicz. Probabilistic location and routing. In *INFOCOM'02*, 2002.

[11] R. Weber, H.-J. Schek, and S. Blott. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In *VLDB*, 1998.

[12] Z. Xu and Z. Zhang. Building low-maintenance expressways for p2p systems. Technical Report HPL-2002-41, HP Laboratories Palo Alto, 2002.

A Case for Associative Peer to Peer Overlays

Edith Cohen
AT&T Labs–Research
180 Park Avenue
Florham Park, NJ 07932, USA
edith@research.att.com

Amos Fiat
School of Computer Science
Tel Aviv University
Tel Aviv 69978, Israel
fiat@cs.tau.ac.il

Haim Kaplan
School of Computer Science
Tel Aviv University
Tel Aviv 69978, Israel
haimk@cs.tau.ac.il

ABSTRACT

The success of a P2P file-sharing network highly depends on the scalability and versatility of its search mechanism. Two particularly desirable search features are scope (ability to find infrequent items) and support for partial-match queries (queries that contain typos or include a subset of keywords). While centralized-index architectures (such as Napster) can support both these features, existing decentralized architectures seem to support at most one: prevailing protocols (such as Gnutella and FastTrack) support partial-match queries, but since search is unrelated to the query, they have limited scope. Distributed Hash Tables (such as CAN and CHORD) constitute another class of P2P architectures promoted by the research community. DHTs couple index location with the item's hash value and are able to provide scope but can not effectively support partial-match queries; another hurdle in DHT deployment is their tight control the overlay structure and data placement which makes them more sensitive to failures.

Associative overlays are a new class of decentralized P2P architectures. They are designed as a collection of unstructured P2P networks (based on popular architectures such as gnutella or FastTrack), and the design retains many of their appealing properties including support for partial match queries, and relative resilience to peer failures. Yet, the search process is orders of magnitude more effective in locating rare items. Our design exploits associations inherent in human selections to steer the search process to peers that are more likely to have an answer to the query.

1. INTRODUCTION

Peer-to-peer (P2P) networks have become, in a short period of time, one of the fastest growing and most popular Internet applications. As for any heavily used large distributed source of data, the effectiveness of a P2P network is largely a function of the versatility and scalability of its search mechanism.

Peer-to-peer networks came to fame with the advent of Napster [23], a centralized architecture, where the shared items of all peers are indexed in a single location. Queries were sent to the Napster Web site and results were returned after locally searching the central index; subsequent downloads were performed directly from peers. The legal issues which led to Napster's demise exposed all centralized architectures to a similar fate. Internet users and the research community subsequently turned to decentralized P2P architectures, where the search index and query processing, as well as the downloads, are distributed among peers.

Existing decentralized architectures can be coarsely partitioned into two groups [27]: unstructured, where search is *blind* (independent of the query or its context) and structured, where search is *routed*. Prevailing decentralized P2P architectures are *unstructured*. One of these architectures is Gnutella [14] under which items are only indexed by the peer that cache them; search can be resolved only by probing these peers; and peers are probed using flooding (that typically cover about 1000 nodes). The recent wave of FastTrack [33]-based P2P architectures (Morpheus, Kazaa [20, 19]) incorporate improved design that allows for more efficient downloads (simultaneous from several peers and ability to resume after failure); and improved search (by designating some peers as search-hubs *supernodes* that cache the index of others).

A feature that undoubtedly contributes to the beaming success of these decentralized unstructured architectures is support for *versatile (partial-match) queries*: Shared items typically have meta-attributes describing their type and properties (e.g., title, composer, performer); the search supports partial-match queries that populate a subset of these fields and may contain typos. Another important feature of these architectures is their "loose" structure, with each particular peer being relatively dispensable; what makes the network overlay more resilient to failures and frequent joins and disconnects. On the flip side, unstructured architectures lack an important feature which Napster had offered: While popular items (current hit movies) can be located and downloaded fairly efficiently, P2P users seemed to have lost the ability to locate less-popular items (60's hits).

A different class of architectures that was proposed and promoted by the research community is decentralized structured P2P architectures [31, 28, 29, 35, 13, 18], commonly referred to as Distributued Hash Tables (DHTs). With DHTs, peers are required to store or index certain data items, not necessarily those items that these peers have contributed or interested in. Additionally, some hashing algorithm is used to identify the peers storing a given data item. The connections between different peers are also a function of the architecture. Thus, while DHTs can be very effective for applications where queries involve unique item identifiers (e.g., P2P Web caching), they require that peers store data for the "common good"; they incur much larger overhead than "unstructured" architectures when peers fail or leave the network; and inherently, they can not efficiently support partial-match queries.

Associative overlays, proposed here, are decentralized P2P architectures, which on one hand, retain the desirable prop-

erties of prevailing unstructured architectures, including being fully decentralized with "loose" structure, and supporting partial-match queries, and on the other hand, address their biggest drawback by boosting the efficiency of locating infrequent items. Another desirable property of associative overlays is that peers are not required to store arbitrary data; peers store only what they use and their actions, including answering queries, have direct self benefit.

1.1 Associative overlays

Associative overlays defines both the formation of the overlay and the search process so that queries can be steered to peers that are more likely to have an answer. The basic premise, which we substantiate in the sequel, is that peers that would have been able to satisfy previous queries by the originating peer are more likely candidates to answer a current query.

Main ingredients in our design are *guide-rules* and *guided search*. A *guide rule* is a set of peers that satisfy some predicate; each peer can participate in a number of guide-rules, and for each guide-rule it participates in it maintains a small list of other peers belonging to the same guide rule. For each rule, the overlay induced by peers that participate in the rule forms an unstructured network and exhibits similar connectivity and expansion properties. *Guided search* retricts the propagation of queries to be within some specified guide-rules. When a peer originates a search for an item, it restricts the search propagation to a subset of its guide-rules. A peer propagating a search can only propagate it to neighbor peers within the specified rule(s).

Guide-rules should be such that peers belonging to some guide rule contain data items that are semantically similar, *e.g.*, contain documents that deal with the philosophy of science, or contain song titles by the artist formerly known as Prince. Guided search can be viewed as a middle ground between blind search used by unstructured networks and the routed search deployed by DHTs: Guided search provides a mechanism to focus the search, that is, the relevance of the peers that the query is propagated to, without tight control of the overlay and item locations. The search process within a rule mimicks search in unstructured networks, by essentially performing a blind search. On the other hand, the search strategy of the originating peer has the flexibility of deciding which guide rules, among those that the originating peer belongs to, to use for a given search.

The particular choice of the underlying set of guide-rules is constrained by both "networking" aspects, which require that the overlay has certain connectivity properties and can be formed and maintained at low cost, and the "data mining" aspects, which require that these rules meaningfully distill common interests; and thus, restricting the propagation of the query to peers within the guide rules of the originating peer yields a more focused search.

1.2 Possession rules

We focus on automatically-extracted guide rules of a very particular form, which we call *possession rules*. Each possession rule has a corresponding data item, and its predicate is the presence of the item in the local index, thus, a peer can participates in a rule only if it shares the corresponding item. Our underlying intuition, taken from extensive previous research in the Data-Mining and Text Retrieval communities ([17, 9, 10, 8, 5, 22, 16]), is that, on average, peers that share items (in particular rare items) are more likely to satisfy each other's queries than random peers. More precisely, search using possession-rules exploits presence of pairwise co-location associations between items.

Beyond the resolution of the search, possession rules provide an easy way to locate many other peers that share the item. This feature is useful for distributing the load of sending large files (parallel downloads are already practiced in FastTrack networks), or locating alternative download sites when a peer is temporarily swamped.

A feature that can allow associative overlays to strive under "selfish" peer behavior is that participation in guide-rules serves dual purpose: Supporting propagation of search through the peer but also allowing the peer to focus its own search process; A peer can participate in a rule only if it shares the corresponding item, and peers that do not participate in rules can not search better than via blind search.

1.3 The RAPIER Search Strategy

The RAPIER strategy is based on the following intuition: let the areas of interest for a given peer be A, B, C, *etc.*, randomly choose one of these areas of interest and perform a blind search amongst those peers that also have interest in this area. RAPIER (Random Possession Rule) selects a possession-rule uniformly at random from the list of previously-requested items by the querying peer.

Evidently, if there are no correlations between items, RAPIER has no advantage over blind search. We use a two-pronged evaluation of RAPIER: First, we use a simple intuitive data model (the Itemset model) to learn how the effectiveness of RAPIER grows with the amount of "structure" in the data. Second, we evaluate RAPIER on actual data, using large datasets of users accessing web sites. We obtained that RAPIER is likely to perform significantly better than blind search, in particular, it can be orders of magnitude more effective in searching for infrequent items.

2. RELATED WORK

The effectiveness of blind search can be boosted by aggregation and replication; for example, by peers summarizing the content available from other peers such as with super-peer architectures [33] and routing-indices [15] or by balancing the number of replicas against the query rates [12, 27]. The drawbacks of aggregation and proactive replication is that they are more sensitive to malicious or selfish peer behaviour and spreading of mis-labeled files. Associative overlays offer an orthogonal approach which could be combined with aggregation but does not require it.

Associative overlays address a networking challenge using an approach that is supported and motivated by extensive previous research in the field of data-mining. The constraints of the P2P setting, however, make it fundamentally different than traditional data-mining applications. A related classic data-mining problem is the *Market-basket problem*, which assumes a large number of items and customers that fill their baskets with some subset of the items. This framework applies to many domains of human activity including supermarket shopping (customers vs items matrix), library checkouts (readers vs books), document classification (word/terms vs documents matrix), Web page hyperlinks (Web pages vs Web pages), Web browsing (Users vs Web pages), and in our context, P2P networks (peers vs items). Common to all these datasets is the presence of

structure in data, namely, that these matrices are far from random. It had been long recognized that these human-selection datasets are in a sense very structured [17, 24, 6, 1]

One purpose of market-basket mining is extracting *Association rules* [2, 3]. An example of an association rule is pairs of items that are often purchased together such as "Champaign and Caviar" or "Beer and Diapers." Such rules had been used for marketing (e.g., placing Beer and Diapers next to each other in the supermarket) and recommendation systems (e.g., recommend books to customers based on previous book purchases) [7, 21, 25, 4]. A computationally challenging important subproblem is to discover association rules that have *high correlation* but *low support* (e.g., the association rule "Champaign and Caviar" that are rare purchases but are often purchased together) [11].

Similarly to these data-mining techniques, we exploit the presence of associations; but the basic difference is our highly distributed setting. Our solution does not (and can not) explicitly obtain association rules but does heavily utilize their presence. Instead of clustering peers into communities we restrict the search to communities without explicitly identifying them.

Recent proposals to exploit "interest locality" to optimize p2p search also include [30], where an existing p2p network is extended by nodes linking directly to nodes that satisfied previous queries; This basic approach does not provide a mechanism to "focus" query propagation beyond the first hop. At the other end of the spectrum, PeerSearch [32], attempts to import traditional vector space Information Retrieval (at the cost of tightly controlled DHT overlay and communication overhead).

3. MODEL AND METHODOLOGY

We represent the data present in the network by the *peer-item* matrix $D \in \{0,1\}^{n \times m}$ where n is the number of peers, m is the number of items, and $D_{ij} = 1$ if and only if peer i contains data item j.

We define the *support set* of the jth item $S_j \subseteq \{1, \ldots, n\}$, $1 \leq j \leq m$, to be

$$S_j = \{\ell | D_{\ell j} = 1\}.$$

I.e., S_j is the set of all row indices (peers) that contain data item j. The *joint support set* of two items j, k,

$$S_{jk} = S_{jk} = \{\ell | D_{\ell k} = 1 \text{and } D_{\ell j} = 1\} \ ,$$

is the set of peers that contain both items. We refer to $X_i = \{j | D_{ij} = 1\}$ (the set of items associated with peer i) as the *index of peer* i. We use the notation $s_j = |S_j|$, $s_{jk} = |S_{jk}|$, and $x_i = |X_i|$.

We define $W_i = \frac{x_i}{|D|}$, where $|D| = \sum_{i=1}^{n} x_i$ is the combined size of all indexes. An item j has low support (is "rare") if $|S_j|/n$ is small. An item has low support with respect to the weights if $\sum_{i \in S_j} W_i \ll 1$.

We view the peer-item matrix as a current instantiation of the data. We measure performance of different algorithms by treating each "1" entry, in turn, as the most recent request: For each peer i and item j such that $D_{ij} = 1$, we refer to the request that corresponds to the i, j entry as the *query* (i, j). Each query triggers a search process, which depends on the matrix D with the entry D_{ij} set to 0 and on the peer i.[1] The search process is a sequence of probes: when a peer is probed, it attempts to match the query against its local index using some algorithm. We assume that this algorithm is perfect in the sense that a query of the form (i, j) can always (and only) be resolved by a probe to peer that contains the item j.[2] The size of a search process is a random variable, and the *Expected Search Size* Ess_{ij}^{A} is the expectation of this random variable.

We compare different strategies by looking at all *queries* (peer-item pairs with $D_{ij} = 1$). We sweep a threshold on the maximum value of the Ess, and look at the cumulative fraction of queries (i, j) that have Ess_{ij} below a threshold.

3.1 Blind Search as Random Search

Following [12, 27] we model the performance of blind search in "traditional" unstructured networks using the *Random Search* model. The intuition of why this abstraction is valid is that the set of probed peers on a query in unstructured networks depends only on the overlay structure which is independent of the query or previous selections by the querying peer. Thus, on average, the effectiveness of each probe can not be better than that of probing a random peer.

When comparing Rapier to blind search and to each other we must ensure that we do not compare apples and oranges. Rapier is somewhat biased towards searching in peers with relatively many items. Thus, comparing Rapier a blind search that chooses peers uniformly at random would be unfair. One might suspect that the advantages shown experimentally are due to the choice of peers with many items, and does not reflect any other property of these algorithms. To avoid this potential pitfall, we seek to ensure that we compare these algorithms to blind search algorithms that compete on equal terms. Specifically, we consider weighted versions of the random search model where hosts have different likelihood of receiving a probe: Each peer i has a weight w_i such that $\sum_i w_i = 1$, and the likelihood that a peer is visited in a random search probe is proportional to w_i. Weighted random search is used as a benchmark for the performance of our associative search algorithms. To obtain a fair comparison, we need to consider weights that reflect the bias of the associative search algorithms towards peers with larger index sizes.

We shall consider two natural weighting schemes:

- Uniform Random Search (Urand) where all peers are equally likely to be probed ($w_i = 1/n$). This models pure blind search.

- Proportional Random Search (Prand), where the likelihood that a peer is probed is proportional to the size of its index $w_i = W_i \propto \sum_{j=1}^{m} D_{ij}$. This models blind search biased towards peers with larger indices. We will show that this bias is exactly equal to the bias introduced by Rapier and thus differences in performance between the two cannot be due to this bias.

[1] Note that the search sequence does not depend on j, as query properties (such as meta-data terms) are not used to determine where to search. It is used only as a stopping condition. See the introduction and conclusion sections for a discussion on this issue.

[2] this simplification is justified as the matching issue of queries to appropriate items is present with other architectures and is orthogonal to the core of our contribution.

With weighted random search, the size of the search for a query (i, j) is a Geometric random variable. The Ess is the mean of this random variable.

A weighted random search for item j by peer i has likelihood of success in each probe $p_{ij} = \frac{\sum_{k \neq i} w_k D_{kj}}{1 - w_i}$. and thus for any weighted random search algorithm A $\text{Ess}_{ij}^{A} = p_{ij}^{-1} = \frac{1 - w_i}{\sum_{k \neq i} w_k D_{kj}}$. (The search is performed on all peers excluding peer i).

Thus, a URAND search for item j by peer i has

$$\text{Ess}_{ij}^{\text{URAND}} = \frac{n-1}{\sum_{k \neq i} D_{kj}} \; ; \tag{1}$$

and a PRAND search has

$$\text{Ess}_{ij}^{\text{PRAND}} = \frac{1 - W_i}{\sum_{k \neq i} W_k D_{kj}} \; . \tag{2}$$

4. POSSESSION-RULE OVERLAYS

Guide rules connectivity. : Peers that participate in the same guide-rule form a sub-overlay that resembles a "traditional" unstructured network. Thus, each guide-rule constitutes a sub-overlay, and these sub-overlays are generally overlapping. Search is conducted using guide rules. Similarly to search in traditional unstructured networks, it is propagated from peer to neighbors but the propagation is only to peers belonging to the selected guide rule. Each guide-rule sub-overlay needs to have the form of a "traditional" unstructured overlay. For each guide-rule it is associated with, a peer needs to remember a small list of peers which belong to the guide rule; and neighbors should be such that guided-search reaches a large number of peers. The specifics can vary from a Gnutella-like design where each peer has few viable neighbors (Typical Gnutella number is 2-4) and many other peers can be reached through them, to a FastTrack-like design where search is facilitated through a core network of supernodes (in our case supernodes are associated with guide-rules). The specifics are orthogonal to our basic approach, we only need to make sure that our selected guide rules are such that the underlying unstructured network can form.

Search strategy. : A search strategy defines a search process as a sequence of guide rule probes. An example of a strategy is "search 100 peers that have item A and 200 peers that have item B, if this is unsuccessful, then search 400 more that have item A and 50 peers with item C, . . . "

Our general expectation is that the total number of guide rules may be large, but a typical peer uses a bounded number of rules. The applicability of a specific set of guide-rules depends on the implementability of the connectivity requirement. This requirement has two parts, first there should be a simple mechanism to locate a peer (and through it other peers) that belong to the same guide rule. It is also a requirement that this selection should result in large connected components. Below we argue that possession-rules fill the first part. As for large components, practice shows that simple neighbor selection strategies of current P2P implementation result in large connected components, and thus, we argue that selections within a guide-rule are likely to result in large components. (Random connections are known to yield large components and apparently actual selections are "sufficiently random" to obtain this property). In any case, the same issue of obtaining large components exists in traditional unstructured architectures and the connectivity algorithms deployed in these networks can be adapted to our context. There is thus no need to re-tackle this issue.

The possession-rule overlay is self-boosting: If peer-A conducts a search for item i that is resolved by peer-B then it is able to obtain through peer-B a list of other peers that index item i. As a result, each peer has a neighbor list which is an array of (item,peer) pairs for (most) items in its index. Thus, for possession rules, the construction of the overlay is symbiotic with the search process. There is seemingly a major issue in that a peer in a guide-rule network may keep track of many other peers, proportional to the number of guide rules it belongs to. Even when bounding the number of guide-rules a peer participates in, the number of neighbors is considerably larger than in existing architectures. This is in contradiction to the philosophy used by existing P2P architectures, which promotes having a small number of neighbors. We argue, however, that there is no reason for guided search to abide by this rule whereas there are clear reasons for other P2P architectures to keep it. Unlike DHTs, the update cost of a neighbor going offline is minimal; we may discover it when trying to search through these peers and may then remove them from our list following one or more unsuccessful tries; replacements are easy to find if at least some of the guide-rule neighbors are active. It is also advantageous for search in unstructured network to have a small fan-out, but we achieve that since each guide-rule sub-overlay has a low degree.

In the sequel, we assume that our network is a possession-rule overaly. Each sub-overlay resembles an unstructured network and we use the model of random search used in [12, 27] to capture the performance of search within a rule.

5. RAPIER SEARCH STRATEGY

RAPIER is a simple search strategy that uses possession-rules overlay. The strategy repeats the following until search is successful (or search size limit is exceeded):

1. Choose a random item from your index.
2. Perform a blind search on the possession-rule for the item to some predetermined depth.

The main parameter we look at is the size of the search which is the total number of peers probed. We model RAPIER search by the following process: For a query for item j issued by peer i, a column k is drawn uniformly from $X_i \setminus \{j\}$ (the index of i excluding j). Then a peer r is drawn uniformly from $S_k \setminus \{i\}$. The search is successful iff $D_{rj} = 1$.

Thus, the likelihood of success for RAPIER per step is

$$p_{ij} = (x_i - 1)^{-1} \sum_{k \in X_i \setminus \{j\}} \frac{s_{kj} - 1}{n - 1} \; .$$

and thus

$$\text{Ess}_{ij}^{\text{RAPIER}} = \frac{(x_i - 1)(n-1)}{\sum_{k \in X_i \setminus \{j\}} (s_{kj} - 1)} \; . \tag{3}$$

As discussed earlier, search strategies may differ to the extent that they utilize peers of different index sizes. RAPIER, in particular, is more likely to probe peers with larger indices, since such peers share items with a larger number of other peers. We can show that averaged over queries, the

likelihood that a peer is probed under RAPIER is equal to W_i (its likelihood to be probed under PRAND). Thus, it is fair to use PRAND as a benchmark for RAPIER since *per-search*, they have the same bias towards peers with larger index sizes. We compare the performance of the two algorithms on the Itemset model and using simulations.

6. THE ITEMSETS MODEL

Frequency and size distributions of items and peers are reasonably-well understood and are typically modeled by Zipf-like distributions. But even though these distributions capture enough aspects of the data to evaluate the performance of blind search, they do not capture correlations that are necessary for evaluating associative search. Models which capture correlations present in market-basket data and Web hyperlink structure had been proposed [3, 25, 26]. We use one such model, the *Itemsets model* (which resembles models in [3, 25]), to convey intuition why and when we anticipate RAPIER to perform well.

The Itemsets model partitions items into N "interest areas" (which we refer to as *itemsets*). Each peer belongs to some subset of the itemsets, and contains f fraction of items (picked uniformly at random) in each itemset it belongs to.

Items in different itemsets are generally not correlated, and items in the same itemset are correlated. Our expectation is that if peers belong to many itemsets (at the extreme, all peers have all itemsets), there is no advantage for RAPIER over PRAND. When peers belong a to a small number of itemsets we expect RAPIER to perform better; and we expect this advantage to increase as the number of itemsets decreases. We formalize this intuition below.

Suppose that each peer belongs to exactly k itemsets[3], and these itemsets are independent or positively correlated, that is, if $p(x)$ is the fraction of peers belonging to itemset x, and $p(x \cap y)$ is the fraction of peers belonging both to itemsets x and y, then $p(x \cap y) \geq p(x)p(y)$. Let $x(\ell)$ be the itemset of item ℓ and let $p(x(\ell))$ be the fraction of the peers that contain itemset $x(\ell)$. Consider a query made to an item ℓ. Then the success probability of a PRAND probe is $R_\ell = fp(x(\ell))$; and the success probability of RAPIER probe is $C_\ell = \frac{f}{k}(1 + (k-1)p(x(\ell)))$. It follows that the ratio of the ESS under PRAND to the ESS under RAPIER for item ℓ by any peer is $\frac{1}{kp(x(\ell))} + \frac{k-1}{k}$. Since $p(x(\ell)) \leq 1$, RAPIER is always at least as effective as PRAND. When $p(x(\ell)) \ll 1/k$, RAPIER is much more efficient than PRAND. This simplistic model provides some intuition to when RAPIER is more effective than PRAND: RAPIER benefits, when users interests are more "focused" (small k) and for items in rare itemsets (small $p(x(\ell))$).

7. SIMULATION RESULTS

As large scale peer-item data is not available publicly, we opted to use a different source of similarly-structured ("market-basket") data. We used Boeing [34] Web proxy logs of a lower-level proxies serving end users and extracted the matrix of users versus hostnames. In the sequel, we refer to users as *peers* and to hostnames as *items*. The resulting data matrices (for each day of the Boeing logs) had about 57K peers, 45K items, and 115K pairs.

[3]Similar results would hold when we assume that each peer belongs to at most k itemsets

As is typical with such data, we observed high skew in both the size of the index peers have and the support-size of items (large fraction of peers having small index sizes and large fraction of items being present at a small fraction of peers. About 60% of queries are issued to items whose support is over 0.01 fraction of peers; so considerable fraction (40%) of queries target unpopular items.

We evaluated the performance of 3 search strategies:

Algorithm	ESS computed according to
URAND	Equation 1
PRAND	Equation 2
RAPIER	Equation 3

The results of the simulations are shown in Figures 1. The figure shows a cumulative fraction of queries that have ESS below a certain threshold. They show the performance for items across support levels and also focus on items that have lower support (occur in the index of at most 10^{-2}-10^{-4} of peers). The figures show that URAND is the worst performer. The ESS of URAND on an item is the inverse of its fraction of peers that index it, thus, when focusing on items occurring in at most 10^{-4} of users, the respective ESS is over 10K, and the URAND curve coincides with the x-axis. The PRAND strategy that prefers peers with larger index sizes manages to see more items in each probe and performs considerably better than URAND, across items of different support levels.

We observe that RAPIER, which has the same bias towards peers with larger index as PRAND, outperform PRAND; moreover, the performance gap is significantly more pronounced for items with low support. This indicates strong presence of the semantic structure RAPIER is designed to exploit; and also emphasizes the qualitative difference between RAPIER and aggregation-based search startegies.

For a typical Gnutella search size, estimated to cover about 1000 peers, the simulations on the Boeing dataset show that RAPIER covers 52% of queries made to items that are present on at most 10^{-4} fraction of peers, whereas PRAND covers only 14% of queries. Out of all queries, RAPIER covers 95% and PRAND covers 90%. On a smaller search size of a 100, RAPIER and PRAND, respectively, cover 30% and 1.3% of items with support below 10^{-4} fraction of peers, and cover 90% and 80% of all items. For search sizes where PRAND covers most queries, RAPIER obtains about half the failure rate of PRAND.

8. CONCLUSION

Associative overlays retain the advantages of unstructured architectures (such as gnutella and FastTrack); including relative insensitivity to peer failures and support for partial-match queries; but can offer orders of magnitude improvement in the scalability of locating infrequent items. Our design exploits presence of associations in the underlying data. Such associations were previously exploited for Web search, Data-mining, and collaborative filtering applications, but the techniques were not portable to the P2P setting which requires simple, resilient, and fully decentralised protocols. Our approach maintains the essense of these techniques while striking a balance with the challenges of the P2P setting.

We argued that RAPIER, the simplest search strategy on possession-rule overlays, can dramatically increase the effectiveneess of search for rare items over that of plain unstructured networks. It is likely that better search performance on possession-rule overlays can be achieved by prefer-

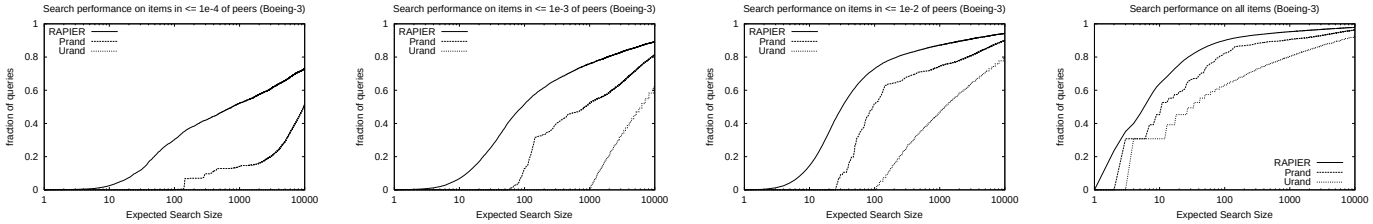


Figure 1: Search performance on items present in (1e-4, 1e-3, 1e-2, all) fraction of peers (Boeing-Day3 log).

ing rules that correspond to recently acquired items or rules where the meta data of the corresponding items is more related to the query terms. It is also possible to design more refined search strategies that acount for relations between guide rules.

9. REFERENCES

[1] D. Achlioptas, A. Fiat, A. Karlin, and F. McSherry. Spectral analysis of data. In *Symposium on Theoretical Computer Science, pp. 619-626*, 2001.

[2] R. Agrawal, T. Imielinski, and A. N. Swami. Mining association rules between sets of items in large databases. In P. Buneman and S. Jajodia, editors, *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, pages 207–216, Washington, D.C., 26–28 1993.

[3] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In J. B. Bocca, M. Jarke, and C. Zaniolo, editors, *Proc. 20th Int. Conf. Very Large Data Bases, VLDB*, pages 487–499. Morgan Kaufmann, 1994.

[4] Y. Azar, A. Fiat, A. R. Karlin, F. McSherry, and J. Saia. Web search through hub synthesis. In *Symposium on Foundations of Computer Science*, 2001.

[5] K. Bharat and M. R. Henzinger. Improved algorithms for topic distillation in a hyperlinked environment. In *Research and Development in Information Retrieval*, pages 104–111, 1998.

[6] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. In *Proceedings of the WWW7 Conference*, 1998.

[7] ACM SIGGROUP resource page on collaborative filtering. `http://www.acm.org/siggroup/collab.html`.

[8] S. Chakrabarti, B. Dom, D. Gibson, J. Kleinberg, S.R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. Hypersearching the web. *Scientific American*, June 1999.

[9] S. Chakrabarti, B. Dom, D. Gibson, R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. Spectral filtering for resource discovery, 1998.

[10] S. Chakrabarti, B. Dom, P. Raghavan, S. Rajagopalan, D. Gibson, and J. Kleinberg. Automatic resource compilation by analyzing hyperlink structure and associated text. *Computer Networks and ISDN Systems*, 30(1–7):65–74, 1998.

[11] E. Cohen, M. Datar, S. Fujiwara, A. Gionis, P. Indyk, R. Motwani, J. D. Ullman, and C. Yang. Finding interesting associations without support pruning. *Knowledge and Data Engineering*, 13(1):64–78, 2001.

[12] E. Cohen and S. Shenker. Replication strategies in unstructured peer-to-peer networks. In *Proceedings of the ACM SIGCOMM'02 Conference*, 2002.

[13] Open Source Community. The free network project - rewiring the internet. In *http://freenet.sourceforge.net/*, 2001.

[14] Open Source Community. Gnutella. In *http://gnutella.wego.com/*, 2001.

[15] A. Crespo and H. Garcia-Molina. Routing indices for peer-to-peer systems. In *ICDE*, 2002.

[16] J. Dean and M. R. Henzinger. Finding related pages in the world wide web. *WWW8 / Computer Networks*, 31(11-16):1467–1479, 1999.

[17] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. Indexing by latent semantic indexing. *Journal of the American Society for Information Science*, 41(6):391–407, September 1990.

[18] A. Fiat and J. Saia. Censorship resistant peer-to-peer content addressable networks. In *Symposium on Discrete Algorithms*, 2001.

[19] KaZaA file sharing network. KaZaA. In *http://www.kazaa.com/*, 2002.

[20] Morpheus file sharing system. Morpheus. In *http://www.musiccity.com/*, 2002.

[21] D. Goldberg, D. Nichols, Oki B. M., and D. Terry. Using collaborative filtering to weave an information tapestry. In *Communications of the ACM, 35(12)*, pages 51–60, 1992.

[22] T. Hofmann. Probabilistic latent semantic indexing. In *Research and Development in Information Retrieval*, pages 50–57, 1999.

[23] Napster Inc. The napster homepage. In *http://www.napster.com/*, 2001.

[24] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999.

[25] S. R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. Recommendation systems: A probabilistic analysis. In *IEEE Symposium on Foundations of Computer Science*, pages 664–673, 1998.

[26] S. R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. Extracting large-scale knowledge bases from the web. In *The VLDB Journal*, pages 639–650, 1999.

[27] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker. Search and replication in unstructured peer-to-peer networks. In *Proceedings of the 16th annual ACM International Conference on supercomputing*, 2002.

[28] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. In *Proceedings of SIGCOMM'2001*, 2001.

[29] A. Rowstron and P. Druschel. Storage management and caching in past, a large-scale, persistent peer-to-peer storage utility. In *Proceedings of SOSP'01*, 2001.

[30] K. Sripanidkulchai, B. Maggs, and H. Zhang. Enabling efficient content location and retrieval in peer-to-peer systems by exploiting locality in interests. *ACM SIGCOMM Computer Communication Review*, 32(1), January 2002.

[31] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of SIGCOMM'2001*, 2001.

[32] C. Tang, Z. Xu, and M. Mahalingam. Peersearch: Efficient information retrieval in peer-to-peer networks. In *Proceedings of HotNets-I, ACM SIGCOMM*, 2002.

[33] FastTrack Peer-to-Peer technology company. FastTrack. In *http://www.fasttrack.nu/*, 2001.

[34] Web traces and logs. `http://www.web-caching.com/traces-logs.html`.

[35] B. Y. Zhao, J. Kubiatowicz, and A. Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Technical Report UCB/CSD-01-1141, University of California at Berkeley, Computer Science Dept, 2001.

Efficient Topology-Aware Overlay Network

Marcel Waldvogel
mwl@zurich.ibm.com

Roberto Rinaldi
rob_rinaldi@virgilio.it

IBM Research
Zurich Research Laboratory
Säumerstrasse 4 / Postfach
8803 Rüschlikon, Switzerland

ABSTRACT

Peer-to-peer (P2P) networking has become a household word in the past few years, being marketed as a work-around for server scalability problems and "wonder drug" to achieve resilience. Current widely-used P2P networks rely on central directory servers or massive message flooding, clearly not scalable solutions. Distributed Hash Tables (DHT) are expected to eliminate flooding and central servers, but can require many long-haul message deliveries. We introduce Mithos, an content-addressable overlay network that only uses minimal routing information and is directly suitable as an underlay network for P2P systems, both using traditional and DHT addressing. Unlike other schemes, it also efficiently provides locality-aware connectivity, thereby ensuring that a message reaches its destination with minimal overhead. Mithos provides for highly efficient forwarding, making it suitable for use in high-throughput applications. Paired with its ability to have addresses directly mapped into a subspace of the IPv6 address space, it provides a potential candidate for native deployment. Additionally, Mithos can be used to support third-party triangulation to quickly select a close-by replica of data or services.

1. INTRODUCTION

The computing world is experiencing a transition from fixed servers and stationary desktop PCs to connected information appliances and ubiquitous connectivity, profoundly changing the way we use information. With cellular data communication, Bluetooth, and IEEE 802.11b (WiFi), the need for a global system that supports these new communication patterns becomes more pressing day by day. Two main patterns can be identified: First, Internet routing table size is surging, second, Internet protocol (IP) forwarding is still a bottleneck in routers, and third, direct serverless communication is gaining importance.

Routing Table Size. The ever increasing size of the Internet routing tables calls for new ways in network protocols. Although the introduction of Classless Inter-Domain Routing (CIDR) [1] enabled large-scale aggregation of routing information and thus provided a respite in the exponential growth of routing and forwarding tables for several years, the expansion has resumed in the first half of 2001 with full strength. Among the reasons given for the increased growth rates are the exhausting of preallocated address ranges, proliferation of always-on connected devices, and, probably most significantly, the tendency for businesses and even small Internet Service Providers (ISPs) to become multi-homed. This fact of being connected to multiple upstream providers breaks the hierarchy model behind CIDR, which is necessary for its aggregation to be efficient.

Forwarding Lookups. In the early Internet days, packet forwarding was done by a single hash or index table lookup. With the introduction of CIDR to keep routing table size under control, a more complex lookup was required, performing a longest prefix match, which has long been an obstacle to building fast routers serving high-speed links. Novel algorithms [2–4] as well as additional protocol layers such as MPLS [5] have reduced the cost of prefix matching. Any new network design aiming for high data rates should provide for inexpensive lookups.

Symmetric, Serverless Communication. While services such as Napster brought publicity to the term peer-to-peer (P2P), serverless communication only started becoming popular when Napster's demise became a possibility. The events of September 11, 2001, have further shown that centralized servers and thus single points of failure should be avoided when system reliability and availability are business-critical. Serverless systems of the first generation heavily relied on flooding as the prime mechanism to query the distributed directory and to support connectivity when network components become unavailable. The second generation being designed now is based on distributed hash tables (DHTs) to allow direct addressing once the ID of the resource, such as document or service, is known.

Although many theoretical schemes for minimizing routing information have been proposed and many designs for DHTs have recently become prominent discussion topics, we are unaware of any practical and efficient system combining both. In this paper, we introduce Mithos, a novel mechanism that combines both, and provides additional benefits, such as its ability to use IPv6 as a native transport mechanism and its support for third-party triangulation.

Unlike other systems that map Internet topology to Cartesian coordinates [6, 7], Mithos, in full P2P spirit, uses *every node* in the entire network also as a topology landmark. This helps achieve accuracy and efficiency without the overhead of numerous dimensions or full-mesh probing of all landmarks. Instead, directed incremental probing is used to find a near-optimal placement, as will be explained below.

In Mithos, routing table size is minimized because every node only needs to know its direct neighbors; transitive routing enables messages to reach any destination nevertheless. To achieve this, Mithos employs a novel approach to routing in multi-dimensional irregular meshes, which is key to achieving minimum routing table size while guaranteeing connectivity.

The remainder of the paper is organized as follows. Section 2 introduces and describes the concepts behind Mithos. Section 3 presents early results from our simulation environment. Related work is discussed in Section 4, and conclusions are drawn in Section 5.

2. MITHOS DESIGN

The basic idea of Mithos is to embed the network into a multi-dimensional space, with every node being assigned a unique coordinate in this space. This is similar to interconnects used in many high-performance parallel computers, enabling optimal global routing with simple knowledge of the local coordinate gradients, i.e., which links lead to higher/lower coordinates in which dimensions. Unlike parallel computers, however, the mesh used for Mithos connectivity is not regular, in order to accommodate dynamic membership as well as to represent locality.

These goals are established for every new node in a three-phase process:

1. Finding close-by nodes and establishing a neighborhood
2. Assigning an ID to the newcomer based on this neighborhood
3. Establishing links with the neighborhood

The individual phases are discussed in more detail below.

2.1 Finding Neighbors

To ensure that neighbors in the overlay network are also close in the "underlay" network, a distance metric and a location process need to be defined. We chose network delay between two nodes as metric for measuring distances, but any metric establishing geometry-like foundations would

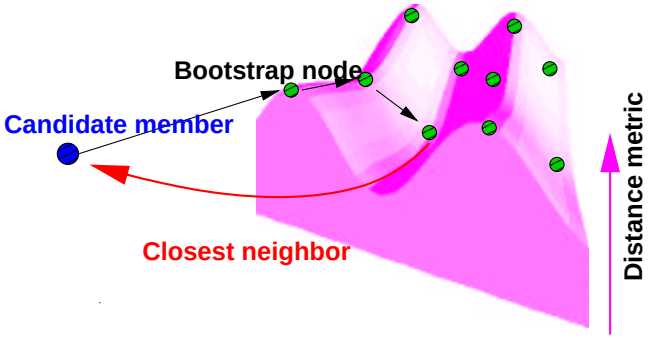


Figure 1: Finding neighbors

be suitable, including any metrics typically used in routing protocols, independent of their Quality-of-Service (QoS) awareness. Examples include physical distance, monetary link cost, or the bandwidth a TCP-compliant stream would achieve.[1] Independent of the metric used, the value is referred to as *distance* below.

It is well known that connectivity and connection parameters are not necessarily symmetric or transitive in the Internet, especially when multiple autonomous systems (AS) are involved [8]. Nevertheless, these metrics provide a reasonable basis for an overlay network. When setting up a sufficiently dense overlay network whose goal is to minimize these connection parameters on a per-link basis, the overlay will adapt itself, trying to get optimal service from the underlay.

When searching for neighbors, the natural choice would be to perform an expanding ring search using a multicast mechanism [9]. Although the protocols were defined more than a decade [10], multicast is still only available as an experimental platform in the Internet, if at all. Therefore, the neighborhood location process has to revert to using unicast.

For bootstrapping, Mithos requires a candidate member to know how to contact (at least) one of the existing members. A nonempty subset of these members is used as the first set of candidate neighbors. Then, knowledge from within the overlay network is used to locate the actual neighborhood as follows. Each candidate neighbor is first asked for its direct neighbors, then these neighbors are probed for their distance according to the metric chosen for the overlay system. The best node is then used as the new candidate neighbor. This process is iterated until no further improvement can be achieved, effectively following the distance gradient (Figure 1).

As this process is prone to terminate at a local instead of the global minimum, local minima must be recognized and avoided. For Mithos, this is currently done by probing all nodes that are two steps away from the current minimum

[1]When setting up a system, care should be taken that the metric chosen is relatively stable for the duration of the P2P network.

before giving up. If a better candidate neighbor is found, the iterative process continues.

2.2 ID Assignment

Now that one of its neighbors has been selected, it is necessary to actually assign an ID to the candidate member. This ID selection process is critical, as an inappropriate assignment will eventually create many local minima, preventing an efficient neighborhood location in the future.

Mithos uses the distances measured during the last step of neighborhood establishment as a basis for ID assignment. The two closest nodes found in the process, their neighbors, and the corresponding distances are used in this computation, which requires no further communication.

For ID calculation, virtual springs are established between the candidate member and its fixed neighbors. The tension of each spring is set to be inversely proportional to the distance measured. Then this virtual equivalent of a physical system is allowed to settle, achieving the minimum energy state. This minimum energy location of the candidate node in the multidimensional space is directly used for its ID.

Now that an ID has been established, distances are *computed* in ID space, no longer requiring *measurements* (and thus message exchanges) according to the distance metric.

2.3 Linking Options

The final step is the establishment of peering relationships between neighbors. To evaluate the possible options for interconnecting neighbors, we established the following criteria:

1. Minimum routing table size;
2. efficient connectivity, full reachability; and
3. fast and simple forwarding algorithm.

These goals would be readily achieved by the strongly regular hypercube or hypertorus interconnect used in many parallel computers. In the presence of network dynamics, the regularity requirement would need to be significantly weakened. Our criterion of maintaining locality between neighbors completely breaks the dynamic supercomputer analogy. Furthermore, locality can lead to some local clustering effects, which need to be dealt with. Alternatives to rectangular connectivity in dynamic, locality-preserving environments are described and evaluated below.

Closest to axis. Along each axis in each direction, find a node that is closest to this axis and establish a link. Then, use the traditional hypertorus forwarding mechanism when delivering messages.

Quadrant-based. Each node establishes a link to the closest neighbor in each quadrant.[2] When forwarding, the next hop is chosen as the neighbor in the same quadrant as the final destination. This can done by computing the difference vector between the current node and the destination, and using the bit vector of the resulting d sign bits (one per dimension) as an index into the next-hop table.

[2]We use the term "quadrant" as a generic term, even when the number of dimensions, d, does not equal 2. All quadrants are determined relative to the current node.

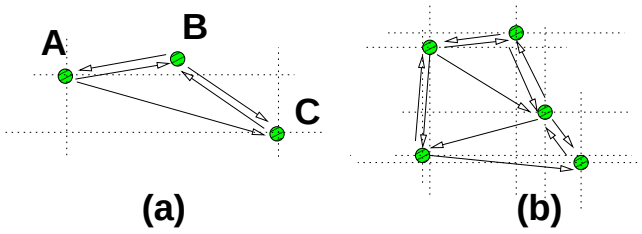


Figure 2: Example quadrant links in 2-space

Rectangular subdivision. Each node is assigned an enclosing axis-parallel multi-dimensional rectangle [11]. Forwarding is done to the rectangle abutting at the point where the vector to the destination intersects with the current node's rectangle boundary.

Delaunay triangulation. Establish links according to a Delaunay triangulation of the nodes. Forward analogous to the previous whose vector is angularly closest to the destination vector.

All of these approaches *typically* achieve small routing tables, although in the worst case (for all but the *axis* mechanism) a single node could have all other nodes in the system as neighbors.

The connectivity is efficient, except when using *closest to axis*, which fails to locate off-axis nodes closer than the next on-axis node.

Forwarding lookups are optimal for the *quadrants* solution, as the final next-hop decision can be made by a simple indexed array access, following a per-dimension subtraction and concentration of sign bits. Many processor architectures offer support for SIMD arithmetic or aggregation of values, as they are easy to implement. Forwarding is still very good for the *axis* method, but as this method is unable to find all nodes without the aid of another algorithm, we consider it impractical, *Rectangles* and *Delaunay* base their decisions on angular calculations and comparisons, requiring expensive multiplications and multidimensional range searches.

We therefore decided to use a *quadrant-based* mechanism, as it easily fulfilled all the criteria.

2.4 Establishing Quadrant Links

Before describing how to achieve quadrant-based links, we first evaluate some of their properties. Figure 2 shows two excerpts of networks situated in 2-space. Looking at Figure 2 (a), even though A has C as its closest southeast neighbor, C does not consider A as its closest northwest neighbor, resulting in asymmetric links. Fortunately, this asymmetry

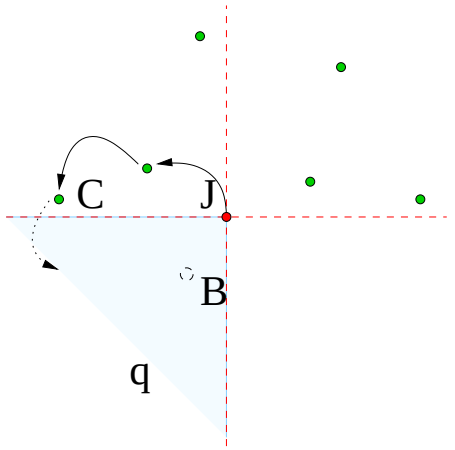


Figure 3: Finding neighbors in all quadrants

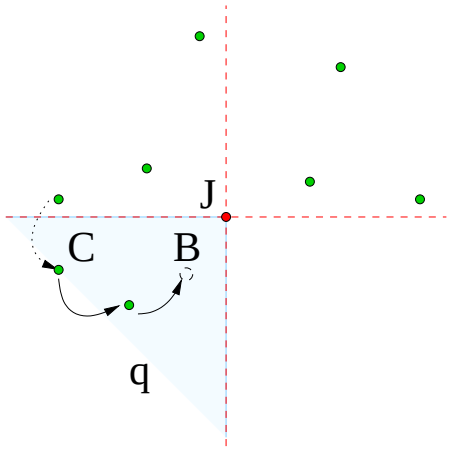


Figure 4: Finding the best neighbor in a quadrant

has no functional drawbacks during forwarding, as all nodes can still be reached efficiently. However, it needs to be taken into account when establishing the links. To simplify the description, the routing and link establishment process establishes bidirectional links, even though some of them will be used only unidirectionally when forwarding. Thus, the forwarding database remains minimum.

When the joining node J has established its ID, the sum of neighbors that helped it establish its ID may have no information about the best neighbor in all of J's quadrants. This can be because J's final position is out of range of the nodes' knowledge, or due to the asymmetry of the routing (cf. Figure 2). Furhermore, even though J might know of a node in each quadrant, this does not necessarily imply that this node is also the node closest to J. Therefore, J needs to identify the best neighbors in the region. The mechanism to achieve this is based on ideas similar to the perimeter walk used in Greedy Perimeter Stateless Routing (GPSR) [12], but has been extended to higher dimensions.

Now that a complete neighborhood has been established, it must be ensured that links are established to the closest neighbors, in order to guarantee correct forwarding operation. Thus the second phase tries to locate a closer neighbor by starting at the known neighbor and scanning towards all quadrant borders (Figure 4).

This second phase is an even further generalization of GPSR [12]. It currently uses parallel path processing, which we expect can be optimized further by taking into account further geometric properties of the node relationships. Our early simulations have revealed that in the vast majority of cases, the best neighbors are already known from the merge step. The process is described in more detail in [13].

Serialization of multiple join events is only necessary if they involve the same neighborhood. As the steps requiring serialization all operate only on highly local areas with short distances, serializing them is not expected to become a bottleneck, although we are looking at ways to improve that.

2.5 Priming the Overlay Network

Starting the network from a single node using the mechanisms described above can lead to a very uneven utilization of the available space. To initialize the constants and provide enough initial points required for the spring forces algorithm, the network is primed with a small number of nodes appropriately distributed throughout the space the overlay network should span. These initial nodes are preferentially selected from early nodes interested in joining the system, but we envision that appropriate public landmarks could also be used to bootstrap the system.

3. RESULTS

Preliminary results indicate that the above algorithms work very well. Figure 5 shows the quality of the minimum-finding algorithm. Despite its simple heuristics, the results are very encouraging. The test network consisted of 10,000 nodes in the underlay network (generated using the INET topology generator[3]) and 1000 nodes in the four-dimensional overlay network. About half of the nodes are optimally placed and more than 90% of the nodes are less than a factor of 5 in delay from their minimum. Further analysis reveals that this is often due to the small absolute delay.

Figure 6 compares the overhead of end-to-end path lengths under different numbers of dimensions (the same underlay network was used, but this time, only 200 nodes are placed in the overlay network for simulation efficiency). As can be seen, already at four dimensions, more than 97% of the paths are less than a factor of 3 from optimal. This is in contrast to non-P2P localization algorithms which require more dimensions and do not provide an efficient addressing scheme at the same time.

We expect better placement heuristics to further improve these results at potentially even further savings during node placement. More of our early results can be found in [14].

[3]Available from `http://topology.eecs.umich.edu/inet/`.

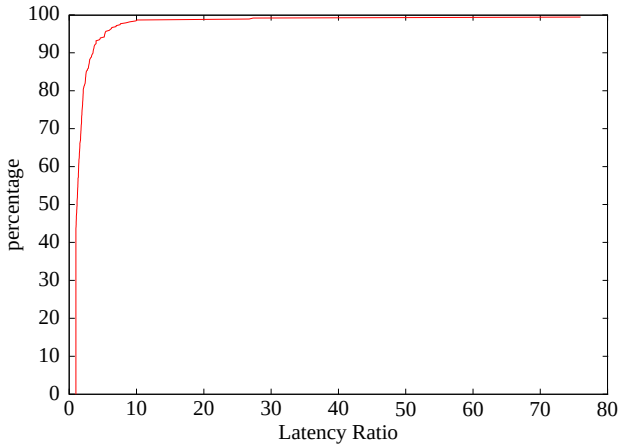


Figure 5: Latency ratio from the local/global minimum for each joining node (CDF)

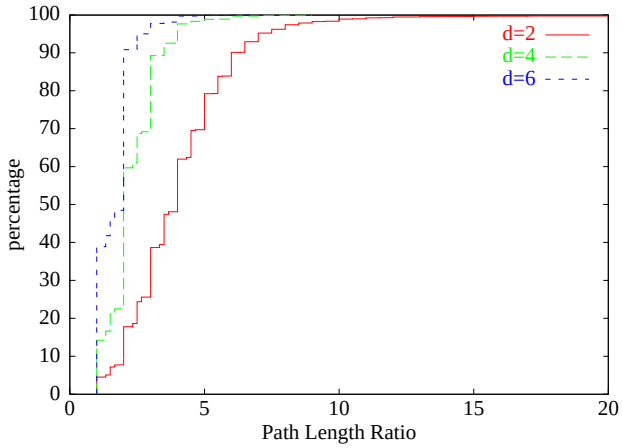


Figure 6: Path length ratios with 2, 4, and 6 dimensions (CDF)

4. RELATED WORK

Cartesian mapping of the Internet has been a topic in the papers by Francis et al. [6] and, more recently, by Ng and Zhang [7] use landmarks and measurements for triangulation. These two systems rely on a small number of landmarks to provide their measurements. For the system to work, there is thus a critical need for a reliable infrastructure offering these landmarks at high availability. Temporary failure or unreachability of a subset of these nodes will make it hard to compare the proximity of new nodes.

A series of scalable overlay networks have recently sprung to life, such as CAN [15], Chord [16], Pastry [17], and Tapestry [18], all offering a DHT service. The respective locality properties of CAN, Chord, and Pastry are discussed below, separated into *geographic layout* and *proximity forwarding,* categories adapted from Castro et al. [19].[4]

CAN is based on connectivity in a d-dimensional space which is subdivided into hypercuboids, which are logically connected along touching surfaces. Initially, CAN's locality was based on proximity forwarding: each node keeps track of the quality of the neighbor links, measured by the ratio of *forwarding progress* (in the d-dimensional space) vs. round-trip time to that neighbor. Later, it was refined to use layout as well, where it adopted a *binning* scheme [20] to determine neighborhood during node placement. This binning scheme is based upon ranking the relative distances to a given set of landmarks as well as the absolute distances, the latter having been heavily quantized before being used for comparisons. A newly joining node is then placed close to an existing node with a similar landmark triangulation.

Chord extends on the ideas of *interval routing* [21] by providing for dynamic behavior and proximity forwarding. All nodes are arranged on a conceptual circle, with each node having forwarding fingers (chords) to various other places along the circle. These fingers are constantly refined to point to nodes in close proximity, which can lead to significant improvements in forwarding.

Pastry (and Tapestry) routing is similar to radix tries. A message reaches a destination by continuously following to a node with a longer shared prefix between the destination and next-hop IDs. Despite being based on a tree structure, there is no central point of failure, as every participant is both a root, a leaf, and a set of interior nodes in a cleverly interwoven set of tries. Again, proximity forwarding is chosen to take advantage of locality. Among the nodes eligible as children of a particular tree node, the closest node known is picked. According to Castro et al. [19], this allows for a child choice from a much larger set than possible with Chord, resulting in shorter paths.

Among the DHTs, CAN is closest to Mithos in terms of features provided, but uses an entirely different approach; nevertheless, we expect the reliance on a small subset of landmarks, the coarse binning scheme, and the weak integration between layout and routing to provide a performance disadvantage.

5. CONCLUSIONS AND FUTURE WORK

By having all nodes in the P2P overlay network provide neighborhood location service through a directed, efficient search, we are able to create an overlay network whose connectivity is close to the optimum achievable with full topology knowledge. In contrast to other approaches, Mithos does not require full topology knowledge, even the forwarding and routing information is minimum and can be used in a highly efficient manner. At the same time, Mithos provides a close conceptual integration between geographic layout and proximity routing, as well as a powerful addressing scheme directly suitable for use in DHTs.

Another key distinguishing factor to both overlay networks as well as the underlying Internet protocol (IP) is the efficiency of the forwarding lookup: its next-hop calculation requires only a few fast processor instructions (or simple

[4]Tapestry does not directly take advantage of locality itself, due to the strong similarity of the routing mechanism to Pastry, the observations discussed below equally apply to both.

hardware) and a single indexed memory lookup, significantly faster than comparable or even less feature-rich systems. We believe that such addresses could be directly used in a native, dedicated subspace of the IP version 6 address space [22] to provide efficient addressing and forwarding, e.g., by using six dimensions of 16 bit resolution each.

In the future, we will investigate the dynamic behavior of the network and how to handle asymmetric underlay failures. We also plan to employ metrics obtained from real networks, including metrics other than pure delay. Further topics include optimizations of the "local minimum" and "spring forces" heuristics, as well as evaluating "asymmetric" dimensions, such as local and non-wrapping dimensions, which we expect to be useful when dealing with non-uniform address space usage, but also will provide significant gains for improving locality.

6. REFERENCES

[1] Vince Fuller, Tony Li, Jessica Yu, and Kannan Varadhan. Classless Inter-Domain Routing (CIDR): An address assignment and aggregation strategy. Internet RFC 1519, September 1993.

[2] Mikael Degermark, Andrej Brodnik, Svante Carlsson, and Stephen Pink. Small forwarding tables for fast routing lookups. In *Proceedings of ACM SIGCOMM*, pages 3–14, September 1997.

[3] Marcel Waldvogel, George Varghese, Jon Turner, and Bernhard Plattner. Scalable high speed IP routing table lookups. In *Proceedings of ACM SIGCOMM*, pages 25–36, September 1997.

[4] Butler Lampson, V. Srinivasan, and George Varghese. IP lookups using multiway and multicolumn search. In *Proceedings of IEEE INFOCOM*, San Francisco, 1998.

[5] E. C. Rosen, A. Viswanathan, and R. Callon. Multiprotocol label switching architecture. RFC 3031, Internet Engineering Task Force, January 2001.

[6] Paul Francis, Sugih Jamin, Vern Paxson, Lixia Zhang, Daniel F. Gryniewicz, and Yixin Jin. An architecture for a global Internet host distance estimation service. In *Proceedings of IEEE INFOCOM*, pages 210–217, New York, NY, USA, March 1999.

[7] T. S. Eugene Ng and Hui Zhang. Predicting Internet network distance with coordinates-based approaches. In *Proceedings of IEEE INFOCOM*, pages 170–179, New York, NY, USA, June 2002.

[8] Stefan Savage et al. Detour: A case for informed Internet routing and transport. *IEEE Micro*, 19(1):50–59, January 1999.

[9] Sally Floyd, Van Jacobson, Steve McCanne, Lixia Zhang, and Ching-Gung Liu. A reliable multicast framework for light-weight sessions and application level framing. In *Proceedings of ACM SIGCOMM*, pages 342–356, September 1995.

[10] Stephen Deering and David R. Cheriton. Multicast routing in datagram internetworks and extended LANs. *ACM Transactions on Computer Systems*, 8(2):85–110, May 1990.

[11] Sylvia Ratnasamy, Scott Shenker, and Ion Stoica. Routing algorithms for DHTs: Some open questions. In *Proceedings of First International Workshop on Peer-to-Peer Systems (IPTPS)*, 2002.

[12] Brad Karp and H. T. Kung. GPSR: Greedy perimeter stateless routing for wireless networks. In *Proceedings of MobiCom*, pages 243–254, August 2000.

[13] Roberto Rinaldi. Routing and data location in overlay peer-to-peer networks. Diploma thesis, Institut Eurécom and Università degli Studi di Milano, June 2002. Also available as IBM Research Report RZ-3433.

[14] Roberto Rinaldi and Marcel Waldvogel. Routing and data location in overlay peer-to-peer networks. Research Report RZ-3433, IBM, July 2002.

[15] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker. A scalable content-addressable network. In *Proceedings of ACM SIGCOMM*, September 2001.

[16] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of ACM SIGCOMM 2001*, pages 149–160, San Diego, CA, USA, August 2001.

[17] Anthony Rowstron and Peter Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, pages 329–350, Heidelberg, Germany, November 2001.

[18] Ben Y. Zhao, John Kubiatowicz, and Anthony Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Technical Report UCB/CSD-01-1141, April 2001.

[19] Miguel Castro, Peter Druschel, Y. Charlie Hu, and Antony Rowstron. Exploiting network proximity in distributed hash tables. In Ozalp Babaoglu, Ken Birman, and Keith Marzullo, editors, *International Workshop on Future Directions in Distributed Computing (FuDiCo)*, pages 52–55, June 2002.

[20] Sylvia Ratnasamy, Mark Handley, Richard Karp, and Scott Shenker. Topologically-aware overlay construction and server selection. In *Proceedings of INFOCOM*, June 2002.

[21] Greg N. Frederickson. Searching intervals and compact routing tables. *Algorithmica*, 15(5):448–466, May 1996.

[22] Robert Hinden and Stephen Deering. IP version 6 addressing architecture. Internet RFC 2373, 1998.

Transience of Peers & Streaming Media*

Mayank Bawa, Hrishikesh Deshpande, Hector Garcia-Molina
Computer Science Department, Stanford University
Stanford, CA 94305
{bawa,hrishi,hector}@cs.stanford.edu

ABSTRACT

Application level multicast schemes have traditionally been evaluated with respect to the *efficiency penalties* incurred in migrating the multicast functionality from the network layer to the application layer. We argue that the current performance measures, and therefore design strategies, are *incomplete* as they do not consider transience of peers. The routers in application level multicast systems are participant clients, and not infrastructure units. As such, the assumptions on the behavior of these application routers are significantly different from the infrastructure routing units that traditional research has dealt with, especially in a peer-to-peer setting where peers are multi-use and the management is decentralized. We argue that the transience in peer behavior has *implications on end-performance enabled*. We outline a design philosophy that seeks to *separate* policy decisions in handling peer behavior from the end-application at a basic infrastructural *peering layer*. As a proof of concept, we have implemented a peering layer prototype, which is available for download.

1. INTRODUCTION

Live streaming media will form a significant fraction of internet traffic in the near future. Recent trade reports indicate that if the current acceptance rate among end-users persists, streaming media could overtake television with respect to the size of the client base [1]. We expect the adoption trends for streaming media to be particularly remarkable in the context of peer-to-peer (P2P) systems.

Since video streams are high bandwidth applications, even a small number of clients receiving the stream by unicast are often sufficient to saturate bandwidth at the source. IP-Multicast [2] was proposed as an extension to Internet architecture to support multiple clients at network level. The deployment of IP Multicast has been slowed by difficult issues related to scalability, and support for higher layer functionality like congestion control and reliability. Several recent research projects [3, 4, 5, 6] have argued for a multicast service at the application layer (*end-host multicast*). The service is to be simulated over unicast-links between hosts, forming an overlay network over end-hosts.

The end-hosts in P2P systems form ad-hoc networks to contribute resources to the community, and in turn use resources provided by other members for a personal goal. Thus, members of a P2P system can be expected to share their bandwidth to spread a media stream to other clients. In principle then, an end-host multicast solution seems to be an ideal fit. Indeed, several such schemes [7, 8, 9] have been developed over the last year.

In practice, we believe however, that the migration of the multicast functionality from the network layer to the application layer leads to a subtle mismatch that has implications on end-application performance. The mismatch arises from a change in the characteristics of the routing infrastructure assumed by the end-application — the behavior of routing elements (clients) in end-host multicast is very different from those (routers) in IP-Multicast.

For example, consider the following scenario in streaming-media delivery via multicast. During an end-hosts multicast session, a large fraction of clients (that were acting as routers) might *unsubscribe together* in the middle of the stream, partitioning the overlay network. The time taken to repair the partitions will result in loss of packets transmitted during the transience period. Such an event, while probable in end-host multicast, is in contrast to network-level multicast where clients only occur as *leaves* in the multicast tree built on routers. Router failures are not as frequent, and the chances of simultaneous failures of a large number of routers, occurring frequently over the stream duration, is extremely small. Hence, while *packet losses* due to router failures is not a probable eventuality in IP-Multicast, it must be accounted for in an evaluation of an end-host multicast proposal. Unfortunately, conventional evaluations of such proposals have glossed over such implications on end-application performance.

It is our thesis that the primary challenge in any P2P architecture will be the *masking* of such peer transience. We will argue that end-applications are not easily insulated from the behavior of the new infrastructure units of a P2P network. A P2P network can be large with hundreds of nodes at

*This research was supported in part by NSF (Grant No. IIS-9817799). Mayank Bawa is supported by a Sequoia Capital Stanford Graduate Fellowship.

any given instant. More significantly, peers are autonomous, unpredictable, and may have short (of the order of minutes) lifetimes. Thus, it is critical for widespread adoption of any P2P application that the service be capable of good performance over a large and dynamic system, and the underlying architecture handle such transience gracefully.

Correspondingly, in this position paper, we will argue the following two points:

- We revisit the issue of evaluating an end-host multicast scheme and urge the use of *end-application metrics* to account for the loss in performance due to the transience of routing units (Section 3).
- We introduce the design philosophy of a *peering layer* that insulates peer transience from end-applications. As a proof of concept, we present an architecture for streaming media over a P2P network that uses the *peering layer* to efficiently enable a connected topology (Sections 4 and 5).

The rest of this position paper outlines the end-host multicast problem and presents conventional evaluation metrics (Section 2), summarizes current implementation status of the peering layer[1] (Section 6), and relates this work to other projects (Section 7). We conclude in Section 8.

2. PROBLEM AND PRELIMINARIES

In this section, we give an overview of a simple end-host multicast architecture to establish the vocabulary, and then point out conventional schemes used to evaluate end-host multicast proposals.

A media stream is a time ordered sequence of packets that is logically composed of two channels: *data* (served using unreliable RTP/UDP) and *control*(sent using reliable RTSP/TCP) channels. A *live* stream has the important property of being *history-agnostic*: a group-member is only interested in the stream from the instant of its subscription onwards.

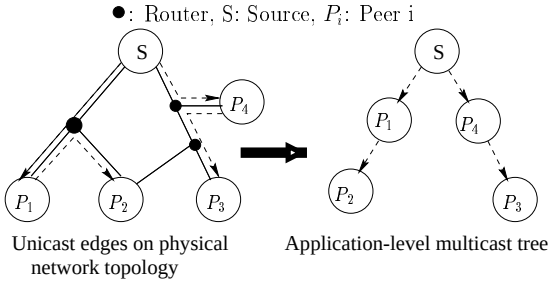


Figure 1: An application level multicast tree built on the peers

A simple architecture for an end-host multicast service organizes the group members into a source-specific spanning tree as shown in Figure 1. The figure uses dotted lines to show the flow of data packets, while the solid lines represent the underlying physical wires. Each node P_i (including the source S) forwards the stream to all its immediate children (if any). Effectively, each host acts as a multicast router, replicating and forwarding packets, and assuming responsibility for group management. The multicast tree is maintained incrementally as nodes join and leave. The scheme specifies protocols to enable an incoming client P_i to identify an existing node in the tree that can act as P_i's parent (*join policy*), and to enable a repair of partitions in the tree when a client opts out of its multicast session (*leave policy*).

The bandwidth requirements of S are now shared by the network of clients, providing a potentially scalable solution for media broadcast. However, there is a loss in efficiency entailed by moving the multicast functionality up the stack. For example, observe that the multicast tree is an overlay network — the actual forwarding paths in the physical network might visit the same network router multiple times. Thus, in Figure 1, the router on the path from S to P_1 will see each packet twice: once when S sends the packet to P_1, and next when P_1 forwards the packet to P_2. Moreover, routing packets along the overlay network results in increased delay from the source S to a client P_i.

Stemming from the influential work of [3] that identified such efficiency concerns, subsequent proposals have been evaluated with respect to the metrics of *stress, relative delay penalty,* and *normalized resource usage*. The *stress* of a physical link is defined as the number of identical copies of a packet carried by the link. The ratio of the delay between the source and a recipient along the overlay to the unicast delay between them is defined as the *relative delay penalty*. Finally, *resource usage* characterizes the network resources consumed in the process of data delivery to all recipients. Correspondingly, it is defined as $\sum_{i=1}^{L} d_i * s_i$, where L is the number of links active in multicast, d_i is the delay of link i, and s_i is the stress of link i. Observe that these metrics succinctly capture concerns on the loss in efficiency. However, note also that it is *not* possible to predict the effects of transience on end-application performance using the above efficiency metrics.

3. PERFORMANCE ANALYSIS FOR END-SYSTEM MULTICAST

In this section, we define a new set of *end-system metrics* to characterize performance of an end-host multicast solution. We then present a performance analysis scheme to enable the study of effects of router-transience on such metrics under a candidate solution.

An end-performance evaluation must, fundamentally, be influenced by the application semantics. A streaming media application exhibits the following characteristics that can stress the candidate architecture:

- *Sensitivity to data loss* that requires minimal packet loss while accommodating changes in membership, and
- *Timeliness constraints* that require changes in membership to be accommodated quickly.

Correspondingly, our *end-system metrics* to characterize performance for streaming media are *packet-losses*, *median lost-block widths*, *relative delay-penalty*, and *time to first packet (response time)* defined as follows:

[1]A skeletal implementation of the peering layer is available for download from [11].

- Let N_i be the total number of packets streamed by S during a client P_i's lifetime. Let N_{li} be the number of packets that were streamed by S during P_i's lifetime, but not received by P_i. Then, packet losses observed at P_i is defined as N_{li}/N_i.
- The disruptions in the stream become visible to the end-user when packet losses at P_i are consecutive, occurring as blocks of lost packets. Hence, it is important to record the sizes of lost packet-blocks at each client P_i. The median lost-block width at a client P_i, defined as the median of all lost packet-blocks observed over the lifetime of P_i, accounts for such disruptions.
- The response-time for a client P_i is defined as the time taken to receive the first stream packet at P_i after a subscribe request was sent by P_i.
- The relative delay-penalty at a client is defined as before (Section 2).

Given the above set of performance metrics, we next outline an analysis scheme. We propose that an evaluation of a multicast proposal must answer the following questions:

- What are the effects of the join policies on the end-system metrics? What are the parameter ranges over which the join policies can scale?
- What are the effects of the leave policies on the shape of the multicast topology? Are the leave policies successful in keeping the topology connected, compact, and stable?
- Which combinations of join and leave policies perform optimally with respect to the packet loss metrics?
- How do the policies compare against a centralized unicast scheme on the response time metric of time to first packet? Does the architecture scale gracefully with increase in the size of the client-base?
- Live streaming media often gives rise to flash crowds at the source, as the clients attempt to subscribe within a short interval of the start of the event. How does the architecture respond to such flash crowds?
- The clients receiving a stream have unpredictable lifetimes. How does the architecture behave under a range of expected lifetimes of the clients? Does the topology remain stable and connected for small lifetimes of clients?
- The clients may have a range of bandwidth capacities, from T3 to symmetric DSL. What is the effect of available uplink bandwidth capacity at a node on the end-system metrics?

A third and final piece remains to be specified — the behavior model of peers in a P2P network. We believe such modeling will be the source of interesting future work. Several P2P systems enjoy considerable deployment and can be used to collect behavioral traces. With these three pieces in hand, an end-host multicast proposal can be evaluated either through analysis or simulations.

To summarize, we believe that the efficiency metrics discussed in Section 2, while essential, were designed to characterize the effects of a functionality migration on the lower (network) layer alone. Unfortunately, it is not only the lower layers that have to bear a penalty for such a migration. Current end-applications are not designed to handle such unpredictability in the behavior of the infrastructure units, resulting in degradation of end-performance returned. We believe that an evaluation of a candidate multicast scheme under the analysis scheme proposed above will enable a more realistic and accurate test of feasibility.

4. THE PEERING LAYER

As we argued in Section 1, *masking* peer transience is the primary challenge in any P2P architecture that supports an end application. In this section, we describe the *peering layer* design philosophy that serves to hide topology changes from the above layers.

The peering layer is envisaged as a basic P2P infrastructure layer that exists below the end application. The peering layer is a placeholder for policies that govern topology maintainance — insertion and deletion of peers. By layering the end service on top of a core transience management service in the peering layer, we believe P2P systems will gain in ease of development.

As a proof of concept that the peering layer functionality is useful in the development of P2P services, we have designed a simple tree-based multicast architecture layered on the peering layer. The solution, called *PeerCast*, uses the peering layers at different nodes to establish and maintain a multicast tree. The PeerCast solution can work with short-lived nodes. Importantly, as we show elsewhere[10], by separating transience management from end applications, we were able to *enable existing implementations* of media players at a peer to use the peering layer. Thus peers were conveniently enabled for stream distribution *without* changing their existing code bases.

4.1 The Design Philosophy

We start by motivating the existence of a peering layer as a practical utility. Notice that a media stream is logically composed of two channels: data and control. An instance of the data and control channels taken together, between a server and a client constitutes a *data-transfer session*. Such a session is distinguished from an *application session* which begins when a client subscribes to a stream from a source, and ends when the client unsubscribes. An application session can subsume several data-transfer sessions as the client changes servers from which it receives feed for a given stream.

However, current implementations of most applications assume a unicast service, and bind the two sessions irrevocably together. We introduce a basic P2P infrastructure layer between the application and the transport layers for streaming that breaks the coupling between the two sessions. All communication between application and transport layers passes through the peering layer, as shown in Figure 2.

Data transfer sessions are established between the peering layers at the two nodes. The end-points are identified by the tuple $\langle$ *Server IP-address, Server port, Client IP-address, Client port, Stream URL* $\rangle$. The first four components identify the transport end-points involved in the session. The last component serves to identify the specific stream data is intended for.

Application sessions are established between the peering layer and the end application. The peering layer allows the appli-

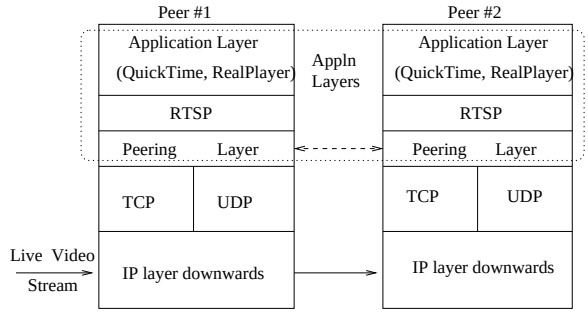


Figure 2: A layered architecture of a peer

cation layer above to specify the stream to obtain, through a "get-stream" interface. Given a stream URL, the peering layer locates a server which can provide the stream, and establishes a data-transfer session with the server. In the event of a data-transfer session termination, it locates a new server for data feed, and restarts the flow of data. If an alternate server cannot be found, an error is flagged to the application above. The application session survives these changes, leaving the end application *unaware* and, indeed, *unconcerned*, of shifts in the underlying multicast topology, as it should be.

The guarantees provided by the peering layer reflect those of the transport layer protocol used in the data-transfer session (for which it has to maintain state). In addition, it guarantees that the data feed to the application above will be maintained as long as an unsaturated server can be found.

4.2 The Redirect Primitive

The peering layer supports a simple, lightweight redirect primitive used to effect changes in the topology. The peering layer uses redirects as hints, to enable discovery of an unsaturated node in the network.

A redirect message is sent by a peer p to another peer c which is either opening a data-transfer session with p, or has a session already open. The message specifies a target peer t. On receipt of the redirect message, c closes its data-transfer session with p, and tries to establish a data-transfer session with t, for the same stream URL. Note that such redirect messages are produced and used at the peering layer, and the application above is unaware of such messages. Thus, the application session persists despite changes in the data-transfer sessions.

4.3 Discovering an Unsaturated Server

A new node n seeking the live stream needs to be able to discover an unsaturated node in the multicast group. The node n contacts the source s of the stream at the known URL. If s is unsaturated, it accepts n as its child and establishes a data-transfer session with n. Otherwise, s redirects n to one of its immediate children c. Then, n attempts to setup a data-transfer session with c. The process continues iteratively, until n gets accommodated. If n is unable to find an unsaturated node within some specified number of tries, the peering layer flags a resource unavailable error to the upper application-layer.

4.4 Managing Unsubscribe of Nodes

If a node n wishes to unsubscribe from the stream, it sends a leave message to its parent p. The parent p frees up resources dedicated to n at its side. However, the descendants of n are now disconnected from s, and experience a break in the stream. To enable a recovery following its unsubscribe, n sends redirects all its immediate children C to some target t (e.g., the source s, or the parent of n). The nodes in C then start the process of finding an unsaturated server by contacting t, as discussed above.

4.5 Handling Failures of Nodes

Intermediate nodes might fail, without being either able to inform their parent, or send redirect messages to their children. The overlay network needs to first detect such a failure, and then recover from it. The peering layer at a node uses a heart-beat mechanism to detect failures, and recovers by following the unsaturated-server discovery process as mentioned in Section 4.3.

5. POLICIES FOR TOPOLOGY MAINTENANCE

The peering layer discussed in Section 4 was said to serve as a placeholder for join and leave policies. Clearly, the nature of policies depend on the unique end application supported. In this section, we discuss simple candidate policies to show that the twin goals of performance and efficiency can be attained simultaneously by decisions *confined* at the peering layer.

5.1 Improving End-Application Performance

As we observed in Section 3, the end-system performance is characterized by packet losses, packet delays, and response times. It can be shown that the shape of the multicast tree has implications on the end-system metrics. In fact, it is easy to derive the following result.

THEOREM 1. *Under homogeneous unicast edge and node characteristics, an almost-complete spanning tree is the optimal overlay tree for packet loss, packet delay, and time to first packet metrics.*

The shape of the multicast tree is determined by the topology maintaining policy. Thus, an optimal policy is one that maintains an almost-complete spanning tree. Such a policy can then be implemented using the mechanisms we discussed in Section 4.

5.2 Improving Overlay Efficiency (or Adapting to Network Dynamics)

The multicast tree that has been formed using the policies discussed in Section 4 does not consider network attributes like link latencies, congestion, or peer bandwidths. Moreover, these quantities are dynamic, and it is important that the topology is rearranged in keeping with dynamic measurements of these quantities. Indeed, a lot of the previous work [3, 4, 5, 6, 12] in end-hosts multicast has focussed on these aspects, and suggested useful heuristics that can be included in the protocol to enable low penalties in efficiency

metrics. We indicate below how such heuristics can be used with our tree management policies of Section 4.

Nodes x and y that seek to establish a data-transfer session from x to y can compute a cost function $F(x, y)$ to characterize network proximity, unicast latency, and bandwidth capacity between them [4]. Given such a cost value, new members can join the multicast tree as suggested below.

- *Knock-Downs* : Each node n periodically recomputes $F(n, c)$ for each of its children c. For each incoming peer x that contacts a node n in the tree, $F(n, x)$ is computed. If n is unsaturated, it accepts x as a child. If n is saturated, it compares $F(n, x)$ with the maximum $F(n, c)$ among all of its children c. If $F(n, x)$ is cheaper, then n accepts x as its child, and redirects the child with the most cost. Otherwise, x contacts each of n's children, and tries to connect to the least cost candidate node.
- *Join-Flip*: Each internal node n periodically computes $F(p, n)$ for its parent p. A new incoming client x sends a $F(p, x)$ along with its request to n. If $F(p, x) + F(x, n)$ is less than $F(p, n) + F(n, x)$, n closes its data transfer session with p, redirects x to p, and then itself to x. Such a policy is used in [6] to form efficient trees.

We can also define incremental optimization heuristics that rearrange an *existing* overlay to improve efficiency.

- *Maintain-Flip*: This policy is similar to the *Join-Flip* discussed above. Instead of working with a new incoming client, existing nodes in the overlay tree are swapped. Each node n computes $F(p, n)$, $F(n, c)$ for its parent p and each of its children c periodically. If $F(p, n) + F(n, c)$ is more than $F(p, c) + F(c, n)$ for some child c, the positions of n and c are swapped. The work in [6] provides similar policies to build near-optimal trees. The work in [3, 5] uses such measures as part of their routing protocols to build a spanning tree with low efficiency penalties.
- *Leaf-Sink*: The authors in [13] propose a centralized algorithm to sink unstable or low-bandwidth nodes to the bottom of the tree. They also propose Redundant Virtual Links in the tree, which allow nodes to receive the same packet from multiple different sources, leading to decreased packet losses.

6. STATUS

The PeerCast system as described is under development. A few simple join and leave policies have been designed, implemented, and field tested. A discrete event based simulator has been implemented to evaluate PeerCast as outlined in Section 3. The initial results from field tests and simulations are encouraging. We are in the process of designing and conducting more detailed simulations.

7. RELATED WORK

To place our work in the context of proposed end-host multicast solutions, we cartooned existing proposals as in Figure 3 that illustrates the domain characteristics catered to. The horizontal axis plots the size of multicast group that can be supported. The vertical axis plots the assumed lifetime of hosts over which the overlay network is formed. The solutions indicated in Figure 3 are a representative set of wider work done in the field.

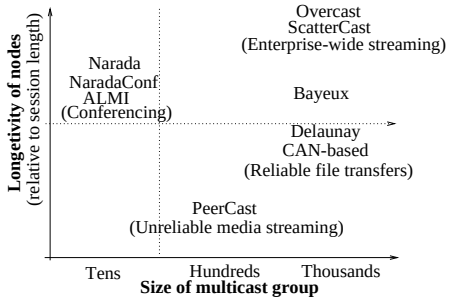


Figure 3: Classifying different end-host multicast proposals.

Solutions in the left portion of the grid [3, 4, 12] have been proposed for multicast over a group of tens of nodes, most of which live through the duration of the session. An example application domain is video conferencing, wherein participants are few (tens of members), and members live through most of the session. The top-right portion of the grid lists proposals [5, 6] that can scale to thousands of nodes, but assume existence of reliable, long-lived infrastructure hosts over which overlays are constructed. As such, these solutions are not adaptable to P2P systems. The middle-right portion lists proposals [7, 8, 9] that rely on primitives proposed in a P2P context to accommodate thousands of clients. These solutions have been proposed and tested in domains in which most of the members are expected to persist through the session. To the best of our knowledge, these proposals have not been proven to scale for hundreds of short-lived nodes forming a *dynamic* group.

The solution we propose, PeerCast, by being aware of the need to manage and mask peer transience from the end-application is able to push down on the longetivity axis in the grid. PeerCast is designed to scale to hundreds of short lifetime nodes that participate in long-durationed multicast session transmitted over unreliable unicast UDP/RTP.

8. CONCLUSIONS

In this paper, we argued that the migration of the multicast functionality from the network layer to the application layer results in a fundamental shift in the underlying infrastructure. The infrastructure units are now participating clients, that are known to be more whimsical and transient that network routers. The observation is especially true for P2P networks, where the participating peers are the providers of an end-service. We urge that the end-host multicast schemes be evaluated on end-system performance metrics of relative delay-penalty, response time, packet losses, and lost-block widths. We indicated that masking peer transience is a primary challenge in such a domain. We introduced the design philosophy of a peering layer at each peer that isolates policies for maintaining the topology from end-application functionality. The applications at a peer now need not be aware of a change in the server providing its data feed.

9. REFERENCES

[1] Shannon Dorey, "Streaming media - will it overtake television?,"

http://www.the-surfs-up.com/news/news4p1.html, 2001.

[2] S. Deering, "Multicast routing in a datagram internetwork," *PhD Thesis, Stanford University, California*, 1991.

[3] Y. Chu, S. Rao, and H. Zhang, "A case for end system multicast," in *Measurement and Modeling of Computer Systems*, 2000, pp. 1–12.

[4] Y. Chu, S. G. Rao, S. Seshan, and H. Zhang, "Enabling conferencing applications on the internet using an overlay multicast architecture," in *Proc. of the ACM SIGCOMM*, 2001.

[5] Y. Chawathe, "Scattercast: An architecture for internet broadcast distribution as an infrastructure service," *PhD Thesis, University of California, Berkeley*, 2000.

[6] J. Jannotti, D. Gifford, K. Johnson, M. Kaashoek, and J. O'Toole, "Overcast: Reliable multicasting with an overlay network," in *Proceedings of the Fourth Symposium on Operating Systems Design and Implementation*, 2000, pp. 197–212.

[7] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker, "Application-level multicast using content-addressable networks," in *Networked Group Communication*, 2001, pp. 14–29.

[8] S. Zhuang, B. Zhao, A. Joseph, R. Katz, and J. Kubiatowicz, "Bayeux: An architecture for scalable and fault-tolerant wide-area data dissemination," in *Proceedings of the Eleventh International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV 2001)*, 2001.

[9] J. Liebeherr, M. Nahas, and W. Si, "Application-level multicast with delaunay triangulations," Tech. Rep. CS-2001-26, University of Virginia, CS Dept., 2001.

[10] H. Deshpande, M. Bawa, and H. Garcia-Molina, "Streaming live media over peers," Tech. Rep. 2001-31, CS Dept., Stanford University, 2001.

[11] "http://www-db.stanford.edu/peers/," .

[12] D. Pendarakis, S. Shi, D. Verma, and M. Waldvogel, "ALMI: An application level multicast infrastructure," in *Proceedings of the 3rd USENIX Symposium on Internet Technologies and Systems (USITS)*, 2001, pp. 49–60.

[13] V. Roca and A. El-Sayed, "A host-based multicast (hbm) solution for group communications," in *1st IEEE International Conference on Networking*, 2001.

Is IP going to take over the world (of communications)?

Pablo Molinero-Fernández
Stanford University
molinero@stanford.edu

Nick McKeown
Stanford University
nickm@stanford.edu

Hui Zhang
Turin Networks and Carnegie Mellon University
hzhang+@cs.cmu.edu

ABSTRACT

While it is technically pleasing to believe that IP will dominate all forms of communication, our delight in its elegance is making us overlook its shortcomings. IP is an excellent means to exchange data, which explains its success. It remains ill suited as a means to provide many other types of service; and is too crude to form the transport infrastructure in its own right. To allow the continued success of IP, we must be open-minded to it living alongside, and co-operating with other techniques (such as circuit switching) and protocols that are optimized to different needs. In this position paper, we question some of the folklore surrounding IP and packet switching. We conclude that while packet-switched IP will continue to dominate best-effort data services at the edge of the network, the core of the network will use optical circuit switching as a platform for multiple services.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Circuit-switching networks, Packet-switching networks*; C.2.5 [**Computer-Communication Networks**]: Local and Wide-Area Networks—*Internet (e.g., TCP/IP)*

Keywords

IP, packet switching, circuit switching

1. INTRODUCTION

Whatever the initial goals of the Internet, there are two main characteristics that seem to account for its success: *reachability* and *heterogeneity*. IP provides a simple, single, global address to reach every host, enables unfettered access between all hosts, and adapts the topology to restore reachability when links and routers fail. IP hides heterogeneity in the sense that it provides a single, simple service abstraction that is largely independent of the physical links over which it runs. As a result, IP provides service to a huge variety of applications and operates over extremely diverse link technologies.

The growth and success of IP has given rise to some widely held assumptions amongst researchers, the networking industry and the public at large. One common assumption is that it is only a matter of time before IP becomes the sole global communication infrastructure, dwarfing and eventually displacing existing communication infrastructures such as telephone, cable and TV networks. IP is already universally used for data networking in wired networks (enterprise networks and public Internet), and is being rapidly adopted for data communications in wireless and mobile networks. IP is increasingly used for both local and long-distance voice communications, and it is technically feasible for packet-switched IP to replace SONET/SDH.

A related assumption is that IP Routers (based on packet-switching) will become the most important, or perhaps only, type of switching device inside the network. This is based on our collective belief that packet switching is inherently superior to circuit switching because of the efficiencies of statistical multiplexing, and the ability of IP to route around failures. It is widely assumed that IP is simpler than circuit switching, and should be more economical to deploy and manage. And with continued advances in the underlying technology, we will no doubt see faster and faster links and routers.

On the face of it, these assumptions are quite reasonable. Technically, IP is flexible enough to support all communication needs, from best-effort to real-time. With robust enough routers and routing protocols, and with extensions such as weighted fair queueing, it is possible to build a packet-switched, datagram network that can support any type of application, regardless of their requirements.

But for all its strengths, we (the authors) do not believe that IP will displace existing networks; in fact, we believe that many of the assumptions discussed above are not supported by reality, and do not stand up to close scrutiny.

It is the goal of this paper to question the assumption that IP will be *the* network of the future. We will conclude that if we started over - with a clean slate - it is not clear that we would argue for a universal, packet-switched IP network. We believe that in the future, more and more users and applications will demand predictability from the Internet;

both in terms of the availability of service, and the timely delivery of data. IP was not optimized to provide either, and so it seems unlikely to displace networks that already provide both.

We take the position that while IP will be the network layer of choice for best-effort, non-mission critical and non-real-time data communications (such as information exchange and retrieval), it will live alongside other networks, such as circuit-switched networks, that are optimized for high revenue time-sensitive applications that demand timely delivery of data and guaranteed availability of service.

We realize that our position is a controversial one. But regardless of whether or not we are correct, as researchers we need to be prepared to take a step back, to take a hard look at the pros and cons of IP, and its likely future. As a research and education community, we need to start thinking how IP will co-exist and co-operate with other networking technologies.

2. IP FOLKLORE

In what follows, we try to identify some folkloric assumptions about IP and the Internet, and examine each in turn. We will start with the most basic assumption, and easiest to dispel: that the Internet *already* dominates global communications. This is not true by any reasonable metric: market size, number of users, or the amount of traffic. Of course, the Internet has not yet reached maturity, and it may still grow to dominate the global communications infrastructure. We should ask ourselves if packet-switched IP offers inherent and compelling advantages that will lead to its inevitable and unavoidable dominance. This requires us to examine some "sacred cows" of networking; for example, that packet switching is more efficient than circuit switching, that IP is simpler, it lowers the cost of ownership, and it is more robust.

2.1 IP already dominates global communications

Although the Internet has been a phenomenal success, it is currently only a small fraction of the global communication infrastructure consisting of separate networks for telephones, broadcast TV, cable TV, satellite, radio, public and private data networks, and the Internet. In terms of revenue, the Internet is a relatively small business. The US business and consumer-oriented ISP markets have revenues of $13B each (2000) [5] [6], by contrast, the TV broadcast industry has revenues of $29.8B (1997), the cable distribution industry $35.0B (1997), the radio broadcast industry $10.6B (1997) [31], and the phone industry $268.5B (1999), of which $111.3B correspond to long distance and $48.5B to wireless [13]. The Internet reaches 59% of US households [22], compared to 94% for telephones and 98% for TV [20, 25]. It is interesting to note that, if the revenue per household remains the same, the total revenue for the ISP industry can at most double.

If we restrict our focus to the data and telephony infrastructure, the core IP router market still represents a small fraction of the public infrastructure, contrary to what happens in the private enterprise data networks. As shown in Table 1 the expenditure on core routers worldwide was $1.7B in 2001, compared to $28.0B for transport circuit switches. So in terms of market size, revenue, number of users, and expenditure on infrastructure, it is safe to say that the Internet does not currently dominate the global communications infrastructure.

The current infrastructure consists of a transport network - made of circuit-switched SONET and DWDM devices - on top of which run multiple service networks. The service networks include the voice network (circuit switched), the IP network (datagram, packet switched), and the ATM/Frame Relay networks (virtual-circuit switched). When considering whether IP has or will take over the world of communications, we need to consider both the transport and service layers.

In what follows, we will be examining which of two outcomes is more likely: Will the packet-switched IP network grow to dominate and displace the circuit switched transport network; or will the (enhanced) circuit-switched TDM and optical switches continue to dominate the core transport network?

Segment	Market size
Core routers	$1.7B
Edge routers	$2.4B
SONET/SDH/WDM	$28.0B
Telecom MSS	$4.5B

Table 1: World market breakup for the public telecommunications infrastructure in 2001 [30].

2.2 IP is more efficient

> "Analysts say [packet switched networks] can carry 6 to 10 times the traffic of traditional circuit-switched networks" – **Business Week**.

From the early days of computer networking, it has been well known that packet switching makes efficient use of scarce link bandwidth [1]. With packet switching, statistical multiplexing allows link bandwidth to be shared by all users, and work-conserving link sharing policies (such as FCFS and WFQ) ensure that a link is always busy when packets are queued-up waiting to use it. By contrast, with circuit switching, each flow is assigned to its own channel, so a channel could go idle even if other flows are waiting. Packet switching (and thus IP) makes more efficient use of the bandwidth than circuit switching, which was particularly important in the early days of the Internet when long haul links were slow, congested and expensive.

It is worth asking: What is the current utilization of the Internet, and how much does efficiency matter today? Odlyzko and Coffman [24, 9] report that the average link utilization in links in the core of the Internet is between 3% and 20% (compared to 33% average link utilization in long-distance phone lines [24, 29]). The reasons that they give for low utilization are threefold; First, Internet traffic is extremely asymmetric and bursty, but links are symmetric and of fixed capacity, second it is difficult to predict traffic growth in a link, so operators tend to add bandwidth aggressively, and

finally as faster technology appears it is more economical to add capacity in large increments.

There are other reasons to keep network utilization low. When congested, a packet-switched network performs badly, becomes unstable, and can experience oscillations and synchronization. Because routing protocol packets are transmitted in-band, they can be lost or delayed due to network congestion or control processor overload. This causes inconsistent routing state, and may result in traffic loops, black holes and disconnected regions of the network, which further exacerbates congestion in the data path [17]. Today, network providers address these problems by keeping network utilization low.

But perhaps the biggest reason that network providers overprovision their network is to give low packet delay. Users want predictable behavior, which means low queueing delay, even under abnormal conditions (such as the failure of several links and routers). As users, we already demand (and are willing to pay for) huge over-provisioning of Ethernet networks (the average utilization of an Ethernet network today is about 1% [9]) just so we do not have to share the network with others, and so that our packets can pass through without queueing delay. We will demand the same behavior from the Internet as a whole. We will pay network providers to stop using statistical multiplexing, and to instead over-provision their networks, as if it were circuit switched [12]. The demand for lower delay will drive providers to decrease link utilization even lower than it is today.

But simply reducing the *average* link utilization will not be enough to make users happy. For a typical user to experience low utilization, the *variance* of the network utilization needs to be low, too. Reducing variations in link utilization is hard; today we lack effective techniques to do it. It might be argued that the problem will be solved by research efforts on traffic management, congestion control, and multipath routing. But to-date, despite these problems being understood for many years, effective measures are yet to be introduced.

On a related note, we might ask whether users experience lower delay in a packet switched or circuit switched network. Intuition suggests that packet switching will lead to lower delay: A packet switched network easily supports heterogeneous flow rates, and flows can always make forward progress because of processor-sharing in the routers. In practice, we find that it doesn't make much difference whether we use packet switching or circuit switching. We explored this in detail in some earlier work, where we studied (using analysis and simulation) the effect of replacing the core of the network with dynamic fine-granularity circuit switches [19]. Let's define the user response time to be the time from when a user requests a file, until this file finishes downloading. Web browsing and file sharing represent over 65% of Internet transferred bytes today [7], and so the request/response model is representative of typical user behavior. Now consider two types of network: One is the current packet-switched network in which packets share links. In the other network each new application flow triggers the creation of a low bandwidth circuit in the core of the network, similar to what happens in the phone network. If there are no circuits available, the flow is blocked until a channel is free. At the core of the network, where the rate of a single flow is limited by the data-rate of its access link, simulations and analysis in [19] indicate that the user response time is essentially the same for packet switching and circuit switching, independent of the flow length distribution.

In summary, we have observed that packet switching can lead to more efficient link utilization. While efficiency was once a critical factor, it is so outweighed by our need for predictability, stability, immediate access and low delay that network operators are forced to run their networks at a low utilization, forfeiting the benefits of statistical multiplexing.

2.3 IP is robust

> "The Internet was born during the cold war 30 years ago. The US Department of Defence [decided] to explore the possibility of a communication network that could survive a nuclear attack." – **BBC**

The Internet was designed to withstand a catastrophic event where a large number of links and routers were destroyed. This goal is in line with users and businesses who rely more and more on the network connectivity for their activities and operations, and who want the network to be available at all times. Much has been claimed about the reliability of the current Internet, and it is widely believed to be inherently more robust. Its robustness comes from using soft-state routing information; upon a link or router failure it can quickly update the routing tables and direct packets around the failed element.

The reliability of the current Internet has been studied by Labovitz et al. [17]. They have studied different ISPs over several months, and report a median network availability equivalent to a downtime of 471 min/year. By contrast Kuhn [15] found that the average downtime in phone networks is less than 5 min/year. As users we have all experienced network downtime when our link is unavailable, or some part of the network is unreachable. On occasions, connectivity is lost for long periods while routers reconfigure their tables and converge to a new topology. Labovitz et al. [16] observed that the Internet recovers slowly, with a median BGP convergence time of 3 minutes, and frequently taking over 15 minutes. By contrast, SONET/SDH rings, through the use of pre-computed backup paths, are required to recover in less than 50 ms; a glitch that is barely noticeable by the user.

As discussed in Section 2.2, the likelihood of a network getting into a inconsistent routing state is much higher in IP networks because (a) the routing packets are transmitted in-band, and therefore are more likely to incur congestion due to high load of user traffic; (b) the routing computation in IP networks is very complex, therefore, it is more likely for the control processor to be overloaded; (c) the probability of mis-configuring a router is high. And mis-configuration of even a single router may cause instability of a large portion of the network. It is surprising is that we have continued to use routing protocols that allow one badly behaved router to make the whole network inoperable [18]. In contrast,

high availability has always been a government-mandated requirement for the telephone network, and so steps have been taken to ensure that it is an extremely robust infrastructure. In circuit networks control messages are usually transmitted over a separate channel or network, and the routing is much simpler.

In datagram networks, inconsistence routing state may cause black holes or traffic loops so that the service to existing user traffic is disrupted – i.e. inconsistent routing is *service impacting*. In circuit networks, inconsistent routing state may result in unnecessary rejection of request for new circuits, but none of the established circuits is affected. In summary, currently with IP, not only are failures more common, but they take longer to be repaired and their impact is deeper.

The key point here is that there is nothing inherently unreliable about circuit switching, and we have proof that it is both possible and economically viable to build a robust circuit-switched infrastructure, that is able to quickly reconfigure around failures. There is no evidence yet that we can define and implement the dynamic routing protocols to make the packet-switched Internet as robust. Perhaps the problems with BGP will be fixed over time and the Internet will become more reliable. But it is a mistake to believe that packet switching is inherently more robust. In fact, the opposite may be true.

2.4 IP is simpler

"IP-only networks are much easier and simpler to manage, leading to improved economics." – **Business Communications Review**

It is an oft-stated principle of the Internet that the complexity belongs at the end-points, so as to keep the routers simple and streamlined. While the general abstraction and protocol specification are simple, implementing a high performance router and operating an IP network are extremely challenging tasks, particularly as the line rates increase.

If we are looking for simplicity, we can do well to look at how circuit-switched transport switches are built. First, the software is simpler. The software running in a typical transport switch is based on about three million lines of source code [28], whereas Cisco's Internet Operating System (IOS) is based on eight million [10], over twice as many. Routers have a reputation for being unreliable, crashing frequently and taking a long time to restart. So much so that router vendors frequently compete on the reliability of their software.

The hardware in the forwarding path of a circuit switch is also simpler than that of a router. At the very least, the line card of a router must unframe/frame the packet, process its header, find the longest-matching prefix that matches the destination address, decrement the TTL, process optional headers, and then buffer the packet. If multiple service levels are added (e.g., DiffServ [3]), then multiple queues must be maintained, as well as an output link scheduling mechanism.

On the other hand, the linecard of an electronic transport switch typically contains a SONET framer to interface to the external line, a chip to map ingress time slots to egress time slots, and an interface to a switch fabric. Essentially, one can build a transport linecard [27] by starting with a router linecard [26] and then removing most of the functionality.

One measure of this complexity is the number of logic gates implemented in the linecard of a router. An OC192c POS linecard today contains about 30 million gates in ASICs, plus at least one CPU, 300Mbytes of packet buffers, 2Mbytes of forwarding table, and 10Mbytes of other state memory. The trend in routers has been to put more and more functionality on the forwarding path: first, support for multicast (which is rarely used), and now support for QoS, access control, security and VPNs (and we thought that all the complexity was in the end system!). By contrast, the linecard of a typical transport switch contains a quarter of the number of gates, no CPU, no packet buffer, no forwarding table, and an on-chip state memory (included in the gate count). Because they use simpler hardware, electronic circuit switches consume less power, allowing more capacity to be placed in a single rack. It should come as no surprise that the highest capacity commercial transport switches have two to twelve times the capacity of an IP router [8, 21, 14], and sell for about half to 1/12 per gigabit per second. So even if packet switching might be simpler for low data rates, it becomes more complex for high data rates. IP's "simplicity" does not scale.

It is interesting to explore how optical technology will affect the performance of routers and circuit switches. In recent years, there has been a lot of discussion about all-optical Internet routers. There are two reasons why this does not make sense. First, a router is a packet switch, and so inherently requires large buffers to hold packets during times of congestion, and there are currently no economically feasible ways to buffer large numbers of packets optically. The second reason is that an Internet router must perform an address lookup for each arriving packet. Neither the size of the routing table, nor the nature of the lookup, lends itself to implementation using optics.

Optical switching technology is much better suited to circuit switches. Devices such as tunable lasers, MEMS switches, fiber amplifiers and DWDM multiplexers provide the technology to build extremely high capacity, low power circuit switches that are well beyond the capacities possible in electronic routers [2].

2.5 Support of telephony and other real-time applications over IP networks

"All critical elements now exist for implementing a QoS-enabled IP network." – **IEEE Communications Magazine**

There is a widely-held assumption that IP network can support telephony and other real-time applications. If we look more closely, we find that the reasons for such an optimistic assumption are quite diverse. One school holds the view that IP is ready today: IP networks are and will continue to be heavily over-provisioned, and the average packet delay in the network will be low enough to satisfy the real-time requirements of these applications. These real-time applications, including telephony, can tolerate occasional packet delay/loss and *adapt* to these network variabilities. While

today's IP networks are heavily over-provisioned, it is doubtful whether a new solution (far from complete yet) that provides a worse performance can displace the reliable and high quality of service (QoS) provided by today's TDM-based infrastructure (which is already paid-for).

Another school believes that for IP to succeed, it is critical for IP to provide QoS with the same guarantees as TDM but with more flexibility. In addition, the belief is that there is no fundamental technical barrier to build a connection-oriented service (Tenet [11] and IntServ [4]) and to provide guaranteed services in the Internet. Unfortunately, after more than 10 years of extensive research and efforts in the standards bodies, the prospect of end-to-end per-flow QoS in the Internet is nowhere in sight. The difficulty seems to be the huge culture gap between the connection and datagram design communities. By blaming the failure on "connections", a third school holds the view that a simpler QoS mechanism such as DiffServ is the right way to go. Again, we are several years into the process, and it is not at all clear that the "fuzzy" QoS provided by DiffServ will be good enough for customers who are used to the simple QoS provided by the existing circuit-switched transport networks.

Finally, no matter what technology we intend to use to carry voice over the Internet, there are few financial incentives to do so. As Mike O'Dell[1] recently said [23]: "[to have a Voice-over-IP service network one has to] create the most expensive data service to run an application for which people are willing to pay less money everyday, [...] and for which telephony already provides a better solution with a marginal cost of almost zero."

3. DISCUSSION

Up until this point, we have considered some of the folklore surrounding the packet-switched Internet. Our overall goal is to provoke discussion and research on fundamental issues that need to be addressed so that IP can continue to revolutionize the world of communications. We hope to provide a vantage point for the IP community to reflect upon the problems that still need to be solved.

3.1 Dependability of IP networks

High dependability, in the broadest sense, is a must if IP is to become the universal infrastructure for high value applications. For example, voice services and private lines are a high-revenue, and very profitable business. Trusting them to today's unreliable, and unpredictable IP networks would be an unnecessary risk, which is why — despite predictions to the contrary — telephone carriers have not done so.

High dependability means several things: robustness and stability, traffic isolation, traffic engineering, fault isolation, manageability, and last but not least, the ability to provide predictable performance in terms of bounded delay and guaranteed bandwidth (QoS). In its current form, IP excels in none of these areas. Although it is clearly a challenge to achieve each of these goals, they must all be solved for IP to become dependable enough to be used as a transport mechanism.

[1]former Senior Vice President of UUNET, responsible for technical strategic direction and architecture of the network.

3.2 How IP should interact with circuits

The current Internet is based on packet switched routers, interconnected by a circuit switched transport network. Given the benefits of circuit switching, it is inconceivable to us that the network providers would remove the existing, robust, reliable, predictable and largely paid-for transport network, and replace it with a technology that seems more complex, less reliable, more expensive and not yet installed.

What seems more likely is that packet switching will continue to exist at the edge of the network, aggregating and multiplexing traffic from heterogeneous sources for applications that have no delay or quality requirements.

At the core of the network, we expect the circuit switched transport network to remain as a means to interconnect the packet switched routers, and as a means to provide high reliability, and performance guarantees. Over time, more and more optical technology will be introduced into the transport network, leading to capacities that electronic routers cannot achieve.

However, IP will not be the only way to access the circuit switched transport network. Because the packet switched network is unlikely to provide the predictability needed for voice traffic, voice will continue to operate over its own, separate circuit switched edge network and be carried over the shared transport network at the core. This leads us to believe that it is more likely that the routers will be allocated a significant fraction of the circuit switched transport infrastructure, which they can control and adapt to best serve their needs. Such a system has the benefit of enabling IP to gain the benefits of fast optical circuit switches in the core, yet maintain the simple service model for heterogeneous sources at the edge.

3.3 What if we started with a clean slate

In the preceding discussion, we predicted an outcome based on historical reasons, in the context of a pre-existing circuit switched transport network. So if we started again, with the benefit of hindsight, would we build a network with circuit switching at the core, and packet switching at the edge? We believe that we would, and that it would look something like this:

Switching in the edges of the network. Packet switching would be used in the edges of the network as well as those links where bandwidth is scarce (such as wireless access links, satellite links, and underwater cables). The reasons for this are twofold. First, packet switching makes a very efficient use of the bandwidth in these cases. Second, it can greatly improve the end-user response time by borrowing all available link bandwidth when other users are not active. The packet-switched network should ideally gather traffic from disparate sources, and multiplex it together in preparation for carriage over a very high capacity, central, circuit-switched core.

Switching in the core of the network. At the core of the network, there seem to be a number of compelling reasons to use circuit switching: Circuit switching has already demonstrated its robustness, and its ability to quickly recover from failures. It is inherently simpler than packet

switching, requiring less work to forward data, and so will cost less per capacity unit as a result, will consume less power, and will take up less space. Last, though probably first, circuit switching provides an easy way to adopt the huge potential of high capacity optical switches at low cost.

Integration of both switching mechanisms. Rather than working independently, both of these mechanisms should be tightly integrated, in such a way that an action in one provokes a reaction in the other. For example, packet switching would have to export the QoS and connection oriented nature of the circuit switched core to the applications that require it. Similarly circuit switching has to respond to the increases in traffic of packet switching, by adapting its capacity among core/edge gateways accordingly. Additionally, we will find more hybrid switches that can do both circuit and packet switching, serving as gateways between the two worlds.

4. REFERENCES

[1] P. Baran. Introduction to distributed communications networks. Memorandum RM-3420-PR, Rand Corporation, Aug 1964.

[2] D. J. Bishop, C. R. Giles, and G. P. Austin. The Lucent LambdaRouter: MEMS technology of the future here today. *IEEE Communications Magazine*, 40(3):75–79, Mar 2002.

[3] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. RFC 2475: An architecture for differentiated services, Dec. 1998.

[4] R. Braden, D. Clark, and S. Shenker. RFC 1633: Integrated services in the Internet architecture: an overview, June 1994.

[5] Cahners. 2001 business isps: Service, size, and share. advanced carrier business report. Report, Cahners, Oct 2001.

[6] Cahners. Information alert newsletter. volume #23. Newsletter, Cahners, July 2001.

[7] CAIDA, Cooperative Association for Internet Data Analysis. *OC48 analysis summary: Distributions of traffic stratified by application*, 2002.

[8] Ciena. *CIENA MultiWave CoreDirector*, 2001.

[9] K. Coffman and A. Odlyzko. *Handbook of Massive Data Sets*, chapter Internet growth: Is there a "Moore's Law" for data traffic? J. Abello, P. M. Pardalos, and M. G. C. Resende editors, Kluwer, 2001.

[10] C. Edwards. Panel weighs hardware, software design options. *EE Times*, June 2000.

[11] D. Ferrari. Real-time communication in an internetwork. *Journal of High Speed Networks IOS Press*, 1(1):79–103, 1992.

[12] C. Fraleigh. *Provisioning IP Backbone Networks to Support Delay Sensitive Traffic*. PhD thesis, Electrical Engineering Dept., Stanford University, 2002.

[13] Industry Analysis Division, Common Carriers Bureau. Trends in telephone service. Report, US Federal Communications Commission, Aug 2001.

[14] Juniper Networks. *T640 Internet Routing Node: Datasheet*, 2002.

[15] R. Kuhn. Sources of failure in the public switched telephone network. *IEEE Computer*, 30(4):31–36, April 1997.

[16] C. Labovitz, A. Ahuja, A. Bose, and F. Jahanian. Delayed internet routing convergence. *IEEE/ACM Transactions On Networking*, 9(3):293–306, June 2001.

[17] C. Labovitz, A. Ahuja, and F. Jahanian. Experimental study of internet stability and wide-area backbone failures. In *Proceedings of FTCS*, Madison, WI, June 1999.

[18] C. Labovitz, R. Wattenhofer, S. Venkatachary, and A. Ahuja. Resilience characteristics of the internet backbone routing infrastructure. In *Proceedings of the Third Information Survivability Workshop*, Boston, MA, October 2000.

[19] P. Molinero-Fernández and N. McKeown. TCP Switching: Exposing circuits to IP. *IEEE Micro Magazine*, 22(1):82–89, Jan/Feb 2002.

[20] National Telecommunications and Information Administration. Falling through the net: Defining the digital divide. Technical report, US Department of Commerce, 1999.

[21] Nortel Networks. *OPTera Connect HDX optical switch*, 2002.

[22] Nua Internet Surveys. *How Many On-line?*, April 2002.

[23] M. O'Dell. Keynote speech. Sigcomm conference, August 2002.

[24] A. Odlyzko. Data networks are mostly empty and for good reason. *IT Professional*, 1(2):67–69, Mar/Apr 1999.

[25] A. Penenberg. The war for the poor. *Forbes Magazine*, 26 Sept 1997.

[26] PMC-Sierra. *Diagram of a 10 Gigabit Core Router Architecture*, April 2002. `http://www.pmc-sierra.com/products/diagrams/CoreRouter_lg.html`.

[27] PMC-Sierra. *Diagram of a Sub-wavelength Optical Cross Connect*, April 2002. `http://www.pmc-sierra.com/products/diagrams/SubWavelengthCrossConnect_l%g.html`.

[28] Private communication. Source requested not to be identified, April 2002.

[29] RHK. North american telecom capex to turn up in 2004. Press release #154, RHK, April 2002.

[30] RHK. Various industry reports and industry news, 2002. IN#114, IR#1101, IR#1102, IR#1079.

[31] US Census. Industry quick report. Technical report, US Department of Commerce, 1997.

Current Issues in Packet Switch Design

Cyriel Minkenberg, Ronald P. Luijten, François Abel, Wolfgang Denzel, Mitchell Gusat
IBM Research, Zurich Research Laboratory
Säumerstrasse 4, CH-8803 Rüschlikon, Switzerland
sil@zurich.ibm.com

ABSTRACT

Addressing the ever growing capacity demand for packet switches, current research focuses on scheduling algorithms or buffer bandwidth reductions. Although these topics remain relevant, our position is that the primary design focus for systems beyond 1 Tb/s must be shifted to aspects resulting from packaging disruptions. Based on trends such as increased link rates and improved CMOS technologies, we derive new design factors for such switch fabrics. For instance, we argue that the packet round-trip transmission time *within* the fabric has become a major design parameter. Furthermore, we observe that high-speed fabrics have become extremely dependent on serial I/O technology that is both high speed *and* high density. Finally, we conclude that in developing the architecture, packaging constraints must be put first and not as an afterthought, which also applies to solving the tremendous power consumption challenges.

1. MOTIVATION

Most research on single-stage, electronic packet switches focuses primarily on high-level architectural issues such as buffering and queuing strategies and scheduling algorithms, but seldom considers all the physical issues that arise when actually building the system, a tendency that has also been noted in [1]. For instance, although many new packet-switch architectures have been proposed in the past few years, the majority of these architectural-level papers reduce memory bandwidth by eliminating the N-fold speed-up required by output queuing and instead optimizing the performance of the centralized scheduling algorithm needed in such architectures. Although reducing memory bandwidth is an important issue, it is not sufficient in itself and may render the resulting system economically infeasible. Good examples are the recent proposals of combined input- and output-queued (CIOQ) switches with limited speed-up, which require speed-up throughout the fabric (i.e., both input and output speed-up), thus multiplying the bandwidth that must be carried across the switch core, whereas the output speed-up implemented in an output-queued switch is purely internal to the switch core.

Designers of practical high-capacity packet switches face challenges on two levels: First, they must choose a design point at the architectural level, in terms of buffering and queuing strategies, scheduling algorithms, and flow-control methods. For example, for the buffering strategy, the choice to be made is between an input-, output-, or combined-queuing structure. Second, they must consider the physical level, i.e., the implementation of the architecture at the system as well as the chip level, in terms of partitioning over racks, cards, chips, and the design of the individual chips comprising the system. Switch designers have little freedom with respect to system packaging issues. On the one hand the technology imposes constraints, on the other hand customers impose their specific requirements, in addition to more general ones such as NEBS (Network Equipment Building System) compliance [2, 3]. NEBS comprises a set of stringent physical (e.g., space planning, temperature, humidity, etc.) and electrical (e.g., EMI, power fault, bonding and grounding, etc.) requirements, originally developed for telephony equipment. Nowadays, NEBS compliance is a prerequisite for networking and computing equipment in general to ensure reliable operation (also under adverse conditions), safety, compatibility, and freedom of interference.

The designer must decide how to distribute functionality over chips, taking into account current technology limitations. At every level, the designer is constrained by requirements and technological feasibility, and has to optimize overall system cost and power. We argue that the new constraints arising from packaging and power invalidate the traditional design approach. Rather than finding a suitable packaging for the architecture, we are forced to find a suitable architecture for the packaging.

Although a multi-Tb/s switch is not (yet) a commodity, for cost reasons we assume the use of "commodity" technology and components in our study, i.e., CMOS technology, standard-cell design (no full custom), commodity chip packaging (< 1000 pins per chip), commercially available connectors, and standard system packaging (e.g., NEBS compliance). We also assume that packet-level QoS and routing are required.

In Sec. 2 we discuss the major trends in switch design in a trend/cause/effect format. Section 3 gives a system-level

HOTNETS '02 10/02 Princeton, NJ, USA

description, introducing some basic assumptions about system structure. In Sec. 4 we discuss the major consequences on switch design in general that result from the trends observed, whereas in Sec. 5 we discuss a number of specific implications for two popular architectures. Finally, we draw our conclusions in Sec. 6. The main purpose of this paper is to draw attention to the issues discussed, rather than to provide solutions.

2. TRENDS

Trend 1. The aggregate throughput will grow by increasing the *number of ports* rather than port speed.

Cause 1. Although transmission line rates have increased rapidly over the past years [OC-3 (155 Mb/s) to OC-192 (10 Gb/s) and OC-768 (40 Gb/s)] and will most likely continue to do so, it appears that the granularity at the switch level will be OC-192 for the coming years [4]. First, the existing installed base of OC-192 line cards must still be supported. Second, dense wavelength-division multiplexing (DWDM) vastly increases the number of channels available on a single fiber, but not the speed of a single channel. At the switch, this translates into more ports at the same speed. Also, on the electrical side the SRAM clock cycle poses a limit in combination with the minimum packet size; the limit is reached when the line rate times the clock cycle exceeds the *minimum* packet size. This limit can only be overcome by enlarging the minimum packet size, which requires packet aggregation techniques. Right now, a cycle of 2 ns is feasible, which, given a minimum packet size of 64 B, yields a maximum line rate of 256 Gb/s, which is just one generation away (OC-3072 = 160 Gb/s).[1] Finally, it is becoming increasingly difficult and costly to perform line-speed processing in the network processors (NP) at these rates.

Contrary to wavelength density, *port* density in terms of ports per line card is not increasing, for two reasons. First, availability requires that single points of failure be avoided as much as possible. Putting multiple NPs and adapters on one line card would cause service interruption for several ports if one component on the card fails. Second, the increase in density offered by CMOS technology is invested in increased functionality, such as level-4 routing, firewalling, MPLS, and DiffServ, rather than increasing speed, or reducing size or power.

Effect 1. The increase in port count at constant port speed translates directly into an increase in physical line-card space. On the other hand, compliance with NEBS equipment requirements for the physical form of the system imposes strict limitations on the number of cards that fit into a rack. Combined with the above trend, it quickly becomes clear that the switch fabric can no longer be built in a compact, single-rack fashion, and that a multi-rack solution is necessary. In a single-rack system, all fabric-internal communication only crosses the backplane, but once the system becomes multi-rack, line cards and switch core become separated by a much greater physical distance, with (long) cables[2] interconnecting the racks, implying that the rack's backplane is replaced by cables. Rack-spacing requirements and other spatial limitations (adapter racks might even be in other rooms than the switch core rack) determine how far apart the racks have to be—this can easily be tens of meters.

Trend 2. With each new switch generation a higher proportion of the switch chip power is used to transport signals across chip boundaries.

Cause 2. The number of chip pins grows by 5 to 10% with each CMOS generation, whereas density doubles. Rent's rule [5] states that the number of pins, N_p, and number of logic gates, N_g, form a relationship of the form $N_p = K_p N_g^{\beta}$, where K_p is a constant and $\beta < 1$. Therefore, higher per-pin signal rates are needed to scale aggregate bandwidth proportionally, which in turn implies more complex analog macros requiring increased power. Table 1 shows numbers for three generations of a typical switch chip.

Effect 2. Switch chips become increasingly I/O- and power-constrained. At the system level currently more than 50% of the power is spent in transporting rather than switching data.

Table 1: Single-chip I/O Power Consumption Proportion

Switch size	16×16	16×16	32×32
Throughput (Gb/s)	6	32	64
Total power (W)	6	12	25
I/O power (W)	1	4	12
I/O power (%)	**16**	**33**	**50**

Trend 3. The growth in CMOS integration density far outpaces the growth in electrical interconnect bandwidth density.

Cause 3. The bandwidth growth over chip pins as well as card edge is a few percent per year, compared with the doubling of CMOS technology density every 18 months. The card-edge limitations impose even tighter constraints than the chip-pin limit in the area of switching.

Effect 3. The maximum switch throughput that can be implemented on a single switch card in a multi-rack system is limited by card connector technology rather than by CMOS density. Given a target system capacity, this limit immediately imposes a lower limit on the number of cards required. Connecting across 5 m of cable plus about 50 cm of FR4 board circuit trace limits the bit rate to 2.5–3.125 Gb/s. Combined with the latest connector technology, this yields a density of about 120 Gb/s/in. With reasonable card sizes

[1]Fabric-internal line-rate escalation (speed-up) is required to compensate for header and segmentation overhead. We assume a typical speed-up of 60%.

[2]By cable we mean electrical and/or optical interconnects spanning a few to tens of meters.

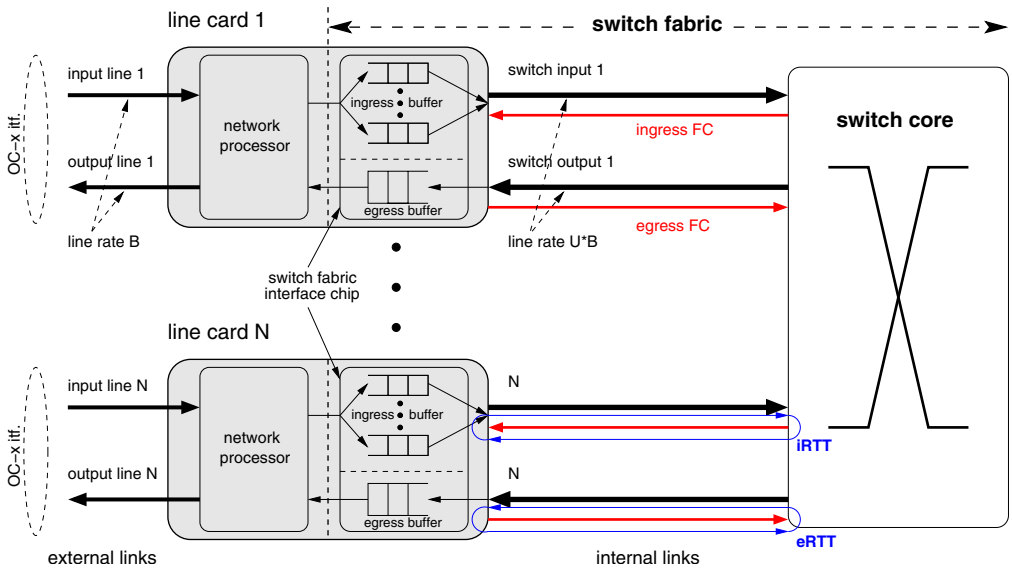


Figure 1: System level architecture.

($\leq$ 50 cm card edge) this results in a maximum card throughput of about 1 Tb/s (bidirectional), although CMOS technology would allow us to build higher throughput on a single card. Consequently, larger fabrics (i.e., multi-Tb/s) have to be split over multiple cards. Optical interconnect technology does not yet provide any improvement here (more space, power, and cost, but not faster), although it may be needed to cover longer distances. To satisfy availability requirements (typically 99.999% uptime is a must, which translates into about 5 min downtime per year), the switch must be implemented with a sufficient degree of redundancy. As the element of redundancy is a card, a small, single-card switch can economically achieve 1+1 redundancy, whereas fabrics larger than 1 Tb/s preferably should employ N+1 redundancy.

Trend 4. The required throughput/power density in terms of Gb/s per Watt per physical volume is increasing.

Cause 4. From one generation to the next, the market expects increased performance at constant or less power consumption. This also applies to performance per physical volume. One of the reasons is that high-end routers are often packaged as telecom racks, which are subject to a fixed power-consumption limit per rack (typically 2 kW/shelf), and must also be NEBS compliant.

Effect 4. The throughput/power density of all system components must be scaled up in proportion to the increase in aggregate bandwidth.

Trend 5. The minimum packet duration has shrunk significantly.

Cause 5. Whether traffic is ATM, IP, or SAN, the minimum packet size is still in the range of 32 to 64 B. The line rate has evolved exponentially to OC-192 currently, and to OC-768 in the near future. Accordingly, the transmission duration of the minimum-sized packet has shrunk from micro- to nanoseconds.

Effect 5. On a given length of cable or backplane trace, more packets are in flight.

Trend 6. Moore's law paradox in the latest CMOS generations.

Cause 6. Although Moore's law doubles performance every 18–24 months at constant cost, this performance is mostly due to increased density rather than increased clock speeds. From one CMOS generation to the next, taking global wiring into account, switch-chip clock speeds can be increased by only 5–10%.

Effect 6. To exploit Moore's law, more parallelism is required, typically at constant clock speeds. In turn, this results in more levels of pipelining in the control path and a higher degree of parallelism in the data path. Similarly, on-chip memory speed does not increase, and memory busses become wider in each new generation.

3. SYSTEM-LEVEL ASSUMPTIONS

To be able to make some more concrete statements, we need to make a few assumptions at the architectural level, see Fig. 1. First, most practical switch systems consist of three components: the ingress line cards, the switch core (routing fabric), and the egress line cards. Typically, the ingress and egress line cards of the same port are implemented together on the same card. The ingress line card comprises a network processor that communicates through a switch-core interface (e.g., SPI-x, CSIX [6]) with an adapter chip that per-

forms the translation to the native packet format. Both network processor and adapter chip contain buffers. Buffers on the egress side are necessary to accommodate downstream blocking and the speed difference between internal and external link rates. These differ to account for the segmentation and header overhead incurred by the switch-internal packet format, which typically does not map directly to the external packet format (e.g., conversion from variable-length TCP/IP packets to short, fixed-length packets). The switch core performs the actual switching function—this can be any type of switch, e.g. a crossbar, a buffered crossbar, a shared-memory switch, etc. We assume that the routing fabric is single-stage and that it contains no further input buffers (all input buffering is performed on the line cards).

To ensure that the switch fabric is internally lossless, flow-control protocols are required between any two stages that contain (limited) buffers. The *round-trip time* (RTT), as illustrated in Fig. 1, is the sum of the latencies of the forward (data) and the reverse (flow control) path, usually expressed in terms of the packet duration. The RTTs at the ingress (iRTT) and egress (eRTT) sides of the switch core may be different. With a buffered switch core, iRTT and eRTT can be decoupled, whereas with a bufferless core they are combined.

4. CONSEQUENCES

We discuss the main consequences that have emerged from the trends observed in Sec. 2. These are typically the result of a combination of several trends.

4.1 Physical system size

Trends 1, 3, and 6 culminate in the following consequence:

Consequence I. Switch fabrics beyond 1 Tb/s aggregate throughput experience a disruption. Single-rack packaging is no longer possible, implying larger systems, cables replacing backplanes, and more transmission power.

This entails significant physical system-packaging issues, see Sec. 4.3, but also immediately gives rise to Consequence II:

Consequence II. Switches have become critically dependent on serial I/O technology that is *both* high-speed *and* high-density and can cross backplanes and cables. These links must be available in the ASIC technology library used by the switching chips and must be very low power because of Trend 4. Owing to transmission line effects combined with power limitations, the I/O technology has not kept pace with the line rates. Therefore, the number of links required to implement a given system capacity has grown substantially. As a result, cabling accounts for a significant part of the total system cost.

4.2 Increased RTT

The physical distance implied by Consequence I combined with the shrinking packet duration due to Trend 5 immediately leads us to Consequence III:

Consequence III. The switch-fabric-internal RTT has significantly increased in terms of the minimum packet duration.

A large RTT only used to be an issue from node to node, but it has now become important also within the switch fabric. Table 2 lists RTT values for four switch-fabric generations (from OC-12 to OC-768), assuming a minimum packet length of 64 B. The RTT is expressed in packet duration, and includes the contribution of both the time of flight (over either backplane or cable) and the pipeline logic and serialization/deserialization (serdes) delay. Note that the latter contribution is at least as important as the former.[3]

Table 2: RTT Expressed in Packet Duration

	Switch generation			
Line rate	OC-12	OC-48	OC-192	OC-768
Conn. dist	1 m, backpl.	1 m, backpl.	6 m, cable	15 m, cable
Pkt. dur.	512 ns	128 ns	32 ns	8 ns
RTT	$\ll 1$	~ 1	16	64

The increased RTT results in more packets in flight, and, independently of the switch architecture chosen, this needs to be accounted for with buffers in the system to ensure that the system is both work-conserving and lossless. In general, the amount of buffering required to ensure both losslessness and work-conservingness must be scaled proportionally to the RTT. However, because of Trend 6 we cannot simply add buffers at leisure. Moreover, the flow-control mechanism selected has a large impact on buffer sizes. As a result, RTT has become a major design parameter for switch architectures.

4.3 Packaging impacts

A decade ago, we could simply build switch chips with the maximum density permitted by CMOS, and a suitable packaging could always be found. Consequences IV and V express the different approach that is required nowadays:

Consequence IV. Now, because of Trend 3, the physical packaging of the systems must be established first, and the switch architecture must be designed to fit these constraints. Owing to Trend 1, long cables are now needed, but because of Trends 2 and 4, adding external cable drivers at leisure is not desirable, for power and cost reasons.

Consequence V. Because of Trend 4, the objective must be to avoid intermediate drivers and drive cables directly from the switch chip.

Packaging examples

Figure 2 shows packaging for a generic 4 Tb/s system, using two racks each with two shelves of 18 cards, with cables connecting the racks. In general, to prevent link blocking, the bandwidth between two racks must equal the maximum bandwidth that can be sourced and sunk by all the line cards

[3]For short and medium range cables.

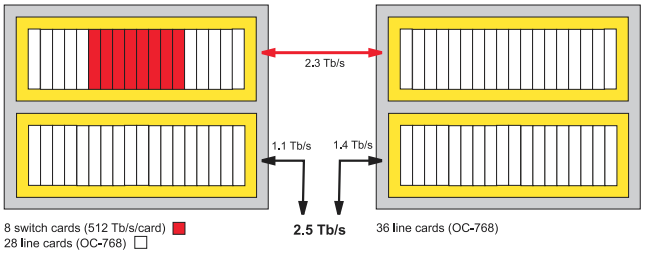


Figure 2: Generic 4 Tb/s system packaging (1.6x speed-up), fitting 64 OC-768 line cards and eight 512 Gb/s switch cards into two racks with two shelves per rack and 18 slots per shelf.

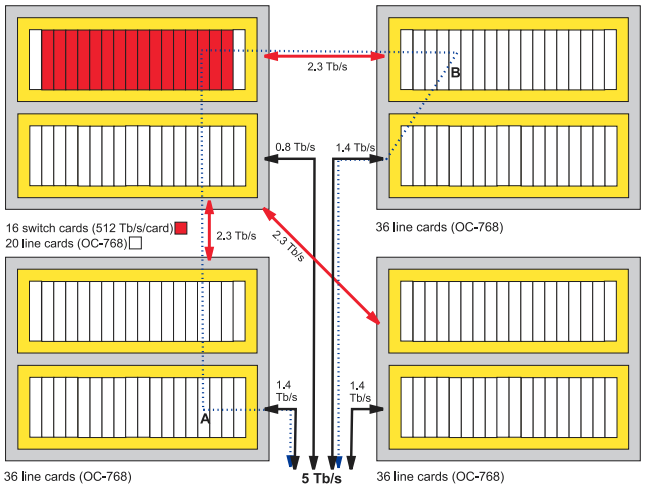


Figure 3: Generic 8 Tb/s system packaging (1.6x speed-up), fitting 128 OC-768 line cards and 16 512 Gb/s switch cards into four racks with two shelves per rack and 18 slots per shelf.

together in one rack, in this case 36 × OC-768 bidirectional. Figure 3 shows an 8 Tb/s system packaged in a similar fashion in four racks. The dotted line shows the path taken by a packet from line card A to line card B.

4.4 Other considerations

The above consequences reflect new requirements for the design of switch fabrics, but most existing requirements also still apply, such as in-order delivery, multicast, losslessness, availability, and the ability to perform nondisruptive repair actions. Of great importance is also the ability to follow an evolutionary migration path that allows as much of the existing investment as possible to be preserved, which requires backward compatibility and the flexibility in the switch core to support both lower and higher line rates.

A robust switch fabric must support any mix of traffic types, which imposes the worst-case requirements of each individual type of traffic simultaneously. Moreover, QoS must be provided for any given number of traffic classes (with widely varying characteristics). First, this has significant implications on buffer dimensioning, and second, as a result of Consequence III, this poses new challenges for intra-fabric flow control. Existing work in the area of link-level flow control, e.g. [7, 8], can most likely be adapted for this purpose.

5. DISCUSSION

The consequences discussed in Sec. 4 have a number of important architectural implications for practical packet-switch designs. Here, we will point out some of these for multi-Tb/s implementations of two popular architectures, namely, the input-queued architecture with virtual output queuing (VOQ), e.g. [9], and no speed-up, and the combined-input-and-output-queued (CIOQ) architecture with limited (2 to 4 times) speed-up, e.g. [10]–[12], both with centralized scheduling. We do not aim for completeness in the coverage of either all currently popular architectures or of all the implications for each architecture, but want to point out the more salient issues that need to be addressed.

5.1 Input-queued (VOQ) with centralized arbitration

In a purely input-queued architecture with VOQ a centralized arbitration unit (scheduler) is required. The architecture consists of three basic parts, which can straightforwardly be mapped to the generic architecture shown in Fig. 1: the input line cards containing the VOQs, the routing fabric (usually a crossbar or parallel crossbars), and the scheduler. We assume that the switch core comprises both the routing fabric and the scheduler. Consequence I implies that multiple racks are required; therefore, at least some, possibly all, of the line cards are at a significant distance from the switch core. The RTT resulting from this distance has some interesting implications. There are two basic approaches to address this issue:

First, the VOQs can be brought as close as possible to the core, either on the same card as the scheduler or on other cards in the same rack. Both options suffer from two drawbacks: First, N extra chips with VOQ, buffering, routing lookup, and flow control are required, thus inefficiently duplicating functionality already implemented in the adapters on the line cards, and, second, Consequence V is violated because the extra chips add to the overall power consumption. Moreover, the first option is clearly not practical except for very small systems because of the space, power, and bandwidth limitations of the card, whereas the second option is very costly because many extra cards (and possibly extra racks) are required.

In the second approach, the VOQs and their associated buffers remain on the line cards, and the scheduler maintains VOQ state information for all N^2 VOQs in the system. The VOQs do not send requests to the scheduler, but rather communicate the arrivals (encoded as a tuple consisting of packet source, destination, and traffic class).[4] The scheduler performs the bookkeeping, computes the matchings based on its local image of the VOQ state information, and sends the corresponding grants.

However, the scheduler is not aware of the arrivals in the last RTT/2 packet cycles, and is therefore likely to compute a

[4]Such an incremental state-update scheme raises the issue of error robustness.

sub-optimal matching, even if the actual matching algorithm is optimal. The impact of the RTT on performance in such a system under various types of traffic requires further study.

Furthermore, in an architecture with a centralized scheduler, communication with all line cards must be tightly coupled because every matching is one-to-one and the core is bufferless, i.e., the request-grant-accept process must be synchronized on a system-wide level. This implies that the RTT between card and switch core must be equal for all line cards. This can be achieved either by using equally long cables or by compensating for cable-length differences by means of delay lines, where all links must be matched to the *longest* cable.

This architecture also suffers from increased latency because packets at the ingress cannot proceed before the request-grant-accept process has been completed. As a result, even packets arriving at an empty input without any contention for their destination have to wait for at least one RTT until the grant arrives.

5.2 CIOQ with limited speed-up

CIOQ architectures with a limited speed-up typically comprise VOQs at the ingress, a bufferless switch core with a centralized scheduler, and output queues at the egress. The switch core runs at a speed-up S times faster than the external line rate, where S is typically between 2 and 4.

The main implication for this type of architecture is that the chip and card bandwidth bottleneck problem is exacerbated by a factor of S because the entire core, including the internal links to the line cards, must be run S times faster (see Fig. 1). Given Consequences II and V this is prohibitively expensive in both hardware (TX/RX macros and cables) and power. Roughly speaking, beyond 1 Tb/s the physical size of the core must grow by a factor of S to accommodate the additional bandwidth.

A possible solution is integrate the VOQs (ingress side) and/or the output queues (egress side) and the switch core, but again this is only feasible for small systems because of space and power limitations.

This architecture also suffers from a possible performance penalty because of the RTT between scheduler and line cards, and faces the same system-wide synchronization challenge as mentioned in Sec. 5.1.

Finally, the introduction of a significant RTT has implications for the egress buffer sizing. Even if there is no internal speed-up (as in Sec. 5.1), egress buffers usually are still necessary to accommodate packets when the downstream links are blocked (e.g., because of congestion in the downstream node). However, if the downstream link is 100% available, any input should be allowed to utilize this bandwidth fully, without the possibility of packets being lost, i.e., under completely unbalanced traffic both work-conservation and losslessness must be ensured. The egress buffer size per output must be scaled proportionally to RTT, speed-up, and number of traffic classes to satisfy these requirements.

6. CONCLUSION

The traditional approach to switch design puts the high-level architecture first and optimizes for performance at this level. The resulting architecture is then mapped to a physical packaging. Although this approach has worked well so far, we have identified a number of trends that, when combined, present a severe disruption at about 1 Tb/s. Beyond that throughput, switch fabrics must be distributed, which introduces a critical dependency on link technology and causes the round-trip time to become an intra-switch issue. However, most importantly, we argue that the switch architecture must be developed starting from the packaging and power requirements, to which the desired architecture must then be suited, instead of the other way around.

7. REFERENCES

[1] A.G. Wassal and M.A. Hasan, "Low-Power System-Level Design of VLSI Packet Switching Fabrics," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 20, no. 6, Jun. 2001, pp. 723-738.

[2] http://www.telcordia.com, "Network Equipment-Building System (NEBS) Requirements: Physical Protection," GR-63-CORE.

[3] http://www.telcordia.com, "Electromagnetic Compatibility & Electrical Safety," GR-1089-CORE.

[4] J. Bolaria and B. Wheeler, "A Guide To Switch Fabrics," Feb. 2002, The Linley Group.

[5] H.B. Bakoglu, "Circuits, Interconnections, and Packaging for VLSI," Addison-Wesley Pub. Co., Reading, MA, 1990.

[6] http://www.npforum.org

[7] H.T. Kung and A. Chapman, "The FCVC (Flow Controlled Virtual Channel) Proposal for ATM Networks," in *Proc. Int. Conf. Network Protocols*, San Francisco, CA, Oct. 1993, pp. 116-127.

[8] C.M. Özveren, R. Simcoe, and G. Varghese, "Reliable and Efficient Hop-by-Hop Flow Control," *IEEE J. Sel. Areas Commun.*, vol. 13, no. 4, May 1993, pp. 642-650.

[9] McKeown, N., "The iSLIP Scheduling Algorithm for Input-Queued Switches," *IEEE/ACM Trans. Networking*, vol. 7, no. 2, Apr. 1999, pp. 188-201.

[10] I. Stoica and H. Zhang, "Exact Emulation of an Output Queueing Switch by a Combined Input Output Queueing Switch," in *Proc. 6th IEEE/IFIP IWQoS '98*, Napa Valley, CA, May 1998, pp. 218-224.

[11] S.-T. Chuang, A. Goel, N. McKeown and B. Prabhakar, "Matching Output Queueing with a Combined Input Output Queued Switch," *IEEE J. Sel. Areas Commun.*, vol. 17, no. 6, Jun. 1999, pp. 1030-1039.

[12] J.G. Dai and B. Prabhakar, "The Throughput of Data Switches with and without Speedup," in *Proc. INFOCOM 2000*, Tel Aviv, Israel, Mar. 2000, vol. 2, pp. 556-564.

Design Guidelines for Robust Internet Protocols

Tom Anderson* Scott Shenker† Ion Stoica‡ David Wetherall*

Abstract

Robustness has long been a central design goal of the Internet. Much of the initial effort towards robustness focused on the "fail-stop" model, where node failures are complete and easily detectable by other nodes. The Internet is quite robust against such failures, routinely surviving various catastrophes with only limited outages. This robustness is largely due to the widespread belief in a set of guidelines for critical design decisions such as where to initiate recovery and how to maintain state.

However, the Internet remains extremely vulnerable to more arbitrary failures where, through either error or malice, a node issues syntactically correct responses that are not semantically correct. Such failures, some as simple as misconfigured routing state, can seriously undermine the functioning of the Internet. With the Internet playing such a central role in the global telecommunications infrastructure, this level of vulnerability is no longer acceptable.

In this paper we argue that to make the Internet more robust to these kinds of arbitrary failures, we need to change the way we design network protocols. To this end, we propose a set of six design guidelines for improving the network protocol design. These guidelines emerged from a study of past examples of failures, and determining what could have been done to prevent the problem from occurring in the first place. The unifying theme behind the various guidelines is that we need to design protocols more defensively, expecting malicious attack, misimplementation, and misconfiguration at every turn.

1 Introduction

Robustness has been, from the very beginning, one of the central design goals of the Internet [9]. As a result, the Internet architecture has been highly resilient to various kinds of catastrophes; the Internet has survived hurricanes, earthquakes, tunnel fires, and terrorist attacks with only temporary and partial loss of end to end connectivity [26]. To a large extent, this success story can be attributed to Internet protocol designers following a set of guidelines for robust design: e.g., end hosts should be responsible for error recovery, failures should be assumed to be the common case, and all critical state should be refreshed periodically.

However, the vaunted robustness of the Internet does not apply to all varieties of failures. An implicit but fundamental assumption underlying the early notion of robustness, and hence the design of many Internet protocols, is that systems are "fail-stop"—when systems fail they completely and detectably stop working. Thus, Internet designers have long assumed that nodes can recognize that another node has failed by either absent or syntactically incorrect (malformed) protocol responses.

Unfortunately, not all failures are so cleanly defined and easily detected; either by malicious intent or by accident, systems obeying the syntax of a protocol may in fact be behaving incorrectly. These more *arbitrary* failures [11] occur with surprising regularity, in part because the increasing scale and heterogeneity of the Internet makes subtle inconsistencies more likely. Many different protocol implementations, operational practices, and user motivations must somehow safely coexist for the Internet to be highly robust.

Moreover, these arbitrary failures can often cascade through the Internet with serious consequences to parties not at fault. In one recent example, widespread outages were triggered because one vendor's BGP implementation ignores but propagates incorrect route announcements, while another vendor's routers terminate any BGP session propagating an obviously incorrect announcement [18]. Simple operator errors in router configuration have recently led to several significant disruptions to Internet connectivity [14, 23]. As a final example, the CodeRed and Nimda worms triggered widespread routing problems due to previously unknown BGP instabilities [10]. To put these incidents in perspective, each arguably caused a more serious Internet outage than the September 11 terrorist attacks in which a large amount of connectivity through New York City was severed by the collapse of the World Trade Center.

We argue that to make the Internet robust in the face of these arbitrary failures, we need to fundamentally rethink the way we design network protocols. We show by example that some protocol designs behave better than others when there are bugs, mistakes, and attacks, even though they are no less efficient or scalable. We examine these case studies in an attempt to draw lessons as to how we should better design protocols in the future. While some of these lessons have already begun to be applied in an ad hoc manner to individual protocols, we argue that we should apply them systematically to all protocols that need to survive arbitrary failures.

Our overarching recommendation is that we should design protocols *defensively*, keeping in mind that other nodes are not to be trusted. We identify six guidelines within this overall theme, and demonstrate by example that robustness could have been significantly improved had those guidelines been applied systematically in the past. Of course, the need for defensive design is not unique to networks—software engineers apply defensive programming to reduce bug rates [19], and operating systems apply defensive programming to isolate errors to individual applications. However networks do pose some significant challenges not found in other disciplines: network protocols often survive and

*CSE Department, University of Washington, e-mails:{tom,djw}@cs.washington.edu. This work was supported in part by DARPA Contract Number F30602-00-2-0565. A more complete version of this paper is available at: http://www.cs.washington.edu/homes/tom/design.pdf

†ICSI Center for Internet Research, e-mail: shenker@icsi.berkeley.edu.

‡EECS Department, Univeristy of California, Berkeley, e-mail:istoica@cs.berkeley.edu. This work was supported in part by NSF Contract Number NSF-ITR 0085879.

evolve for decades, they are typically implemented by numerous separate organizations, and they often must work across organizational boundaries. These requirements make it all the more difficult and all the more important to design protocols to be robust against arbitrary failures of the participants.

2 Related Approaches

There already exist some approaches that help protect against arbitrary failures. This is because robustness in its many guises has been a longstanding design goal for distributed systems. In particular, cryptographic authentication, fault-tolerance via consensus, and formal (and informal) protocol specification are extremely useful, and should be used whenever appropriate. However, in this section we argue that by themselves they do not suffice. We do so by exploring a motivating example, an early ARPANET routing incident. We first present the failure and then consider the utility of each of the above approaches in preventing it.

Routing in the early ARPANET was based on a link-state design. In link-state protocols, the advertisements sent by a node carry increasing sequence numbers that are used to distinguish a new advertisement from old ones; routers forward new advertisements to all of their neighbors, and silently discard old ones. In the ARPANET scheme, a "modulo" sequence number space was used to avoid the problem of becoming stuck when the maximum sequence number was reached. That is, the sequence number counter simply wrapped, and the test to determine whether one sequence number was larger or smaller than another was whether it was a shorter distance forward or backward in sequence number space, respectively.

This scheme worked well when all the nodes were functioning correctly. However, a network-wide storm was caused by a malfunctioning router that injected a series of messages with incorrect sequence numbers before it crashed. The faulty messages had sequence numbers that formed a cycle of progressively "larger" advertisements. That is, the sequence numbers increased by jumps, wrapping as necessary, with each growing "larger", yet the first in the sequence was "larger" than the last. This led to an infinite sequence of updates, forwarded in a cycle to all nodes in the network, without any further input on behalf of the malfunctioning node. To address this problem, the network had to be entirely purged of the faulty messages—not a pleasant thought were a similar self-sustaining chain of events to occur in the Internet today. This is precisely the kind of unanticipated critical failure that we would like to preclude by more robust design.

How could this failure have been prevented? Clearly, cryptographic security mechanisms would have been of no help in this incident. In general, encryption-based authentication is extremely important for helping prevent malicious attackers from hijacking protocol communication. Passwords and encryption can be used to validate that only authorized users or machines are allowed to participate in a protocol, eliminating a whole class of potential problems. By itself, however, strong authentication is not sufficient for completely eliminating arbitrary failures. Authentication can demonstrate that the speaker is who she says she is, but it cannot validate the semantic content of the protocol information—the speaker could have been compromised or could be simply behaving incorrectly. In the ARPANET case, authentication would not have prevented the malfunctioning router from sending (authentic) messages with improper sequence numbers.

Approaches for achieving fault tolerance via consensus are also not a complete solution. There is a vast literature on the theory of algorithms for achieving Byzantine agreement [17] as well as recent work on more practical systems [6]. However, these solutions typically depend on the ability to replicate a computation and then run an agreement algorithm to determine the correct answer. In some cases, it can make sense to replicate state and computation, and hence apply consensus techniques. However, in many protocols it does not appear feasible to do so while preserving efficiency. For example, the computation that causes a node to issue new routing advertisements depends on the state of the attached links, state which is available only locally.

Finally, we consider whether better or more precise specifications could have helped with the ARPANET incident. Unfortunately, the fact is that there is nothing wrong with the protocol when it is implemented and operated as specified—it does converge to the available routes as expected. It was only when a malfunctioning node injected corrupt data that an unanticipated, cascading error was induced. Our point is that despite being correct, the protocol is fragile or vulnerable to arbitrary failures, which makes it an undesirable design. Thus, even if we posit better tools for checking the correctness of implementations before they are applied to live systems [15], we also need to check whether the protocol itself is vulnerable if some nodes are misimplemented or malicious. Formal methods that automatically check specifications for vulnerabilities— while a promising avenue for future research—are not yet in widespread use beyond security protocols [5].

As an aside, the actual solution adopted in the ARPANET routing protocol was to revise the design to remove modulo sequence numbers; modern routing protocols such as OSPF and IS-IS follow this lead. Linear sequence numbers were used instead, with a separate timeout-based mechanism (that is, aging) for resetting when the maximum sequence number is reached. With this design, no advertisement can be propagated unless it was recently injected by some router.

In sum, the traditional robustness techniques of authentication, consensus, and specification are invaluable in modern distributed systems, and Internet protocols should incorporate them whenever applicable. However, these techniques are not sufficient by themselves to make Internet protocols robust against arbitrary failures. We believe that we must complement these techniques with additional guidelines for protocol design to protect against arbitrary failures.

3 Design Guidelines for Robust Protocols

The Internet is an immense and diverse distributed system. It spans many different administrative domains, incorporates a wide variety of underlying technologies, and supports a vast and rapidly evolving array of applications. One cannot approach such a system with a rigid set of narrow rules without killing the vitality that has been at the heart of the

Internet's success. This was recognized by the early designers of the Internet, who, instead of enforcing rigid rules, worked with a set of much more general and imprecise guidelines that offer advice on basic design decisions such as how and where to keep state or initiate recovery. Examples include endpoint error recovery, critical state should be refreshed periodically, and implementations should be engineered to interoperate with non-conforming peers.

The main purpose of this paper is to call for the identification of a similar set of guidelines to improve robustness against arbitrary failures. The overarching philosophy of the guidelines we propose is that one should *design defensively*. That is, everpresent in the design process should be a recognition that incoming information might be incorrect, and that nodes might be malicious or broken. While one wants to stop short of a paralyzing paranoia, protocols should always adopt a cautious skepticism. Of course, we are not the first to propose defensive design, and many protocols already exhibit this, albeit in an ad hoc fashion. Our goal is to synthesize and generalize the underlying ideas behind some of the better protocols that have been recently proposed, in the hope that making these guidelines explicit will drive the design of future protocols.

Our first guideline is perhaps the least controversial but one of the hardest to achieve in practice: use very clean and simple interfaces that do not overload mechanism with multiple functions, have clean functional (not procedural) semantics, and do not embed performance optimizations in the protocol definition.

Guideline #1: *Value conceptual simplicity*

The natural tendency is for interfaces to become increasingly complex over time, as designers evolve systems by adding features in a backwardly compatible way. But this can lead to problems! It is much harder for multiple parties to correctly implement complex designs, and unforeseen interactions between components can hide potentially devastating vulnerabilities.

One illustration of the effect of complex semantics is the persistent route oscillations that can arise in BGP [22]. In BGP, a router selects its best route among possible alternatives according to an ordered decision criteria, e.g., the highest "local preference" before the shortest AS path length. One of these criteria is the Multi-Exit Discriminator (MED), which is used by one ISP to specify ingress preferences to another. This provides a traffic engineering mechanism for overriding "hot-potato" or "early-exit" routing. Unfortunately, MED has semantics that differ from the other attributes. The other attributes are straightforward in that they can be used to order and directly compare routes, so that the best route may be selected by simply selecting amongst best candidates. MED, on the other hand, cannot be used to order routes for selection directly because MED values are only comparable when learned from the same neighboring AS. This can result in, for example, a route with a MED causing the current best route to be dropped, yet without the MED route becoming the new current best route. This means that with MEDs, all candidate routes must be gathered and then selected in a single pass.

This dependence of routes on each other can manifest itself as oscillations when route reflectors [2] are used. Here, we discuss only the overall problem and refer readers to [22] for details. To understand the problem, note that BGP routers within an AS must coordinate with each other to make consistent route selections. This was originally achieved by having every BGP router peer with every other router, a configuration known as a BGP mesh in which all routers receive all information and run compatible decision procedures in parallel. However, meshes are inherently unscalable to large ASes. A route reflector is a more scalable alternative in which a central location, the route reflector, receives information from and redistributes best route information to all BGP routers. Unfortunately the premise behind route reflectors—scalability via information hiding—conflicts with the semantics of MEDs because all information is required at each router to ensure consistency. The result can be that as routes propagate through the system they cause other routes to change in a vicious cycle. This is handled today by operational guidelines that discourage vulnerable configurations, but arguably a better factoring in the protocol design would have avoided the problem in the first place.

Our second guideline is less obvious: if other nodes are potentially untrustworthy because they could be misimplemented or even malicious, then protocols should be explicitly designed to reduce each node's circle of trust to the minimal set of information required to achieve the protocol's purpose. Caution in trusting other nodes is merely prudent in a large and heterogeneous distributed system like the Internet.

Guideline #2: *Minimize your dependencies*

When protocols are being initially designed, it is often more convenient to assume the other nodes participating in a protocol are trustworthy, but that trust is often misplaced. The problems often manifest themselves not when the protocol is first being deployed, but later as the protocol is used by a wider population with differing motivations and security policies. For example, the TCP congestion control mechanism relies on information provided by the receiver as to which packets arrived correctly. Although it seems natural that the TCP sender and receiver need to cooperate to transfer data, the protocol is defined in such a way that a sender must simply trust the congestion control information being provided by a receiver. This allows malicious receivers to trick today's Web servers into sending at rates far above TCP-friendly rates [27].

As one example vulnerability, the TCP fast recovery mechanism uses duplicate acknowledgements to first trigger a retransmission of the missing packet and then to trigger additional sends, reasoning that each duplicate acknowledgment implies that another packet has left the network. This opens the door for an attack—an unscrupulous receiver can send an infinite stream of duplicate acknowledgements! Indeed, since the IP packet delivery model allows duplication, this is an undetectable error in the existing TCP protocol specification. A simple fix is to require selective acknowledgements (SACK [21]). These are more robust than duplicate acknowledgements because they can be designed to be unambiguous and idempotent in their effects.

Of course, one cannot avoid all dependencies in protocols. In cases where a node has to rely on information from another node, the protocol designer should add mechanisms into the protocol to allow for verification whenever possible, for example, by actively testing the node's responses or by

comparing the data to the information provided by other nodes.

Guideline #3: *Verify when possible*

Note that explicitly adding redundancy into protocols conflicts with the common practice of stripping out all redundancy as superfluous. At times the desire to verify information may conflict with the desire to reduce complexity; that is, verification may require information that is not necessary for the basic functioning of the protocol. How to best balance these two competing goals–reducing complexity and collecting enough information to reliably verify—will depend on the context.

As a concrete example, consider Explicit Congestion Notification (ECN) [25]. With ECN, routers mark packets by setting an ECN bit in the packet header when they become congested. The marks are then returned from receiver to sender via acknowledgement packets. If the ECN bit is set, the server interprets it as a congestion indication and reacts accordingly by reducing the congestion window. This mechanism has the advantage of signaling the congestion early without causing loss.

However, it was observed in [13] that signaling congestion with marks is not as robust as signaling with drops. Once a packet is dropped, it cannot be "undropped", at least not in a manner that achieves reliable transport. This makes it highly likely that drops will be observed by the sender. Yet with marks, the receiver could fail to return the mark signal to the sender, or the signal could be stripped off by devices along the network path between the marking router and original sender (e.g., firewalls that normalize IP header bits or VPN boxes that strip off an outer IP header). This kind of arbitrary failure can result in a flow obtaining much more than its fair share of the bandwidth. And the fact that a receiver that fails to return mark signals might receive significantly more bandwidth encourages this form of failure.

A solution to this problem (described in [13]) is to revise the design to use nonces, in a manner that generalizes the TCP duplicate acknowledgement example above. Packets carry a single-bit nonce that is erased by marking routers to signal congestion. Receivers echo the nonce sum in acknowledgements to the sender along with their indication of whether the packet was marked. (The sum is used to cover sequences of packets because acknowledgements can be lost.) When marks are not being signaled, this sum gives the sender confidence that the packets were in fact received and that they were not marked, *i.e.*, that congestion has not been signaled. This is a strong result because it depends on no assumptions about receiver behavior for its correctness.

An important observation is that this verification mechanism is lightweight. It uses few header bits and little state and computation at end systems to detect misbehavior probabilistically. The chance that any incident of erasing a mark will be detected is 50%, as only one bit of check information is used. However, because nonces are random, each loss represents an independent trial. This means that repeated misbehavior will be detected quickly.

Our fourth guideline concerns unsolicited requests by unauthenticated parties; this can leave systems vulnerable to intentional or accidental resource exhaustion. Recent denial of service attacks are just one example of this. Careful planning is needed to keep the resources of one node from being at the mercy of other nodes:

Guideline #4: *Protect your resources*

Although this may seem intractable, the potential for resource exhaustion can be greatly reduced through careful protocol design. Consider the example of the TCP connection setup. Connection state is established on receipt of an initial SYN packet and must be maintained until the three-way handshake and subsequent connection completes, or until the connection attempt is timed out. This behavior has been exploited by "SYN flood" denial-of-service attacks that send a flood of initial SYN packets (typically with spoofed source addresses) and never complete the handshake. A SYN flood can exhaust connection resources at a server and thereby cause legitimate traffic to be shut out [7]. This problem is not readily addressed by verification because the end-system receiving a TCP connection request cannot determine, at the time a SYN packet arrives, whether the packet is part of a legitimate connection establishment sequence. The vulnerability arises because the server allocates resources to connection attempts that are not yet known to be valid.

Fortunately, a simple re-design of the connection setup protocol can protect the server's resources by shifting the resource burden. Instead of keeping state on the server, return the state to the initiating client, since they are the party for whom the state is being maintained. This can be achieved by returning the state as part of the SYN-ACK, and later have the client re-supply the server with missing state when the handshake is completed. Assuming there is an inexpensive way to verify the integrity of the re-supplied state, the server will not be left holding incomplete or incorrect connection state. SYN cookies is a backwards-compatible Linux implementation of this more robust connection setup protocol [3]. In SYN cookies, the initial sequence number (ISN) is used to encode and return connection state to the client, and the encoding is based on a secure hash that is efficient to verify.

The above philosophy can be applied to protect the local resources (computation, buffer space, connection state, bandwidth, and so forth) that can be consumed in response to messages received from parties whose behavior is uncertain. In fact, the cookie technique is quite general, being originally used in the design of Photuris [16], a protocol used for establishing shared session keys, to avoid accumulating state during the key exchange.

Fifth, we observe that since errors cannot always be prevented or caught, we must also design protocols to limit the possible damage resulting from incorrect behavior. We would like to prevent instabilities from occurring in the first place, but if they do occur, we also want to make sure the effects do not cascade out of control.

Guideline #5: *Limit the scope of vulnerability*

Again, this can seem intractable, yet oftentimes very simple techniques can protect systems against broad classes of unforeseen errors. Consider the example of route flapping. Until fairly recently, router and link instabilities could cause serious load issues for BGP because routes were "flapping".

The problem was that, at the BGP level, route announcements and withdrawals were essentially propagated throughout the world, so that every location saw the cumulative sum of all route flaps in the entire Internet. To decrease the extent to which local link and router instabilities effect distant networks, a Route Flap Damping [12] mechanism has been added to BGP. It works by holding down routes that are repeatedly withdrawn and then announced, with infrequent changes resulting in no hold-down, and oscillation causing the greatest amount of hold-down. Every router that implements damping thus creates a barrier separating oscillations from the rest of the network.

Another recent example is BGP error processing. The BGP specification requires that an incorrect announcement causes the entire session to be dropped and restarted, since error checking is not provided at the announcement level (as is done in OSPF and IS-IS). However, the result is that one bad announcement can cause the entire routing session (that is roughly 100K announcements today) to be dropped and restarted. This was compounded when one vendor chose to pass bad announcements rather than drop the entire session, while other vendors dropped sessions as required [8]. In June 2001, a single operator's misconfiguration tickled this bug: it was propagated to a substantial fraction of the Internet, where it caused many sessions to be dropped, resulting in Internet-wide outages.

Finally, we observe that, as with the previous example, some robustness problems can lie dormant, only being triggered by other errors that occur in the system. Thus, making systems more robust may also require that system designers and operators seek out and fix errors before they begin to cascade. Unfortunately, there is a tension here. If systems can continue operating in the face of failures, errors can persist indefinitely without anyone having the visibility or motivation to fix the underlying problem. If left untreated, however, failures can combine with severe unforeseen consequences. Thus, we end with our sixth guideline:

Guideline #6: *Expose errors*

As an example, the recent investigation of TCP checksum failures resulted in the discovery that data was being corrupted at hosts or routers [28]. Because these failures were discovered at the receiver and then ignored (once the packet was discarded) the sending host received no indication that anything was amiss, other than a negligible increase in the rate of retransmissions. As another example, researchers have recently found a flaw in a common BGP configuration strategy by which ISPs connect to their customers [20]. ISPs often filter routes advertised by their customers, to prevent them from using the ISP to provide transit to other customers. The flawed configuration filters the customer routes based on their prefixes. This works correctly when there are no additional failures. However, if the customer is multihomed, and the link to the customer fails, the filtering rules may result in an ISP providing unintended transit to the customer's other ISP. This can be avoided by repairing the latent misconfiguration, filtering on AS-PATH rather than prefixes. Again, the more fundamental problem is that the protocol is designed in such a way that an operator cannot easily tell if they have a flawed configuration without physically causing the failure.

4 Discussion

In this paper, we have argued that it is time to systematically rethink protocol design to defend against arbitrary failures – implementation bugs and other failures that, through error or malice, are not fail-stop as are simple link and node failures. We have discussed and categorized several examples of problems and solutions to argue three main points. First, arbitrary failures occur surprisingly often and can result in significant disruptions to the Internet; consider the TCP bugs that have proved to be pervasive and BGP vulnerabilities that have caused widespread outages. Hence arbitrary failures must be tackled if we are to significantly improve the dependability of the Internet. Second, arbitrary failures are not sufficiently well addressed by existing bodies of work. For example, authentication helps determine which party sent a message but not whether the contents of the message make sense. Third, we argue that the long-term solution is to develop a set of design guidelines to help protocol designers defend against arbitrary failures. Design guidelines for robustness have been extremely successful at providing a solid foundation for tolerating fail-stop failures. Fail-stop failures are regularly considered during the design stage and concepts such as soft-state are applied systematically. In contrast, arbitrary failures tend to be viewed as isolated incidents and dealt with in an ad hoc fashion.

As the Internet evolves into a global communication infrastructure that encompasses all facets of the social and economic life, we can no longer ignore the impact of these arbitrary failures on the Internet. To make the Internet robust in the presence of these failures, we need to fundamentally change the way we design network protocols. In particular, we put forward a candidate set of guidelines to be used in future protocol designs. We presented examples that provide an existence proof that protocols can be designed to withstand arbitrary failures, and that these designs can be essentially as efficient and scalable as designs that do not tolerate arbitrary failures. If we are to make the leap from ad hoc treatment of arbitrary failures to systematically applying the principles of defensive design, then we must be able to extract lessons from one protocol setting and apply them in another.

We should note that some of our guidelines can be seen as modifying previously established guidelines for designing Internet protocols and implementations to be robust against fail-stop errors. It is possible that the guidelines that helped the Internet gain its success may need circumscribing now that the Internet is so widely used by such a heterogeneous community. Specifically, one popular guideline is to "Be liberal in what you accept, and conservative in what you send" [24, 4]; this is key to achieving quick, low cost interoperability – by not insisting that every implementation exactly follow the specification, we enhance the likelihood that two different implementations will work together. However, it can sometimes reduce a node's ability to protect itself from a misbehaving implementation or even a malicious attacker; a more nuanced version might suggest "be liberal in what you accept but be conservative in what you believe" [1]. Likewise, end-to-end error recovery mechanisms are clearly key to end-to-end robustness in a distributed system like the Internet where failures are the common case. However, those recovery mechanisms can hide persistent problems that can lurk uncorrected until they cause other, sometimes worse,

problems. Thus we believe end-to-end recovery must be complemented with mechanisms to expose errors so that they can be fixed.

Finally, all our effort will be for naught unless the guidelines that are ultimately developed are used in practice. Thus we hope to raise the level of consciousness of arbitrary failures during the period that protocols are being designed. One suggestion we would like to make is that every RFC contain a "Robustness Considerations" section that answers a simple question: What would go awry if the protocol messages are corrupted? We hope that such a "Robustness Considerations" section, with its focus on a simple and direct question, will be of more use than "Security Considerations" sections have been in the past.

References

[1] Martin Abadi and Roger M. Needham. Prudent engineering practice for cryptographic protocols. *Software Engineering*, 22(1), 1996.

[2] T. Bates, R. Chandra, and E. Chen. BGP route reflection - an alternative to full mesh IBGP. RFC 2796, IETF, Apr. 2000.

[3] D. J. Bernstein. SYN cookies. `http://cr.yp.to/syncookies.html`, 1996.

[4] R. Braden. Requirements for Internet hosts – communication layers. RFC 1122, IETF, Oct. 1989.

[5] Michael Burrows, Martin Abadi, and Roger Needham. A logic of authentication. *ACM Transactions on Computer Systems (TOCS)*, 8(1), Feb. 1990.

[6] Miguel Castro and Barbara Liskov. Practical Byzantine fault tolerance. In *Operating Systems Design and Implementation (OSDI)*, 1999.

[7] CERT advisory ca-1996-21 TCP SYN flooding and IP spoofing attacks. `http://www.cert.org/advisories/CA-1996-21.html`, Sep. 1996.

[8] Cisco security advisory: Cisco IOS BGP attribute corruption vulnerability. `http://www.cisco.com/warp/public/707/ios-bgp-attr-corruption-pub.shtml`, May 2001.

[9] David D. Clark. The design philosophy of the DARPA Internet protocols. In *ACM SIGCOMM*, Aug. 1988.

[10] Jim Cowie, Andy Ogielski, BJ Premore, and Yougu Yuan. Global routing instabilities during Code Red II and Nimda worm propagation. `http://www.renesys.com/projects/bgp_instability`, Oct. 2001.

[11] Flaviu Cristian. Understanding fault-tolerant distributed systems. *Communications of the ACM*, 34(2), 1991.

[12] Ramesh Govindan Curtis Villamizar, Ravi Chandra. BGP route flap damping. RFC 2439, IETF, Nov. 1998.

[13] David Ely, Neil Spring, David Wetherall, Stefan Savage, and Tom Anderson. Robust congestion signaling. In *9th International Conference on Network Protocols (ICNP)*, Nov. 2001.

[14] Jim Farrar. C&W routing instability. NANOG mail archives, Apr. 2001. `http://www.merit.edu/mail.archives/nanog/2001-04/msg00209.html`.

[15] G. Holzmann. The model checker SPIN. *IEEE Trans. on Software Engineering*, 23(5), May 1997.

[16] Phil Karn and William Allen Simpson. Photuris: Session-key management protocol. RFC 2522, IETF, March 1999.

[17] Leslie Lamport, Robert Shostak, and Marshall Pease. The Byzantine generals problem. *TOPLAS*, 4(3), July 1982.

[18] Matt Levine. BGP noise tonight? NANOG mail archives, Oct. 2001. `http://www.merit.edu/mail.archives/nanog/2001-10/msg00221.html`.

[19] Steve Maguire. *Writing Solid Code : Microsoft's Techniques for Developing Bug-Free C Programs.* Microsoft Press, 1993.

[20] Ratul Mahajan, David Wetherall, and Tom Anderson. Understanding BGP misconfiguration. In *ACM SIGCOMM*, Aug. 2002.

[21] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow. TCP selective acknowledgement options. RFC 2018, IETF, April 1996.

[22] Danny McPherson, Vijay Gill, Daniel Walton, and Alvaro Retana. BGP persistent route oscillation condition. Internet Draft draft-mcpherson-bgp-route-oscillation-01.txt, IETF, March 2001.

[23] Stephen A Misel. Wow, AS7007! NANOG mail archives, Apr. 1997. `http://www.merit.edu/mail.archives/nanog/1997-04/msg00340.html`.

[24] J. Postel. Internet Protocol (IP). RFC 791, IETF, Sept. 1981.

[25] K. Ramakrishnan, Sally Floyd, and D. Black. The addition of explicit congestion notification (ECN) to IP. RFC 3168, IETF, Sep. 2001.

[26] Peter Salus. The Internet under stress. Talk at NANOG 23, Oct. 2001.

[27] Stefan Savage, Neal Cardwell, David Wetherall, and Tom Anderson. TCP congestion control with a misbehaving receiver. *Computer Communication Review*, 29(5), 1999.

[28] Jonathan Stone and Craig Partridge. When the checksum and the data disagree. In *ACM SIGCOMM*, Aug. 2000.

The Power of Epidemics: Robust Communication for Large-Scale Distributed Systems

Werner Vogels
Dept. of Computer Science
Cornell University
vogels@cs.cornell.edu

Robbert van Renesse
Dept. of Computer Science
Cornell University
rvr@cs.cornell.edu

Ken Birman
Dept. of Computer Science
Cornell University
ken@cs.cornell.edu

ABSTRACT

Building very large computing systems is extremely challenging, given the lack of robust scalable communication technologies. This threatens a new generation of mission-critical but very large computing systems. Fortunately, a new generation of "gossip-based" or epidemic communication primitives can overcome a number of these scalability problems, offering robustness and reliability even in the most demanding settings. Epidemic protocols emulate the spread of an infection in a crowded population, and are both reliable and stable under forms of stress that will disable most traditional protocols. This paper describes some of the common problems that arise in scalable group communication systems and how epidemic techniques have been used to successfully address these problems.

1 INTRODUCTION

Distributed computing will be central to advances in a broad range of critical applications, including intelligence information systems, military command and control, air traffic control, electric power grid management, telecommunications, and a vast array of web-based commercial and government applications. Indeed, a massive rollout of such systems is already underway. Yet while impressive capabilities have been easy to develop and demonstrate in small-scale settings, once deployed these systems often stumble badly.

Software that runs securely and reliably in small-scale mockups may lose those properties as numbers of users, the size of the network and transaction processing rates all increase. Whereas small networks are well behaved, any sufficiently large network behaves like the public Internet, exhibiting disruptive overloads and routing changes, periods of poor connectivity and throughput instability. Failures rise in frequency simply because the numbers of participating components are larger. A scalable technology must ride out such forms of infrastructure instability.

Our studies reveal that very few existing technologies have the necessary properties. Most, including the most prevalent commercial software, exhibit scalability problems when subjected to even modest stress. This finding reveals an imminent (and growing) threat to the full spectrum of emergent mission-critical computing systems. If we can't solve the scalability problem, and develop a methodology yielding applications that remain secure and robust even when failures occur – indeed, even under attack, or during denial-of-service episodes – the very technologies that hold the greatest promise for major advances will prove to be the Achilles Heel of a future generation of mission-critical military and public-sector enterprises.

In the past 4 years the Spinglass project has worked to overcome scalability barriers, starting with an idea that was first proposed in the context of replicated database systems. These systems employed what were called "epidemic-style" or "gossip" update algorithms, whereby sites periodically compare their states and reconcile inconsistencies, using a randomized mechanism for deciding when and with whom each participant will gossip. Traditionally, the database systems used gossip protocols at low speeds. Our work employs gossip at very high speeds, yielding a new generation of protocols that have an unusual style of probabilistic reliability guarantees –guarantees of scalability, performance, stability of throughput even under stress, and remarkable scalability. These properties hold even on Internet-like platforms. Epidemic protocols lend themselves to theoretical analysis, making it possible to predict their behavior with high confidence. However, the focus of our work at Cornell is mostly practical: we are using epidemics, together with other more traditional mechanisms, to develop new generations of scalable communications software and middleware for a wide variety of settings.

In the initial toolset epidemic techniques were used for failure-detection, group communication and distributed state-management. Now that these tools are maturing, we are applying the techniques to other distributed systems areas that face similar scalability challenges, notably the areas of sensor networks and world-wide publish/subscribe systems. Our objective in the present paper is to review the scalability problem for group communication, and to summarize how we used epidemic techniques to solve it. The need for brevity limits the technical detail here, but other publications are available for the interested reader who wishes to learn more.

2 SCALABILITY PROBLEMS IN CURRENT COMMUNICATION SYSTEMS

The scalability of distributed protocols and systems is a major determinant of success in demanding systems. For example, consider the September 2000 field-test of the Navy's Cooperative Engagement Capability (CEC). During a two week period this system (which offers an over-the-horizon cooperative targeting capability for naval battleships) was subjected to a very modest stress test. The value of this system depends upon timely identification of threats and rapid decisions about which ship will respond to which threat: threats may be incoming missiles moving at several times the speed of sound. This translates to internal deadlines of about a second for the communication subsystem, which had been demonstrated easily capable of meeting the requirements under laboratory conditions with small numbers of participating computing systems. Yet under load, when even small numbers of battleships were added to the system, the underlying Data Distribution System (DDS) became unstable, failing outright, or delivering data after much more than the one-second threshold. (Defense News, October 16, 2000). The result was that the CEC basically failed – and this under rather benign conditions in which the only variable that changed was the number of participants.

This paper focuses on scalability of distributed protocols providing some form of guaranteed reliability when communication among multiple participants is required. Use of these reliable group communication (multicast) protocols is very popular in distributed systems, as there is a natural match between the group paradigm and the way large distributed systems are structured. These protocols allow systems to be built in pure peer-to-peer manner, removing the need for centralized servers, removing one of the bottlenecks in system scalability.

Traditional reliable multicast protocols, including those developed by our research group, all exhibit severe scaling problems, especially when applied in large Internet-style settings. Even though some of these protocols appeared to be very promising in terms of scalability they all failed to operate as soon as the network conditions were no longer ideal. In general the performance analyses of the protocols only focuses on two extreme cases: performance of the protocol under ideal conditions, when nothing goes wrong, and the disruptive impact of a failure. Reality forces use to take a look at these protocols from a different perspective: what happens to these protocols under mundane transient problems, such as network or processor scheduling delays and brief periods of packet loss. One would expect that reliable protocols would ride out such events, but we find that this is rarely the case, particularly if we look at the impact of a disruptive event as a function of scale. On the contrary, reliable protocols degrade dramatically under this type of mundane stress, a phenomenon attributable to low-probability events that become both more likely and more costly as the scale of the system grows.

Because of the need for brevity, we'll limit ourselves to a summary of our finding with respect to the growth rate of disruptive overheads for a number of widely used multicast protocols. Elsewhere [3], we present a more detailed analysis of a variety of scenarios, modeled after the work of Gray *et. al.* [4], where a similar conclusion is reached with respect to database scalability. It is clear that scalability represents a widespread problem affecting a broad range of technologies and systems.

2.1 Common Problems

When subjected to transient problem that are related to scaling the environment, there are important categories of problems that appear:

- *Throughput Instability*. All of the protocols that implement reliable group communication are subject a breakdown of message throughput as soon as one or more members experience perturbation. A single slow receiver, which can be cause by CPU overhead or some localized message loss, will eventually have an impact on the throughput to the overall group. The results in [3] show that this impact is even more dramatic and happens more rapidly if we scale up the system, making the protocol stability extremely volatile under even moderate scaling conditions.

- *Micropartitions.* In reaction to the throughput instability problem, designers often go for the approach to as soon as possible to remove the trouble-causing member from the group by using more aggressive failure detection mechanisms. However when scaling the system to moderate Internet environments, one quickly discovers that this has the adverse effect that transient network problems, which occur frequently, frequently trigger incorrect failure-suspicions. Erroneous failure decisions involve particularly costly "leave/rejoin" events, where the overall system constantly needs to reconfigure itself. We will term this a *micropartitioning* of the group, because a non-crashed member effectively becomes partitioned away from the group and later the partition (of size one) must remerge. In effect, by setting failure detection parameters more and more aggressively while scaling the system up, we approach a state in which the group may continuously experience micropartitions, a phenomenon akin to thrashing.

  Costs associated with micropartitions rise in frequency with the square of the size of the group. This is because the frequency of mistakes is at least linear in the size of the group, and the cost of a membership change is also linear in the group size: a quadratic effect.

- *Convoys*. An obvious response to the scalability problems just presented is to structure large systems hierarchically, as trees of participant groups. Unfortunately this option is also limited by disruptive random events, albeit in a different way. Experimentation has shown that such a tree structure, when confronted with moderate network instability, exhibits an amplification of the burstiness of the message traffic. Even though messages enter the system at a steady rate, the reliability and buffer strategies at each of the intermediate nodes in the trees have compressing effects on messages rates, especially when confronted with message loss. The term "convoy" has been used by the database community to describe this phenomenon, which is also well known to the packet routing community.

2.2 Unsuccessful Solutions

Some reliable multicast protocols have been successful at larger scale, but only by limiting the functionality of the protocols. Techniques the developers have resorted to achieve scalability are:

- *Anonymous Membership*. The protocol basically streams out information to whoever wants to receive messages, without any notion of admission control, failure detection or session state management.
- *Single Sender Groups*. These protocols build a single dissemination tree per sender where each node is a potential receiver who cooperates in localized retransmission schemes. Groups with multiple senders, which are very common in distributed systems, are treated as multiple groups with single sender, requiring an explosion of state managed at each participant, and making it impossible to correlate messages from different senders.
- *Infinite Retransmission Buffers*. One of the more complex problems in multi-sender protocols is the management of the messages stored for retransmission. The messages can be released once the system is certain that no retransmission requests can arrive any more. Given that many protocols have no knowledge about the receivers, this certainty can never be achieved and they resort to a method called application level framing to require the application to reconstruct message for retransmission. This was applicable to some multi-user collaboration tools, but was unusable for the majority of distributed systems.
- *Complete Lack of Security*. In most protocols that combine application level framing with localized retransmission (others than the sender can retransmit lost messages), it is impossible guarantee secure communication, as nodes that retransmit can not sign the message with senders key. The original messages are not kept in lack of a garbage collection protocol, and the retransmission node cannot sign it with its own key as they are anonymous, and as such the receiver has no mechanism for checking the signature.

These techniques are unacceptable if one wants to build robust, reliable, secure distributed systems that can be the basis for the mission critical enterprise systems.

But the picture is not entirely bleak. After presenting these arguments, we shift attention to a new class of protocols based on an idea from Clearinghouse [2], the database replication technology developed at Xerox Parc in the 1980's, and later used by Golding for the refDBMS system [5], and from NNTP, the gossip-based algorithm used to propagate Usenet "news" in the Internet. These turn out to be scalable under the same style of analysis that predicts poor scalability for their non-gossip counterparts.

3 EPIDEMIC TECHNIQUES FOR SCALABLE MULTICAST PROTOCOLS

Not all protocols suffer the behavior seen in these reliability mechanisms. In the class of reliable multicast protocols, epidemic multicast protocols scale quite well and easily rides out the same phenomena that cause problems with these other approaches to reliability and scalability.

For example, *Bimodal Multicast*, a protocol reported in [1], uses an epidemic control protocol that somewhat resembles the old NNTP protocol, but is running at much higher speeds. The multicast protocol consists of two sub-protocols. One of them is an unreliable data distribution protocol similar to IP multicast, or based on IP multicast when available. Upon arrival, messages are buffered, and they are delivered to the application layer in FIFO order. The buffered messages are garbage collected after some period of time.

The second sub-protocol, is based on epidemic techniques, and is used to repair gaps in the message delivery flow and to assist the garbage collection. It operates as follows: Each process in the system maintains a list containing some random subset of the full system membership. In practice, we weight this list to contain primarily processes from close by – processes accessible over low-latency links – but these details go beyond the scope of this paper. At some rate (but not synchronized across the system) each participant selects one of the processes in its membership list at random and sends it a *digest* of its current message buffer contents. This digest would normally just list messages available in the buffer: "messages 5-11 and 13 from sender *s,* …" for example. Upon receipt of a *gossip* message, a process compares the list of messages in the digest with its own message buffer contents. Depending upon the configuration of the protocol, a process may *pull* missing messages from the sender of the gossip by sending a retransmission *solicitation*, or may *push* messages to the sender by sending unsolicited retransmissions of messages apparently missing from that process.

This simplified description omits a number of important optimizations to the protocol. In practice, we use gossip not just for multicast reliability, but also to track system membership and perform failure detection based on it [6]. We sometimes use unreliable multicast with a regional TTL value instead of unicast, in situations where it is likely that multiple processes are missing copies of the message. A weighting scheme is employed to balance loads on links: gossip is done primarily to nearby processes over low-latency links and rarely to remote processes, over costly links that may share individual routers [11]. The protocol switches between gossip pull and gossip push, using the former for "young" messages and the latter for "old" ones. Finally, we don't actually buffer every message at every process; a hashing scheme is used to spread the buffering load around the system, with the effect that the average message is buffered at enough processes to guarantee reliability, but the average buffering load on a participant decreases with increasing system size.

An epidemic-style protocol has a number of important properties: the protocol imposes constant loads on participants, is extremely simple to implement and rather inexpensive to run. More important from the perspective of this paper, however, such a protocol overcomes the problems cited earlier for other scalable protocols. Bimodal Multicast has tunable reliability that can be matched to the needs of the application (reliability is increased by increasing the length of time before a message is garbage collected, but this also causes buffering and I/O costs to rise). The protocol gives very steady data delivery rates with predictable, low, variability in throughput. For real-time applications, this can be extremely useful. And the protocol imposes constant loads on links and routers (if configured correctly), which avoids network overload as a system scales up. All of these characteristics are preserved as the size of the system increases.

4 OVERCOMING LIMITATION TO SCALE

We can generalize from the phenomena enumerated above. Distilling these down to their simplest form, and elaborating slightly:

- With the exception of the epidemic protocols, the different reliability model of group communication systems involves a costly, but infrequent fault-recovery mechanism:
  - Virtual synchrony based protocols employ flow control, failure detection and membership-change protocols; when incorrectly triggered, the cost is proportional to the size of the group.
  - Local repair based protocols have a solicitation and retransmission mechanism that involves multicasts; when a duplicate solicitation or retransmission occurs, all participants process and transmit extra messages.
  - FEC-based reliability mechanisms try to reduce retransmission requests to the sender by encoding redundancy in the data stream. As the group size grows, however, either the average multicast path length increases, hence so too the risk of a multi-packet loss. The sender will see increasingly many retransmission requests (consuming a scarce resource), or the redundancy of the stream itself must be increased (resulting in a bandwidth degradation and a system-wide impact).
- Again with the exception of the epidemic protocols, the mechanisms we've reviewed are potentially at risk from convoy-like behaviors. Even if data is injected into a network at a constant rate, as it spreads through the network, router scheduling delays and link congestion can make the communication load bursty. Under extreme condition this behavior can even trigger message loss at the end nodes. To smoothen burstiness of messages from multiple senders one needs a global view of the system, which most reliable protocols have found impossible to implement. Epidemic techniques however are ideal to implement this global state sharing and allow the overall system to gracefully adapt to changes in the network.
- Many protocols depend upon configuration mechanisms that are sensitive to network routing and topology. Over time, network routing can change in ways that take the protocol increasingly far from optimal, in which case the probabilistic mechanisms used to recover from failures can seem increasingly expensive. Periodic reconfigurations, the obvious remedy, introduce a disruptive system-wide cost.

In contrast, the epidemic mechanisms used in NNTP, the Xerox Clearinghouse system, the Astrolabe state management systems [7,8], and Bimodal Multicast protocol appear to scale without these kinds of problems. Throughput is stable if measured over sufficiently long periods of time – gossip protocols can be rather *unstable* if metered on a short time scales. Overheads are flat and predictable, and can be balanced with information about network topology, so that links and routers won't become overloaded. And, the levels of reliability achieved are very high – indeed, potentially as high as those of the protocols purporting to offer stronger guarantees.

Probabilistic guarantees may sound like a contradiction in terms, because one's intuition suggests that anything but an absolute reliability guarantee would be the equivalent of no reliability at all. Our work suggests that this is not at all the case. First, it is possible to design mechanisms that have stronger guarantees, such as virtual synchrony, and yet reside in an end-to-end manner over the basic network architecture afforded by our gossip infrastructure.

An important observation, contributing to the overall success of this approach, is also that the epidemic tools exploit scalability, and as such turn scale into an advantage instead of a problem that must be overcome.

5 OTHER APPLICATIONS OF EPIDEMIC TECHNIQUES

We have applied the epidemic techniques to develop a large set of protocols that each has excellent scalability properties for their target domains. This reaches from multicast and failure detection, through state-management and data-fusion to ad-hoc routing protocols and protocols for controlling power grids. An extensive list of relevant publications can be found at:

http://www.cs.cornell.edu/Info/Projects/Spinglass.

In our current activities we are expanding the application of epidemic techniques to other areas where scalability plays an important role. Two of these areas are power and coverage aware communication for sensor networks [9] and world-wide peer-to-peer publish-subscribe networks [10].

This research is supported by DARPA/ONR under contract N0014-96-1-10014, by the National Science Foundation under Grant No. EIA 97-03470 and by grants from Microsoft Research.

REFERENCES

[1] Birman, Ken, Hayden, Mark, Ozkasap, Oznur, Xiao, Zhen, Budiu, Mihai and Minsky, Yaron "Bimodal Multicast" ACM Transactions on Computer Systems, Vol. 17, No. 2, pp 41-88, May, 1999.

[2] Demers, A. et. al. Epidemic Algorithms for Replicated Data Management. Proceedings of the 6th Symposium on Principles of Distributed Computing (PODC), Vancouver, Aug. 1987, 1-12.

[3] Gupta, Indranil, Birman, Ken, and van Renesse, Robbert, "Fighting Fire with Fire: Using Randomized Gossip to Combat Stochastic Scalability Limits", Special Issue of Quality and Reliability of Computer Network Systems, Journal of Quality and Reliability Engineering International, May/June 2002, Vol. 18, No. 3, pp 165-184

[4] Gray, Jim, Helland, Pat, O'Neil, P., and Shasha, D., The dangers of replication and a solution. Proceedings 1996 SIGMOD Conference, June 1996.

[5] Golding, Richard and Taylor, Kim,. Group Membership in the Epidemic Style. Technical report UCSC-CRL-92-13, University of California at Santa Cruz, May 1992.

[6] van Renesse, Robbert, Minsky, Yaron, and Hayden, Mark, “A Gossip-Based Failure Detection Service”, in the Proceedings of Middleware '98. England, August 1998.

[7] van Renesse, Robbert, and Birman, Kenneth P, “Astrolabe: A Robust and Scalable Technology for Distributed System Monitoring, Management, and Data Mining”, Submitted to ACM TOCS, November 2001

[8] van Renesse, Robbert, and Birman, Kenneth P, Dumitriu, Dan and Vogels, Werner, “Scalable Management and Data Mining Using Astrolabe” in the. Proceedings of the First International Workshop on Peer-to-Peer Systems (IPTPS),. Cambridge, Massachusetts. March 2002.

[9] Robbert van Renesse, Power-Aware Epidemics, In the Proceedings of the International Workshop on Reliable Peer-to-Peer Systems, Osaka, Japan. October 2002.

[10] Werner Vogels, Chris Re, Robbert van Renesse and Ken Birman, .”A Collaborative Infrastructure for Scalable and Robust News Delivery”. In the Proceedings of the IEEE Workshop on Resource Sharing in Massively Distributed Systems (RESH'02), Vienna, Austria, July 2002.

[11] Xiao, Zhen and Birman, Ken. A Randomized Error Recovery Algorithm for Reliable Multicast. In the Proceedings of FTCS 2001. July 2001.

Data-Centric Storage in Sensornets

Scott Shenker
ICSI, Berkeley, CA 94704
shenker@icsi.berkeley.edu

Sylvia Ratnasamy*
ICSI/UCB, Berkeley, CA 94704
sylviar@cs.berkeley.edu

Brad Karp†
ICSI, Berkeley, CA 94704
bkarp@icsi.berkeley.edu

Ramesh Govindan
USC Comp. Sci., LA, CA 90089
ramesh@usc.edu

Deborah Estrin
UCLA Comp. Sci., LA, CA 90095
destrin@cs.ucla.edu

1. INTRODUCTION

Recent engineering advances in the design of small energy-efficient hardware and compact operating systems have enabled the development of large-scale distributed sensing networks (sensornets) made up of many small sensing devices equipped with memory, processors, and short-range wireless communication. These sensornets will provide an unprecedented amount of detailed measurements over wide geographic areas. However, these data are distributed across the entire sensornet, and so are hard to use. Communication between sensornet nodes requires the expenditure of energy, a scarce commodity in most sensornets. Thus, making effective use of sensornet data will require scalable, self-organizing, and energy-efficient data dissemination algorithms.

Since the content of the data is more important than the identity of the node that gathers them, sensornet researchers have found it useful to move away from the Internet's point-to-point communication abstraction and instead adopt abstractions that are more data-centric. This approach entails *naming* the data and using communication abstractions that refer to those names rather than to node network addresses [1, 9]. In particular, previous work on data-centric routing has shown it to be an energy-efficient data dissemination method for sensornets [10]. Herein, we propose a useful companion method, data-centric storage (DCS). In DCS, relevant data are stored by name at nodes within the sensornet; all data with the same general name (*e.g.,* elephant sightings) will be stored at the same sensornet node (not necessarily the one that originally gathered the data). Queries for data with a particular name can then be sent directly to the node storing those named data, without the flooding required in some data-centric routing proposals.

Several data-centric dissemination methods are conceivable, each with rather different performance characteristics. The appropriate data dissemination method for a particular task will depend on the nature of the sensornet, its intended deployment environment, and the expected workload. We make three principal contributions in this paper:

- We propose a novel data dissemination method, DCS, and show where it outperforms other approaches.
- We provide an organizing framework for comparing among three canonical data dissemination approaches, and predict where each performs best.
- We give an overview of GHT, a Geographic Hash Table system that implements DCS.

Our claim is not that data-centric storage is always the method of choice, but rather that under some conditions it will be the most desired option. In fact, we expect that future sensornets will embody all of these (or similar) data-centric dissemination methods, and that users will choose the appropriate method based on the task at hand. To understand the relative behavior of each dissemination method under different conditions, one must in turn understand the context in which these algorithms will be deployed.

For this reason, we begin our paper with an overview of related work (in Section 2) that gives a sense of the role played by data dissemination in a complete sensornet system. Thereafter, Section 3 provides a general discussion of sensornet dissemination algorithms—their constituent components and the environments in which they may be used. In Section 4 we describe three canonical dissemination methods and use a simple analytical model to compare their costs.

There may be many viable system designs for data-centric storage. We present a scalable DCS system design that builds on two recent advances: (1) the GPSR geographic routing algorithm [11] and (2) a new generation of efficient peer-to-peer lookup systems such as Pastry, CAN, Chord, and Tapestry [6, 18, 21, 24]. In Section 5, we describe the high-level design for our DCS system, which we call Geographic Hash Table (GHT). The design details and evaluation of GHT are described in a companion paper [19].

*Sylvia Ratnasamy's current affiliation: Intel Research, Berkeley, CA 94704.

†Brad Karp's current affiliation: Intel Research and Carnegie Mellon University, Pittsburgh, PA 15213, bkarp@cs.cmu.edu.

2. RELATED WORK

In this section, we briefly review related work on sensornets. We organize this discussion in "layers" ordered from bottom to top. These layers are used only to clarify the presentation and convey a sense of the role of data dissemination in a complete sensornet system; we don't mean to imply that sensornet architecture is organized into clean, well-separated layers. We begin our review at layer three (packet routing), as we are concerned with data dissemination, which interacts directly with layer three and above. Layers one (physical and OS) and two (low-level communication and self-configuration) are comparatively orthogonal to data dissemination.

L3: Packet routing:

Packet routing algorithms are required to deliver packets between nodes that are not in mutual radio range. Packet routing systems based on node identifiers are ill-suited to sensornets, where communication isn't addressed to node identifiers. It is expected that sensornets will instead implement geographic routing systems that deliver packets to nodes based on their location. Below we describe several types of geographic routing systems, each with its own communication abstraction and energy cost. In the following, we let n be the number of nodes in the sensornet, and assume that the diameter of the sensornet is approximately $O(\sqrt{n})$.

Strongly geographic routing algorithms, like GPSR [11], allow nodes to send to a particular location. To go from one random location to another requires $O(\sqrt{n})$ packet transmissions, which is our (approximate) metric for total energy consumption. *Weakly geographic* routing algorithms like GEAR [23] allow a node to send packets to a region and then distribute the packet within that region. The transmission costs here are $O(\sqrt{n})$ packet transmission to reach the region and $O(r)$ packet transmissions within the region, where r is the number of nodes in the region.

In addition to geographic routing, two other packet routing primitives are likely to be available in sensor networks. *Scoped flooding* algorithms flood to a limited region around the sending node. Their transmission cost is $O(r)$ where r is the number of nodes in the region. *Flooding* sends a packet to the entire sensornet, and requires $O(n)$ packet transmissions.

L4: Local collaborative information processing:

Event detection sometimes requires synthesizing results from several different sensors. The algorithms in this class only require collaboration between local nodes; *i.e.,* those that can be reached by a tightly-scoped flood. An example of such algorithms is described in [22].

L5: Wide-Area data dissemination:

Under the data-centric architecture for sensornets, data are named. The data dissemination methods we refer to here allow nodes and users to access data by name across the sensornet. Note that, in contrast, the local collaborative information processing only used data that could be found nearby; these wide-area data dissemination methods are needed to do collaborative processing in the wide area, as we describe below.

The most commonly mentioned wide-area data dissemination technique is *directed diffusion* [9, 10], an example of *data-centric routing*: routing decisions are based on the name of the data rather than on the identity of the sending and receiving nodes. We discuss directed diffusion at greater length in Section 4.3. In this paper we propose another data dissemination approach: *data-centric storage*, whereby event information is stored by name within the sensornet.

It should be noted that directed diffusion (and most other data-centric routing proposals) do not require any packet forwarding methods other than flooding. In contrast, the data-centric storage proposal we present here requires strongly geographic routing. Thus, the data dissemination method choice may be limited by the nature of the underlying packet routing mechanisms.

L6: Wide-area collaborative information processing:

These methods are akin to the local collaborative information processing methods mentioned above, except that the collaborating nodes need not be local. An example of such a collaboration is that required for tracking an object across a sensor field. In this case, scalable collaborative methods must be built on efficient wide-area data-dissemination algorithms. Zhao *et al.* [25] describe a collaborative tracking application built on top of directed diffusion.

L7: User-level tasking and querying:

The highest layer is where users insert their tasking and querying commands. An example of an approach that fits here is work that has been done on defining database semantics for queries on sensor networks [2, 8, 14].

3. ASSUMPTIONS AND TERMINOLOGY

This section lays out the context for the dissemination algorithms discussed in the following section. We first state our basic assumptions about the class of sensornets we consider and then describe some basic concepts regarding sensornet data and how they are organized.

3.1 Assumptions

Projected sensornet designs in the literature [5] differ greatly in their characteristics and intended uses. Here we focus attention on that class of sensornets where wide-area data dissemination is a needed service.

We consider large-scale sensornets with nodes that are spread out over some well-defined area, whose approximate geographic boundaries are known to network operators.

We assume nodes have short range communication, but are within radio range of several other nodes. We further assume that nodes know their own locations. GPS or some other approximate but less burdensome localization algorithm [3, 7, 16, 17, 20] provides this information. This assumption is crucial for our proposed data-centric storage mechanism. We believe it a reasonable assumption because in many cases the sensornet data are useful only if the locations of their sources are known.

We further assume that communication to the outside world takes place through one or more access points in the sensornet, and so getting data from a sensornet node to the outside world requires sending the data through the sensornet to the access point. We use the term *access path* to refer to the set of paths data take from sensor nodes to the access point(s). This assumption is not required by our DCS mechanism *per se*, but is part of our model for comparing costs of different dissemination mechanisms.

We assume energy is a scarce commodity for sensornet nodes [15], such that data dissemination algorithms should seek to minimize

communication in order to extend overall system lifetime. While the mapping between communication and energy consumption is complicated—it depends greatly on the precise hardware involved and the packet transmission pattern—in what follows we will focus on two simplified metrics of energy consumption:

Total usage: The total number of packets sent in the sensornet

Hotspot usage: The maximal number of packets sent by any particular sensornet node

While in the rest of the paper we treat all nodes as having the same capabilities, it is likely that real sensornets will have a tiered architecture, where some nodes have very limited data storage capacity and others have much more significant storage (and perhaps also more battery power and better communication facilities). Our discussion applies to this tiered approach as well, as long as the these "macronodes" are numerous and widely dispersed [4].

These assumptions describe the physical environment of the sensornet. We next discuss how these sensornets might be used.

3.2 Observations and Events

The purpose of sensornets is to provide detailed sensing capabilities across a wide geographic area. The low-level readings from these sensors, which we call *observations*, are named (as described, for example, in [1, 9]). While sensornets give unprecedented access to detailed observations of the physical world, sending this overwhelming volume of observations directly to the access point(s) would quickly exhaust the energy reserves of the sensornet. Fortunately, in most cases users don't want the complete set of raw unprocessed data, but rather are more interested in specific *events*, such as earthquakes or animal sightings.

We use the term *events* to refer to certain pre-defined constellations of low-level observations. For example, detailed temperature and pressure readings might constitute observations, while a particular combination of temperature and pressure might define an "elephant-sighting" event. A sensornet system will be designed to detect several well-defined *types* of events. Typically, the large volume of observations prohibits communicating them directly to the outside world. Events are thus derived by processing the low-level observations within the sensornet through *collaborative information processing* techniques.

Events can be defined not only in terms of low-level observations but also in terms of other events. For instance, detecting an animal migration would involve many individual animal sightings. In general, there will be a *web* of events, with some events defined in terms of others. These events are not necessarily in a strict hierarchy, but in the context of a particular application there is some sense that some events are lower-level than others, and could be used to define the higher-level events.

3.3 Tasks, Actions, and Queries

The preceding discussion identified the various types of information—observations and events—that might be provided by sensornets. We now describe the operations used to manipulate these data.

Users send instructions (by flooding or some other global dissemination method) to sensornet nodes to run certain local identification *tasks*. These tasks could be simple, such as taking temperature readings, or complex, such as identifying an animal from a collection of sensor readings. In essence, one can think of tasks as downloaded code.

Once an event has been identified, nodes can take one of three *actions*: a node could send event information to external storage, store the event information locally, or use data-centric storage. Recall that data-centric storage involves storing the event information at a sensornet node that is chosen based on the event's name.[1] These three possible actions—external store, local store, and data-centric store—form the core of the three canonical approaches we describe in Section 4.

Unless the information has been sent to external storage, at this stage the event information is still not in the user's hands. *Queries* are used to elicit the event information from the sensornet. How queries are executed will depend on the actions nodes take upon event detection. If event information is stored locally then queries must be flooded to all nodes (unless the user has prior knowledge about the location of the event). If event information is stored using data-centric storage, the query can be sent to the sensornet node associated with that event name.

4. DATA-DISSEMINATION METHODS

The main goal in a data-dissemination algorithm is to extract relevant data efficiently from within the sensornet. In this section, we consider three *canonical* methods that combine the pieces described in the preceding section differently, yielding three very different approaches to sensornet design. We first describe these methods and then compare their costs analytically.

All the dissemination methods begin by flooding the tasks to the entire sensornet. The tasks are the set of identification instructions, specifying which events to detect, how to detect them, and what actions to take upon detection. We assume that the tasking instructions remain in force for long periods of time, so that the initial cost of issuing tasks is dominated by the cost of the ensuing data processing.[2]

We also assume that event locations are not known in advance and are distributed randomly throughout the sensornet. The case where this assumption does not hold is discussed in the following section.

Finally, in evaluating communication costs we assume asymptotic costs of $O(n)$ message transmissions for floods and $O(\sqrt{n})$ for point-to-point routing where n is the number of nodes.

4.1 Canonical Methods

Earlier we described three basic actions nodes could take upon detecting an event. These lead directly to three canonical sensornet methods.

External Storage (ES):

Upon detection of events, the relevant data are sent to external storage where they are further processed as needed. This entails a cost

[1] This approach, like all data-centric approaches, requires a naming scheme. We do not address this issue here, but merely note that the naming scheme is part of the definition of events and is supplied by the globally disseminated tasking instructions.

[2] In situations where tasks are short-lived, the cost of flooding tasks dominates all other costs, and it doesn't matter much which of the approaches below is used.

of $O(\sqrt{n})$ for each event. There is no cost for external queries since the event information is already external; queries generated by internal nodes in the process of event detection will incur a cost of $O(\sqrt{n})$ to reach the external storage.

Local Storage (LS):
Event information is stored locally (at the detecting node) upon detection of an event; this incurs no communication costs. Queries are flooded to all nodes at a cost of $O(n)$. Responses are sent back to the source of the query at a cost of $O(\sqrt{n})$ each.

Data-Centric Storage (DS):
Here, after an event is detected the data are stored by name within the sensornet. The communication cost to store the event is $O(\sqrt{n})$. Queries are directed to the node that stores events of that name, which returns a response, both at a cost of $O(\sqrt{n})$.

These three canonical methods have very different cost structures; we compare these analytically in the next subsection.

4.2 Approximate Communication Costs

This section uses a simple analytical model to derive approximate formulae for the communication costs for the three canonical methods; these formulae suggest which method is best suited for a particular sensornet workload.[3]

The cost structure for the canonical methods is described by several parameters. We consider a sensornet with n nodes equipped to detect T event types. We let D_{total} denote the total number of events detected, Q denote the number of event types for which queries are issued, and D_q denote the number of events detected for each event queried for. We assume there is no more than one query for each event type, so there are Q queries in total.

In comparing costs, we also consider the case where users only care about a summary of the events rather than a listing of each one; *e.g.,* one might just want a count of the number of elephants seen rather than a listing of each elephant sighting.

We compare costs using approximations for both the total number of packets and the packets arriving at the access point. The packet count at the access point is a good estimate of the hotspot usage, since we expect the access point to be the most heavily used area of the sensornet.

External Storage:
Total: $D_{total}\sqrt{n}$ Hotspot: D_{total}

Local Storage:
Total: $Qn + D_q\sqrt{n}$ Hotspot: $Q + D_q$

Data-Centric Storage:
Total: $Q\sqrt{n} + D_{total}\sqrt{n} + D_q\sqrt{n}$ (list)
Total: $Q\sqrt{n} + D_{total}\sqrt{n} + Q\sqrt{n}$ (summary)
Hotspot: $Q + D_q$ (list) or $2Q$ (summary)

In the above, (list) indicates a full listing of events is returned (requiring a packet for each event) and (summary) indicates only a summary of events is returned (requiring only one packet).

[3] In a companion paper [19], we verify the validity of these approximations through simulation.

These calculations suggest a few straightforward observations. First, if all other parameters are held fixed, then as n increases the local storage method incurs the greatest total packet count. Second, external storage always incurs a lesser total message count than data-centric storage, but the ratio $1 + \frac{Q+D_q}{D_{total}}$ is unlikely to be great if there are many events detected (and, if there is at least one event detected of each type, this ratio is bounded by 3). Third, if $D_q \gg Q$ and events are summarized, then data-centric storage has the least load (of all three methods) on the access path. Fourth, if events are listed and $D_{total} \gg D_q$ then data-centric storage and local storage have significantly lesser access loads than external storage.

These observations in turn suggest that data-centric storage is preferable in cases where (a) the sensornet is large, (b) there are many detected events and not all event types are queried, so that $D_{total} \gg \max[D_q, Q]$. This performance advantage increases further when summaries are used. However, if the number of events is large compared to the system size, $D_{total} > Q\sqrt{n}$, and event lists (rather than summaries) are used, then local storage may be preferable.

4.3 Additional Dissemination Methods

The three canonical methods described in the previous section certainly do not exhaust the design space; combinations of them yield hybrid methods specialized for particular needs. Examples of such combinations include:

Using Data-Centric Storage for Location Guidance:
For certain applications, one might combine LS and DCS by storing detailed event information locally and using DCS to inform a querier of an event's location so that subsequent queries can be directed to the proper local store.

Using Data-Centric Storage for Context:
In the course of processing local data, nodes may find it useful to have some context about global parameters. For instance, data-centric storage could give nodes access to the number of other animals sighted when a node is trying to determine if a migration is underway.

Geographically Targeted Queries:
The canonical methods are designed for cases where one doesn't *a priori* know the event location. If one already knows the location of the event through out-of-band techniques, then one can direct queries to that location using geographic routing methods (see [23]). This LS variant stores data locally, and queries are sent (at cost $O(\sqrt{n})$) to the relevant locations to retrieve the desired data. It avoids the cost of flooding in the canonical LS approach, and the cost of storing each event in the canonical DCS approach.

5. THE GEOGRAPHIC HASH TABLE

We now offer a brief overview of our design for the Geographic Hash Table (GHT) system that supports the data-centric storage abstraction; the detailed design and evaluation of GHT are described in [19].

GHT provides a *(key,value)*-based associative memory. Events are named with keys. Both the storage of an event and its retrieval are performed using these keys. GHT is naming-agnostic; any naming scheme that distinguishes the requisite events suffices. The operations GHT supports are:

Put(k, v) stores v (the observed data) according to the key k, the name of the data.

Get(*k*) retrieves whatever value is stored associated with key *k*.

In terms of functionality (*i.e.,* the above interface), GHT is inspired by the new generation of Internet-scale Distributed Hash Table (DHT) systems such as Pastry, CAN, Chord, and Tapestry [6, 18, 21, 24]. In these systems, nodes are assigned virtual identifiers and a data object's key is also hashed to a virtual identifier. Given a particular key, a name-based routing algorithm is then used to locate the node with virtual identifier closest to the key's virtual identifier. This node then serves as the storage node for that key.

Although GHT provides functionality equivalent to that of the DHTs, it would be inappropriate to adopt the DHT routing algorithms for use on sensornets. These algorithms typically interconnect nodes in a way determined by their logical identifiers in the DHT system, which are largely independent of their proximity in the physical network topology. On the Internet the IP routing system offers connectivity between nodes that are not topologically close. But in the energy-constrained sensornet environment, maintaining routing among all pairs of nodes is infeasibly expensive.

Instead, the core idea in GHT is to use the true geographic (*i.e.,* latitude-longitude) space occupied by the sensornet as the logical identifier space and use geographic routing as the underlying name-based routing technique. Thus, a key is hashed to a geographic position and geographic routing is used to locate the node physically closest to this position. This approach allows us to achieve the required hash-table-like functionality while working with only the true physical connectivity between nodes.

GHT uses GPSR [11], a geographic routing system for multi-hop wireless networks. Under GPSR, a sender that wishes to transmit packets to a destination node marks these packets with the destination's *position*.[4] All nodes know their own positions and the positions of nodes a single hop away from them. Using only this local knowledge, GPSR can route a packet to any destination node.

GHT, however, uses geographic routing in a slightly different manner. Under GHT, unlike under GPSR, the originator of a packet does *not* know the node that is the eventual destination of a packet. The originator of a GHT **Put()** or **Get()** for a key *k* hashes the name *k* into geographic coordinates that act as the destination of the packet for that operation. The hash function is ignorant of the placement of individual nodes in the topology; it merely spreads the different key names evenly across the geographic region where the network is deployed. Thus, it is quite likely that there is no node at the precise coordinates the hash function produces. Fortunately, the manner in which GPSR treats such a packet is precisely the behavior desired by GHT—GPSR forwards the packet until it reaches the node geographically closest to the destination coordinates. Under GHT, this closest node consumes and processes the packet. Note that GHT does not change the underlying GPSR routing algorithm; we merely leverage a previously unexploited characteristic of GPSR that allows all packets destined for an arbitrary location to be routed consistently to the node closest to it.

The above approach is sufficient to support our GHT interface provided sensornet nodes are completely stationary and reliable. However, as with DHTs, much of the subtlety in the GHT system design arises specifically to ensure robustness and scalability in the face of the many sorts of failures possible in a distributed system. GHT uses a novel *perimeter refresh protocol* to provide both persistence and consistency when nodes fail or move. This protocol replicates stored data for key *k* at nodes around the location to which *k* hashes, and ensures that the appropriate storage node for *k* is chosen consistently.

By hashing keys, GHT spreads storage and communication load between different keys evenly throughout the sensornet. When many events with the same key are stored, GHT avoids concentrating communication and storage at their shared home node by employing *structured replication*, whereby events with the same key can be divided among multiple mirrors.

GHT fundamentally requires that a node know its own geographic position. While this assumption seems reasonable for most sensornets, an open question is how (if at all) one might achieve DCS using only approximate geographic information, or better still, without requiring any position information at all. This question is the subject of our continuing research.

[4] A sender maps the destination's identifier to its current position using a location database, such as GLS [13].

6. CONCLUSION

We believe future sensornets will incorporate many different data dissemination mechanisms and users will choose among them based on the particular circumstances. In this paper we proposed data-centric storage as a novel approach to wide-area data dissemination. We identified the settings where data-centric storage may be the preferred method because of its energy efficiency. We provided a framework for reasoning about data dissemination that divides the design space into three canonical approaches, and gave a simple model for characterizing their respective energy costs.

We briefly described the design of GHT and our approach to achieving data-centric storage, but there are doubtless other approaches. In particular, the GRID location system (GLS) [13] can be extended to provide a similar capability. GLS constructs and maintains a distributed database that maps node addresses to geographic positions. While the goal in GLS is to provide a node location service, for our sensornet application, this database is merely an unnecessary level of indirection; we map event names directly to locations. The SCOUT [12] location tracking system might also be used in a similar manner. While SCOUT uses hierarchical addressing and routing based on landmark routing, GHT uses GPSR, a flat routing algorithm wherein nodes are addressed with geographic coordinates.

A Linux version of the GHT design for DCS is under development for iPAQs communicating with both 802.11 radios and mote radios. In our initial applications of the system, we will experiment with event naming schemes toward the goal of realizing the benefits of DCS fully.

7. REFERENCES

[1] W. Adjie-Winoto, E. Schwartz, and H. Balakrishnan, The Design and Implementation of an Intentional Naming System, In *Proceedings of the Symposium on Operating Systems Principles*, pp. 186–201, (Charleston, South Carolina, December 1999).

[2] P. Bonnet, J. Gehrke, and P. Seshadri, Querying the Physical World, IEEE Personal Communications Magazine, Special Issue on Networking the Physical World, (October 2000).

[3] N. Bulusu, J. Heidemann, and D. Estrin, GPS-Less Low Cost Outdoor Localization for Very Small Devices, IEEE Personal

Communications Magazine, Special Issue on Smart Spaces and Environments, (October 2000).

[4] A. Cerpa, J. Elson, D. Estrin, L. Girod, M. Hamilton, and J. Zhao, Habitat monitoring: Application Driver for Wireless Communications Technology, In ACM SIGCOMM Workshop on Data Communications in Latin America and the Caribbean, (Costa Rica, April 2001).

[5] Defense Advanced Research Projects Agency, Sensor Information Technology, `http://www.darpa.mil/ito/research/sensit`.

[6] A. Rowstron and P. Druschel, Pastry: Scalable, Distributed Object Location and Routing for Large-scale Peer-to-peer Systems, In *Proceedings of the 18th IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, (Heidelberg, Germany, November 2001).

[7] L. Girod and D. Estrin, Robust Range Estimation Using Acoustic and Multimodal Sensing, In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems*, (Maui, Hawaii, October 2001).

[8] R. Govindan, J. Hellerstein, W. Hong, S. Madden, M. Franklin and S. Shenker, The Sensor Network as a Database, preprint, (2002).

[9] J. Heidemann, F. Silva, C. Intanagonwiwat, R. Govindan, D. Estrin, and D. Ganesan, Building Efficient Wireless Sensor Networks with Low-level Naming, In *Proceedings of the Symposium on Operating Systems Principles*, pp. 146–159, (Alberta, Canada, October 2001).

[10] C. Intanagonwiwat, R. Govindan, and D. Estrin, Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks, In *Proceedings of the Sixth Annual ACM/IEEE International Conference on Mobile Computing and Networking (Mobicom)*, (Boston, Massachusetts, August 2000).

[11] B. Karp and H.T. Kung, Greedy Perimeter Stateless Routing, In *Proceedings of the Sixth Annual ACM/IEEE International Conference on Mobile Computing and Networking (Mobicom)*, (Boston, Massachusetts, August 2000).

[12] S. Kumar, C. Alaettinoglu, and D. Estrin, SCalable Object-tracking through Unattended Techniques (SCOUT), In *Proceedings of the 8th International Conference on Network Protocols(ICNP)*, (Osaka, Japan, November 2000).

[13] J. Li, J. Jannotti, D. DeCouto, D. Karger, and R. Morris, A Scalable Location Service for Geographic Ad-hoc Routing, In *Proceedings of the Sixth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom 2000)*, (Boston, Massachusetts, August 2000).

[14] S. Madden, M. Shah, J. Hellerstein, and V. Raman, Continuously Adaptive Continuous Queries over Streams, In *Proceedings of the ACM SIGMOD Conference*, (Madison, WI, June 2002).

[15] G. Pottie and W. Kaiser, Wireless Integrated Sensor Networks (WINS): Principles and Approach, *Communications of the ACM*, Vol. 43, Number 5, pp. 51–58, (May 2000).

[16] N. Priyantha, A. Chakraborty, and H. Balakrishnan, The Cricket Location Support System, In *Proceedings of the Sixth Annual ACM/IEEE International Conference on Mobile Computing and Networking (Mobicom)*, (Boston, Massachusetts, August 2000).

[17] N. Priyantha, A. Liu, H. Balakrishnan, and S. Teller, The Cricket Compass for Context-Aware Mobile Applications, In *Proceedings of the Seventh Annual ACM/IEEE International Conference on Mobile Computing and Networking (Mobicom)*, (Rome, Italy, July 2001).

[18] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, A Scalable Content-Addressable Network, In *Proceedings of the ACM SIGCOMM*, (San Diego, California, August 2001).

[19] S. Ratnasamy, B. Karp, L. Yin, F. Yu, D. Estrin, R. Govindan, and S. Shenker, GHT: A Geographic Hash Table for Data-Centric Storage in SensorNets, In *Proceedings of the First ACM International Workshop on Wireless Sensor Networks and Applications (WSNA)*, (Atlanta, Georgia, September 2002).

[20] A. Savvides, C.-C. Han, and M. B. Srivastava, Dynamic Fine-Grain Localization in Ad-Hoc Networks of Sensors, In *Proceedings of the Seventh Annual ACM/IEEE International Conference on Mobile Computing and Networking (Mobicom)*, (Rome, Italy, July 2001).

[21] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications, In *Proceedings of the ACM SIGCOMM*, (San Diego, California, August 2001).

[22] K. Yao, R. Hudson, C. Reed, D. Chen, and F. Lorenzelli, Blind Beamforming on a Randomly Distributed Sensor Array, In *IEEE Journal on Selected Areas in Communication*, Vol. 16, Number 8, (October 1998).

[23] Y. Yu, D. Estrin, and R. Govindan, Geographical and Energy-Aware Routing: A Recursive Data Dissemination Protocol for Wireless Sensor Networks, UCLA Computer Science Department Technical Report, UCLA-CSD TR-01-0023, (May 2001).

[24] B. Y. Zhao, J. Kubiatowicz, and A. Joseph, Tapestry: An Infrastructure for Fault-tolerant Wide-area Location and Routing, Tech. Rep. UCB/CSD-01-1141, University of California at Berkeley, Computer Science Department, (April 2001).

[25] F. Zhao, J. Shin, and J. Reich, Information-Driven Dynamic Sensor Collaboration for Tracking Applications, In *IEEE Signal Processing Magazine*, (March 2002).

DIMENSIONS: Why do we need a new Data Handling architecture for Sensor Networks?

Deepak Ganesan
Dept of Computer Science
UCLA
Los Angeles, CA 90095
deepak@cs.ucla.edu

Deborah Estrin
Dept of Computer Science
UCLA
Los Angeles, CA 90095
destrin@cs.ucla.edu

John Heidemann
USC/ISI
4676, Admiralty Way
Marina Del Rey, CA 90292
johnh@isi.edu

ABSTRACT

An important class of networked systems is emerging that involve very large numbers of small, low-power, wireless devices. These systems offer the ability to sense the environment densely, offering unprecedented opportunities for many scientific disciplines to observe the physical world. In this paper, we argue that a data handling architecture for these devices should incorporate their extreme resource constraints - energy, storage and processing - and spatio-temporal interpretation of the physical world in the design, cost model, and metrics of evaluation. We describe DIMENSIONS, a system that provides a unified view of data handling in sensor networks, incorporating long-term storage, multi-resolution data access and spatio-temporal pattern mining.

1. INTRODUCTION

Wireless Sensor Networks offer new opportunities for pervasive monitoring of the environment and ultimately for studying previously unobservable phenomena. Using traditional techniques, the data handling requirements of these systems will overwhelm the stringent resource constraints on sensor nodes [1]. In this paper, we describe DIMENSIONS, a system to enable scientists to observe, analyze and query distributed sensor data at multiple resolutions, while exploiting spatio-temporal correlation.

Application	*Observed phenomena*
Building Health Monitoring [2]	response to earthquakes, strong winds
Contaminant Flow	Concentration pattern, pooling of contaminants, plume tracking
Habitat micro-climate monitoring [3]	spatial and temporal variations

Table 1: Example Scientific Applications

Sensor networks place several requirements on a distributed storage infrastructure. These systems are highly data-driven (Table 1): they are deployed to observe, analyze and understand the physical world. A data handling architecture must, therefore, reconcile conflicting requirements:

- A fully centralized data collection strategy is infeasible given the energy constraints on sensor node communication, and inefficient given that sensor data has significant redundancy.
- Many queries over these systems will be spatio-temporal in nature. The storage system should support efficient spatio-temporal querying and mining for events of interest. Such events exist at specific spatio-temporal scales, and therefore in order to extract information from data, one has to perform interpretation over a certain region. Local processing alone is not sufficient. For example, identifying pooling of contaminants in a region.
- Users will routinely require compressed summaries of large spatio-temporal sensor data. However, periodically or occasionally, users will require detailed datasets from a subset of sensors in the network.

In addressing the storage challenges of sensor networks, one question immediately comes to mind: *can we use existing distributed storage systems for our purpose?*. We argue that there are fundamental differences in the cost models, nature of the data, and intended forms of use of sensor networks, that motivate new approaches and metrics of evaluation.

- Hierarchical web caches [4] are designed to lower latency, network traffic and load. The cost models that drive their caching strategy is based on user web access patterns, strategically placing web pages that are frequently accessed. Peer-to-peer systems are designed for efficient lookup of files in a massively distributed database. These systems do not capture key challenges of sensor networks: (a) they are designed for a much less resource constrained infrastructure, unlike in sensor networks, where communication of every bit should be accounted for (b) they are optimized for bandwidth which, while limited, is a non-depletable resource, unlike energy on sensor nodes (c) the atomic unit of storage is a file, and unlike sensor data, different files are not expected to exhibit significant spatio-temporal correlations.
- Geographic Information Systems (GIS) deal with data that exhibit spatial correlations, but the processing is centralized, and algorithms are driven by the need to reduce search cost, typically by optimizing disk access latency.
- Streaming media in the internet uses a centralized approaches to compression of spatio-temporal streams such as MPEG-2, and are optimized for different cost functions. Consider the problem of compressing a 3 dimensional datacube (dimensions: x,y,$time$) corresponding to data from a single sensor type on a grid of nodes on a plane, much like a movie of

sensor data. MPEG-2 compresses first along the spatial axes (x,y), and uses motion vectors to compress along the temporal axis. The cost model driving such an approach is perceptual distortion and transmission latency. Communication constraints in sensor networks drive a time first, space next approach to compressing the datacube, since temporal compression is local and far cheaper than spatial compression.

- Wavelets [5, 6] are a popular signal processing technique for lossy compression. In a centralized setting, compression using wavelets can use entropy-based metrics to tradeoff compression benefit with reconstruction error. In a sensor network setting, pieces of the data are distributed among nodes in the network, and communication constraints force local cost metrics that tradeoff the **communication overhead** with the compression benefit.

Thus, large scale, untethered devices sensing the physical world call for building systems that incorporate their extreme resource constraints and spatio-temporal interpretation of the physical world in the design, cost model, and metrics of evaluation of a data handling architecture. DIMENSIONS constrains traditional distributed systems design with the need to make every bit of communication count, incorporates spatio-temporal data reduction to distributed storage architectures, introduces local cost functions to data compression techniques, and adds distributed decision making and communication cost to data mining paradigms. It provides unified view of data handling in sensor networks incorporating long-term storage, multi-resolution data access and spatio-temporal pattern mining.

2. DESIGN GOALS

The following design goals allow DIMENSIONS to minimize the bits communicated:

Multi-Resolution Data Storage: A fundamental design goal of DIMENSIONS is the ability to extract sensor data in a multi-resolution manner from a sensor network. Such a framework offers multiple benefits (a) it allows users to look at low-resolution data from a larger region cheaply, before deciding to obtain more detailed and potentially more expensive datasets (b) Compressed low-resolution sensor data from large number of nodes can often be sufficient for spatio-temporal querying to obtain statistical estimates of a large body of data [7].

Distributed: Design goals of distributed storage systems such as [8, 9] of designing scalable, load-balanced, and robust systems, are especially important for resource constrained distributed sensor networks. We have as a goal that the system balances communication and computation load of querying and multi-resolution data extraction from the network. In addition, it should leverage distributed storage resources to provide a long-term data storage capability. Robustness is critical given individual vulnerability of sensor nodes. Our system shares design goals of sensor network protocols that compensate for vulnerability by exploiting redundancy in communication and sensing.

Adapting to Correlations in sensor data: Correlations in sensor data can be expected along multiple axes: temporal, spatial and between multiple sensor modalities. These correlations can be exploited to reduce dimensionality. While temporal correlation can be exploited locally, the routing structure needs to be tailored to spatial correlation between sensor nodes for maximum data reduction. The correlation structure in data will vary over time, depending on the changing characteristics of the sensed field. For example, the correlation in acoustic signals depend on the source location and orientation, which can be time-varying for a mobile source. The storage structure should be able to adapt to the correlation in sensor data.

3. APPROACH

The key components of our design are (a) temporal filtering (b) *wavRoute*, our routing protocol for spatial wavelet subband decomposition (c) distributed long-term storage through adaptive wavelet thresholding. We outline key features of the system, and discuss cost metrics that tradeoff communication, computation, storage complexity, and system performance. A few usage models of the storage system are described, including multi-resolution data extraction, spatio-temporal data mining, and feature routing. To facilitate the description, we use a simplified grid topology model, whose parameters are defined in Table 2. We expect to relax this assumption in future work (Section 8).

The approach to DIMENSIONS is based on wavelet subband coding, a popular signal processing technique for multiresolution analysis and compression [5, 6]. Wavelets offer numerous advantages over other signal processing techniques for viewing a spatio-temporal dataset (a) ability to view the data at multiple spatial and temporal scales (b) ability to extract important features in the data such as abrupt changes at various scales thereby obtaining good compression (c) easy distributed implementation and (d) low computation and memory overhead. The crucial observation behind wavelet thresholding when applied to compression is that for typical time-series signals, a few coefficients suffice for reasonably accurate signal reconstruction.

n	Number of Nodes in the network
R	Region participating in the wavelet decomposition. Simplified model assumes grid placement of $\sqrt{n}$x$\sqrt{n}$ nodes
λ	Number of levels in the spatial wavelet decomposition. $\sqrt{n} = 2^{\lambda}$ and $\lambda = \frac{1}{2}logn$
(X_{apex}, Y_{apex})	Location of the apex of the decomposition pyramid
D_0	Time-series data at each node, before the spatial decomposition
Huffman	Entropy encoding scheme used

Table 2: Parameters

3.1 Temporal decomposition

Temporal data reduction is cheap since it involves only computation at a single sensor, and incurs no communication overhead. The first step towards constructing a multi-resolution hierarchy, therefore, consists of each node reducing the time-series data as much as possible by exploiting temporal redundancy in the signal and apriori knowledge about signal characteristics. By reducing the temporal datastream to include only potentially interesting events, the communication overhead of spatial decomposition is reduced.

Consider the example of a multi-resolution hierarchy for building health monitoring (Table 1). Such a hierarchy is constructed to enable querying and data extraction of time-series signals corresponding to interesting vibration events. The local signal processing involves two steps: (a) Each node performs simple real-time filtering to extract time-series that may represent interesting events. The filtering could be a simple amplitude thresholding i.e. events that cross a pre-determined SNR threshold. The thresholding yields short time-series sequences of building vibrations. (b) These time-series snippets are compressed using wavelet subband coding to yield a sequence that capture as much energy of the signal as possible, given communication, computation or error constraints.

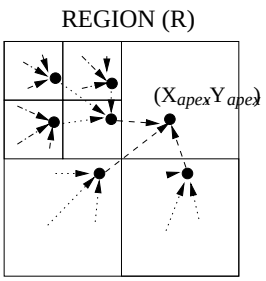


Fig 1(a) $wavRoute$ hierarchy

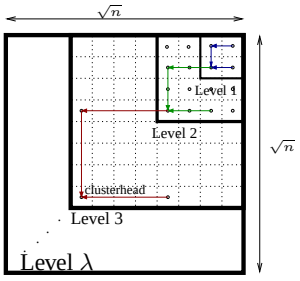


Fig 1(b) Analytical Approximation

Figure 1: Grid Topology Model

Algorithm 1: ClusterWaveletTransform

Data : R: co-ordinates of region; λ: number of decomposition levels; t - current time
Result : Spatial wavelet decomposition of λ levels
/* Location of apex of the pyramid */;
(X_{apex}, Y_{apex}) = HashInRegion(R,t);
l = 1 /* initialize level */;
while $l \leq \lambda$ **do**
  (X_l, Y_l) = LocateClusterHead (l, (X_{apex}, Y_{apex}));
  if *I am clusterhead for level* $l-1$ **then**
    GeoRoute Coefficients to (X_l, Y_l);
    current section becomes this one;
  if *I am clusterhead for level* l **then**
    Get coefficients from clusterheads at level $l-1$;
    Perform a 2 dimensional Wavelet transform;
    Store Coefficients and Deltas Locally. Save Coefficients for next iteration;

3.2 wavRoute: A Routing Protocol for Spatial Wavelet Decomposition

Spatial data reduction involves applying a multi-level 2D wavelet transform on the coefficients obtained from 1D temporal data reduction described in Section 3.1. Our goals in designing this routing protocol are twofold: (a) minimize the communication overhead of performing a spatial wavelet decomposition (b) balance the communication, computation and storage load among nodes in the network. *wavRoute* uses a recursive grid decomposition of a physical space into tiles (such as the one proposed in [10]), in conjunction with geographic routing [11, 12] as shown in Algorithm 1. At each level of the decomposition, data from four tiles at the previous level are merged at a chosen node, which stores it in its local store. The merged streams are further subband coded, thresholded, and quantized to fit within specified optimization criteria (discussed in Section 3.4). The compressed stream is sent to the next higher level of the hierarchy as shown in Figure 2. The algorithm is executed recursively (Figure 1(a)): all nodes in the network participate in Step 1, in following steps, only clusterheads from the previous step participate. In the following sections, we elaborate on the algorithm used to the select clusterhead location and the geographic forwarding protocol used.

Algorithm 2: LocateClusterHead

Data : l: current level of decomposition; (X_{apex}, Y_{apex}): location of apex of Pyramid
Result : (X_{ch}, Y_{ch}): Co-ordinates of clusterhead location for level i tile
$X_{0\ tile} = \lfloor \frac{X_{my}}{2^l} \rfloor$; $X_{1\ tile} = X_{0\ tile} + 2^l$;
$Y_{0\ tile} = \lfloor \frac{Y_{my}}{2^l} \rfloor$; $Y_{1\ tile} = Y_{0\ tile} + 2^l$;
Compute Centroid of Tile;
Compute Euclidean Shortest Path Vector L from Centroid to Storage Node;
(X_{sn}, Y_{sn}) = Intersection Between Path Vector L and Tile boundary;

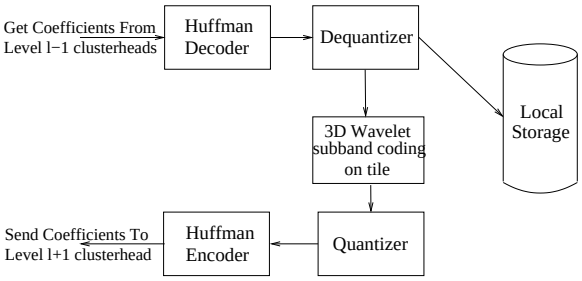


Figure 2: Protocol at clusterhead at depth i

Clusterhead Selection: To minimize communication cost, the choice of clusterhead should be tied into the routing structure. We use a simple clusterhead selection procedure that gives us good complexity properties (described in Section 5). First, the apex of the decomposition pyramid is first chosen by hashing into the geographic region, R (Figure 1(a)). Then, for each tile, the euclidean shortest path between the centroid of the tile and the apex location $((X_{apex}, Y_{apex}))$ is computed. The point where this path intersects the tile boundary is chosen to be the location of the clusterhead. Since each node can independently calculate the location of the tile based on its own geographic co-ordinates, the clusterhead location can be independently computed by each node.

Modified GPSR: We use a modified GPSR approach proposed in [11] to route packets to clusterheads. A brief review of the approach is described below, details can be obtained from [12] and [11]. GPSR is a *strongly geographic* routing protocol that takes a location rather than an address to deliver packets. [11] propose a modified GPSR protocol that ensures that packets are delivered to the node *closest* to the destination location.

3.3 Long-term Storage

Long term storage is provided in DIMENSIONS by exploiting the fact that thresholded wavelet coefficients lend themselves to good compression benefit [5, 6]. Our rationale in balancing the need to retain detailed datasets for multi-resolution data collection and to provide long-term storage is that if scientists were interested in detailed datasets, they would extract it within a reasonable interval (weeks). Long-term storage is primarily to enable spatio-temporal pattern mining, for which it is sufficient to store key features of data. Thus, the wavelet compression threshold is aged progressively as shown in Figure 3, lending older data to progressively better compression, but retaining key features of data.

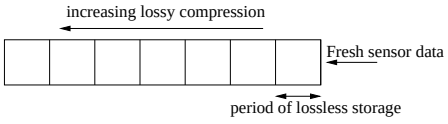


Figure 3: Long term storage

3.4 Choosing an Appropriate Cost Metric

An appropriate cost metric must weight multiple parameters: (a) communication overhead (b) query performance (c) error or signal distortion from lossy compression (d) computation overhead (e) storage cost . Combinations of the above parameters can be tuned to specific sensor network deployments.

We have explored configurations that compress based primarily on either communication cost or error. Bounding the communication cost at each level of the hierarchy ensures that clusterheads do

not incur significantly higher communication overhead than other nodes in the network. The compression error can, however, vary depending on the spatio-temporal correlation in the data. For example, an area with high correlation may be described with low error within the specified communication bound, but a region with low correlation would incur greater error. Conversely, bounding the compression error would result in the compression adapting to the correlation in different regions, but unbounded communication overhead if the data is highly uncorrelated. We are exploring joint optimization criteria that include both these parameters.

In a different scenario, a sensor network might consist of low-end, highly resource constrained devices (such as motes [13]), where computation, memory and storage are highly constrained, in additional to energy. In these networks, the hierarchy can be constructed by propagating approximation coefficients of the wavelet transform, which is computationally less expensive than subband coding.

3.5 Load Balancing and Robustness

Due to space constraints, we will introduce only a flavor of techniques that we are exploring in this area. As discussed above, bounding the communication overhead at each level of the decomposition can reduce the uneven distribution of energy consumption. Additional load-balancing can be provided by periodically hashing to a new apex location, implicitly changing the choice of clusterheads at different levels.

A strictly hierarchical configuration (as described above) is vulnerable to node failures, since loss of a node cuts off data from any of its children. We are therefore exploring decentralized, peer-based structures. One approach we are considering communicates summarized data to all members of the next higher level, rather than to just the clusterhead. Such an approach will have higher overall energy consumption and will require greater data compression, but it becomes immune to node failure and is naturally load balanced since data is replicated equally to all nodes.

4. USAGE MODELS

A distributed multi-resolution storage infrastructure benefits search and retrieval of datasets that exhibit spatial correlation, and applications that use such data.

Multi-Resolution Data Extraction: Multi-resolution data extraction can proceed along the hierarchical organization by first extracting and analyzing low-resolution data from higher level clusterheads. This analysis can be used to obtain high-resolution data from sub-regions of the network if necessary.

Querying for Spatio-Temporal features: The hierarchical organization of data can be used to search for spatio-temporal patterns efficiently by reducing the search space. For example, consider the search for pooling of contaminant flow. Such a feature has a large spatial span, and therefore significant energy benefit can be obtained by querying only a few clusterheads rather than the entire network. Temporal patterns can be efficiently queried by drilling down the wavelet hierarchy by eliminating branches whose wavelet coefficients do not partially match the pattern, thereby reducing the number of nodes queried. Summarized coefficients that result from wavelet decomposition have been found to be excellent for approximate querying [14, 7], and to obtain statistical estimates from large bodies of data (Section 6).

Feature Routing and Edge Detection: Target tracking and routing along spatio-temporal patterns such as temperature contours, have been identified as compelling sensor network applications. The problem of edge detection has similar requirements, and is important for applications like geographic routing, localization, beacon placement and others, where explicit knowledge of edges can improve performance of the algorithm. Our architecture can be used to assist these applications, since it good at identifying discontinuities. By progressively querying for the specific features, communication overhead of searching for features can be restricted to only a few nodes in the network.

Debugging: Debugging data is another class of datasets that exhibits high correlation. Consider packet throughput data: throughput from a specific transmitter to two receivers that are spatially proximate are closely correlated; similarly, throughput from two proximate transmitters to a specific receiver are closely correlated. Our system serves two purposes for these datasets: (a) they can be used to extract aggregate statistics (Section 6) with low communication cost, and (b) discontinuities represent network hotspots, deep fades or effects of interference, which are important protocol parameters, and can be easily queried.

5. COMMUNICATION, COMPUTATION AND STORAGE COMPLEXITY

In this section, we provide a back-of-the-envelope comparison of the benefits of performing a hierarchical wavelet decomposition over a centralized data collection strategy. Our description will only address the average and worst case communication load on the nodes in the network, while Table 3 provides a more complete summary of the communication, computation and storage complexity.

A square grid topology with n nodes is considered ($\sqrt{n}$ side as shown in Figure 1(b)), where $\sqrt{n} = 2^{\lambda}$. Clusterheads at each step are selected to be the one closest to the lower left-corner of the tile. Communication is only along edges of the grid, each of which is of length 1 unit. The cost of transmission and reception of one unit of data along a link of length 1, costs 1 unit of energy. The chosen cost metric is communication bandwidth, and each clusterhead is constrained transmit at most D_0 data. While realistic topologies are far from as regular as the one proposed [15], and the cost model can be more complicated, the simple case captures essential tradeoffs in construction of multi-resolution hierarchies.

Communication Overhead: The overhead for *centralized* decomposition can be computed from the total number of edges traversed on the grid ($n(\sqrt{n}-1)$ for a $\sqrt{n}$x$\sqrt{n}$ grid) The total cost, therefore, is $n(\sqrt{n}-1)D_0$, giving an *Average-case* communication complexity of $O(\sqrt{n}D_0)$. The worst communication overhead is the storage node itself, or the sensor node(s) closest to the storage node, if the storage node is not power constrained. Thus, the *Worst-case* communication complexity is $O(n^{1.5}D_0)$.

To compute the overhead of the *hierarchical* decomposition, we use the fact that at level l of the decomposition, a communication edge is of length 2^{l-1}, and there are $2^{2\lambda-2l}$ tiles (computed as $\frac{Area\ of\ Region}{Area\ of\ Tile}$), giving a communication overhead of $2^{2\lambda+1-l}$. Over λ levels of decomposition, the total cost is, thus, $\sum_{0\leq l\leq\lambda} 2^{2\lambda+1-l} = 2^{\lambda+1}(2^{\lambda}-1)$. Thus, the total communication cost is $O(nD_0)$ and the *Average-case* communication cost is $O(D_0)$. In the worst case, a node is on the forwarding path at every level of the hierarchy. Since there are λ levels and each level forwards data $3D_0$ to a clusterhead, worst case data passing through a node is $3\lambda D_0 = 3D_0 \log n$. Thus, *Worst-case* Communication Complexity = $O(D_0 \log n)$.

6. SAMPLE DATA ANALYSIS

Our system is currently under development, but following are some initial results from offline analysis of experimental data. We provide initial results from two different sensor datasets to demon-

	Centralized		Hierarchical	
	Avg. Case	Worst Case	Avg. Case	Worst Case
Communication	$O(\sqrt{n}D_0)$	$O(n^{1.5}D_0)$	$O(D_0)$	$O(D_0 \log n)$
Computation	$O(D_0 \log D_0)$	$O(nD_0)$	$O(D_0 \log D_0)$	$O(D_0 \log(nD_0))$
Storage	$O(D_0)$	$O(nD_0)$	$O(D_0)$	$O(D_0 \log n)$

Table 3: Communication, Computation, and Storage Overhead

strate the practical applicability of our system.

6.1 Precipitation Dataset

The first sensor dataset that we consider is a geospatial dataset that provides 50 km resolution daily precipitation for the Pacific NorthWest from 1949-1994 [16]. While the dataset is at a significantly larger scale than densely deployed sensor networks, the data exhibits spatio-temporal correlations, providing a useful performance case study.

The setup comprises a 15x12 grid of nodes, each recording daily precipitation values. A wide set of queries can be envisaged on such a dataset: (a) range-sum queries, such as total precipitation over a specified period from a single node or a region; (b) drill-down max queries, to efficiently access the node or region that receives maximum precipitation over a specified period; (c) drill-down edge queries, to efficiently access and query nodes on an edge between a high-precipitation and low-precipitation region.

The hierarchy construction proceeds as outlined in Section 3, with nodes at level 0 performing a one-dimensional temporal wavelet subband decomposition, and clusterheads at other levels combining data from the previous level, and performing a three-dimensional decomposition. We use communication bandwidth as the cost metric for the compression. Thus, at level i, data from four level $i-1$ nodes are combined, subband coded, thresholded, quantized, and losslessly compressed using Run-Length encoding and Huffman encoding to fit within the specified target bandwidth. An approximate target communication bandwidth is chosen since it is computationally intensive to choose parameters to exactly fit the target bandwidth.

In the following example, we look at the performance of the hierarchy for a specific range-sum query: "Find the annual precipitation between 1949-1994 for each sensor node". We use two performance metrics:

- *Compression Ratio* at level i is ratio of number of bits of transmitted data to the raw data size. Table 4 shows that the number of bits transmitted is approximately the same at each level of the hierarchy, giving us large compression ratios, especially at higher levels since more data is combined into the target bandwidth.

- *Query error* at level i is calculated as $\frac{|measured-true|}{true}$. This metric corresponds to the accuracy in the response for the range-sum query when clusterheads at level i are queried. While the error increases at higher levels of the hierarchy, it is still reasonably low at all layers. For example, 80% of measurements are within 30% of the true answer for level 1, and 80% of measurements are within 50% at level 3. We suggest that that such error is sufficient to make preliminary searchs and then, if desired, drill-down with more detailed queries.

6.2 PacketLoss Dataset

We now look at the performance of our system on a packet throughput dataset from a 12x14 grid of nodes with grid spacing 2 feet

Level	*Raw data size (Kbits)*	*Mean data sent to next level (Kbits)*	*Compression Ratio (Ratio of raw data size to transmitted data size)*
1	262.5	5.6	46.8
2	984.4	3.8	257.2
3	3937.7	4.0	987
4	11813.2	5.2	2286.2

Table 4: Compression Result for Sum Query

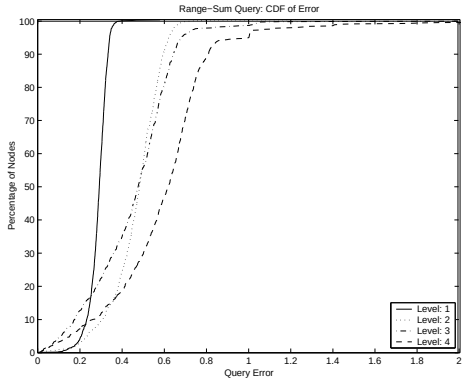


Figure 4: Query Error for Range-Sum Query

(detailed descriptions can be obtained from [15]). Each node has throughput data from all other transmitters, for twenty different settings of transmit signal strength from each node. Two forms of correlations can be exploited to reduce size of data: (a) correlation in throughput between adjacent transmitters to a particular receiver, and (b) correlation in throughput from a single transmitter to adjacent receivers.

Figure 5 shows the results of a query to obtain the throughput vs distance map from the compressed data. The error is small in the approximated data, and gives us large compression benefit. These results show that our algorithm works for a broad class of data with different correlation characteristics.

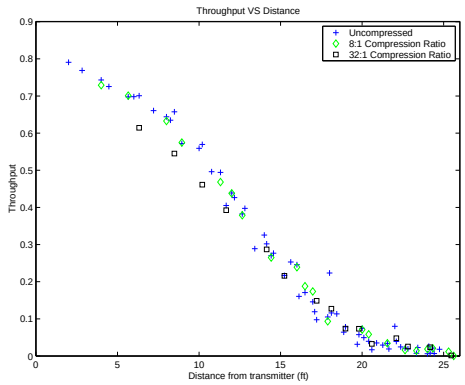


Figure 5: Throughput VS Distance

7. RELATED WORK

In building this system, we draw from significant research in information theory [5, 6, 17], databases [7, 14], spatio-temporal data mining [18, 19], and previous work in sensor networks [11, 20]. [14] and others have used wavelet-based synopsis data structures for approximate querying of massive data sets. Quadtrees are popularly used in image processing and databases [19], and resemble the hierarchies that we build up. Other data structures such as R-trees, kd-trees and their variants [18] are used for spatial feature indexing. Triangular Irregular Networks (TIN [18]) is used in for multi-layered processing in cartography, and other geographical datasets. Some of these structures could find applicability in our context, and will be considered in future work. [11] proposes a sensor network storage architecture that leverage the excellent lookup complexity of distributed hash tables (DHT) for event storage. DHTs are useful when queries are not spatio-temporal in nature, while our system organizes data spatially to enable such queries.

8. SUMMARY AND RESEARCH AGENDA

This paper made the case for DIMENSIONS, a large-scale distributed multi-resolution storage system that exploits spatio-temporal correlations in sensor data. Many important research issues still need to be addressed:

What are the benefits from temporal and spatial data compression? To fully understand correlations in sensor data, and the benefits provided by temporal and spatial compression, we are deploying large scale measurement infrastructure, for sensor data collection in realistic settings,

What processing should be placed at different levels of the hierarchy? The clusterheads at various levels of the hierarchy can interpret the data at different scales, to obtain information about spatio-temporal patterns. A challenging problem is the development of algorithms for spatio-temporal interpretation of data.

How can we increasing compression benefit without sacrificing query performance? Our system can be augumented with other compression techniques such as delta coding, and recent work on blind source coding [17], to obtain better data compression. Correlation statistics can be learnt by clusterheads, and exploited to reduce data at the source.

How can we obtain energy savings from distributed compression? Better compression doesn't necessarily translate into energy savings in communication since the cost of passive listening is comparable to transmission [1]. Obtaining energy savings in communication of data involves (a) reducing size of data to be transmitted (b) scheduling communication to minimize listen time as well as transmit time. How do we schedule communication to translate compression benefit to energy benefit?

How can this research be applied to other networked systems? These techniques may have relevance to other systems where data is correlated, and massive distributed datasets need to be queried, such as scientific applications of Grid Computing [21].

Acknowledgements

We would like to thank Stefano Soatto for very useful feedback on theoretical aspects of our system. We would also like to thank members of the LECS lab for providing feedback on the paper.

9. REFERENCES

[1] G.J. Pottie and W.J. Kaiser. Wireless integrated network sensors. *Communications of the ACM*, 43(5):51–58, May 2000.

[2] M. Kohler. UCLA factor building. CENS Presentation.

[3] A. Cerpa, J. Elson, D. Estrin, L. Girod, M. Hamilton, and J. Zhao. Habitat monitoring: Application driver for wireless communications technology. In *Proceedings of the 2001 ACM SIGCOMM Workshop on Data Communications in Latin America and the Caribbean*, April 2001.

[4] A. Chankhunthod, P.B. Danzig, C. Neerdaels, M.F. Schwartz, and K.J. Worrell. A hierarchical internet object cache. In *USENIX Annual Technical Conference*, pages 153–164, 1996.

[5] R. M. Rao and A. S. Bopardikar. *Wavelet Transforms: Introduction to Theory and Applications*. Addison Wesley Publications, 1998.

[6] M. Vetterli and J. Kovacevic. *Wavelets and Subband coding*. Prentice Hall, New Jersey, 1995.

[7] J. S. Vitter, M. Wang, and B. Iyer. Data cube approximation and histograms via wavelets,. In D. Lomet, editor, *Proceedings of Seventh International Conference on Information and Knowledge Management (CIKM'98)*, pages 69–84, Washington D.C, November 1998.

[8] J. Kubiatowicz, D. Bindel, Y. Chen, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, and B. Zhao. Oceanstore: An architecture for global-scale persistent storage. In *Proceedings of ACM ASPLOS*. ACM, November 2000.

[9] A. Rowston and P. Druschel. Storage management and caching in past, a large-scale, persistent peer-to-peer storage utility. In *18th ACM SOSP*, volume 1, Lake Louise, Canada, October 2001.

[10] I. Gupta, R. van Renesse, and K. Birman. Scalable fault-tolerant aggregation in large process groups, 2001.

[11] S. Ratnasamy, D. Estrin, R. Govindan, B. Karp, L. Yin S. Shenker, and F. Yu. Data-centric storage in sensornets. In *ACM First Workshop on Hot Topics in Networks*, 2001.

[12] B. Karp and H. T. Kung. GPSR: greedy perimeter stateless routing for wireless networks. In *Proceedings of Mobicom*, 2000.

[13] J. Hill, R. Szewczyk, A Woo, S. Hollar, D. Culler, and K. Pister. System architecture directions for networked sensors. In *Proceedings of ASPLOS-IX*, pages 93–104, Cambridge, MA, USA, November 2000. ACM.

[14] K. Chakrabarti, M. Garofalakis, R. Rastogi, and K. Shim. Approximate query processing using wavelets. *VLDB Journal: Very Large Data Bases*, 10(2–3):199–223, 2001.

[15] D. Ganesan, B. Krishnamachari, A. Woo, D. Culler, D. Estrin, and S. Wicker. Complex behavior at scale: An experimental study of low-power wireless sensor networks. Technical Report CSD-TR 02-0013, UCLA, 2002.

[16] M. Widmann and C.Bretherton. 50 km resolution daily preciptation for the Pacific Northwest, 1949-94.

[17] S. Pradhan, J. Kusuma, and K. Ramchandran. Distributed compression in a dense sensor network. *IEEE Signal Processing Magazine*, 1, March 2002.

[18] M. Krevald, J. Nievergelt, T. Roos, and P. Widmayer. *Algorithmic Foundations of Geographic Information Systems*. Springer-Verilag, 1997.

[19] R. Agrawal, C. Faloutsos, and A.N. Swami. Efficient Similarity Search In Sequence Databases. In D. Lomet, editor, *Proceedings of the 4th International Conference of Foundations of Data Organization and Algorithms (FODO)*, pages 69–84, Chicago, Illinois, 1993. Springer Verlag.

[20] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In *Proceedings of Mobicom*, pages 56–67, Boston, MA, August 2000. ACM Press.

[21] I. Foster, C. Kesselman, J. Nick, and S. Tuecke. Grid services for distributed system integration. *Computer*, 35(6), 2002.

Wireless Sensor Networks: A New Regime for Time Synchronization

Jeremy Elson
Department of Computer Science
University of California, Los Angeles
Los Angeles, California, USA
jelson@cs.ucla.edu

Kay Römer
Department of Computer Science
ETH Zurich
8092 Zurich, Switzerland
roemer@inf.ethz.ch

ABSTRACT

Wireless sensor networks (WSNs) consist of large populations of wirelessly connected nodes, capable of computation, communication, and sensing. Sensor nodes cooperate in order to merge individual sensor readings into a high-level sensing result, such as integrating a time series of position measurements into a velocity estimate. The physical time of sensor readings is a key element in this process called data fusion. Hence, time synchronization is a crucial component of WSNs. We argue that time synchronization schemes developed for traditional networks such as NTP [23] are ill-suited for WSNs and suggest more appropriate approaches.

Keywords

time synchronization, wireless sensor network

1. INTRODUCTION

Wireless sensor networks [1] are an increasingly attractive means to bridge the gap between the physical and virtual world. A WSN consists of large numbers of cooperating small-scale nodes, each capable of limited computation, wireless communication, and sensing. In a wide variety of application areas including geophysical monitoring, precision agriculture, habitat monitoring, transportation, military systems and business processes, WSNs are envisioned to be used to fulfill complex monitoring tasks. With this new class of networks also come new challenges in many areas of the system's design.

Sensor nodes are small-scale devices; in the near future, low-cost platforms with volumes approaching several cubic millimeters are expected to be available [17]. Such small devices are very limited in the amount of energy they can store or harvest from the environment. Thus, *energy efficiency* is a major concern in a WSN. In addition, many thousands of sensors may have to be deployed for a given task—an individual sensor's small effective range relative to a large area of interest makes this a requirement, and its small form factor and low cost makes this possible. Therefore, *scalability* is another critical factor in the design of the system. To achieve scalability, an important design principle is *locality*—all but the smallest networks can not depend on having global state. In [5], Cerpa reports network saturation when as few as 40 sensor nodes use global broadcasts.

In contrast to traditional wired networks, a WSN is both highly *dynamic* and *ad hoc*. For example, initial deployment may involve throwing nodes from an aircraft into an area of interest at random. Over time, sensors fail due to destruction or battery depletion; new sensors might be added in higher concentration in areas of interest. Sensors experience changes in their position, available energy, and task details. Changes in the environment can dramatically affect radio propagation, causing frequent network topology changes and network partitions. At high densities, WSNs also become much more likely to suffer communication failures due to contention for their shared communication medium—in [12] Ganesan reports a message loss of 20% and above between adjacent nodes in a dense WSN. These factors lead to strong *self-configuration* and *robustness* requirements in a WSN. Static configuration is unacceptable; the system must continuously adapt to make the best use of available resources.

While individual sensor nodes have only limited functionality, the global behavior of the WSN can be quite complex. The true value of the network is in this property of *emergent behavior*: the functionality of the whole is greater than the sum of its parts. This is achieved, in part, through *data fusion*, the process of transforming and merging individual sensor readings into a high-level sensing result. This is where time synchronization enters the stage, as it plays a key role in many types of data fusion.

Time synchronization in distributed systems is a well-studied problem. Many solutions exist for traditional networks and distributed systems. NTP [23], for example, has been widely deployed and proven effective and robust in the Internet. In this paper, we explore the question: do the traditional methods apply in sensor networks as well? Our answer is no. Many assumptions on which existing schemes are based no longer hold in this new area of WSNs. We claim that something new is needed.

The organization of the remainder of this paper is as follows. In Section 2, we discuss in more detail the applications and requirements of synchronized time in a WSN. We then review existing time synchronization schemes in Section 3, and examine their shortcomings when applied in this new context. In Section 4, we describe general design principles for WSN time synchronization, based on experiences with a number of prototype systems built by the authors. Finally, in Section 5, we draw our conclusions and describe future work.

2. SYNCHRONIZED TIME IN A WSN

Time synchronization is an important feature of almost any distributed system. A confluence of factors makes flexible and robust time synchronization particularly important in WSNs, while simultaneously making it more difficult to achieve than in traditional networks. In this section, we will describe some of these factors: the tight link between sensors and the physical world; the scarcity of system energy; the need for large-scale, decentralized topologies; and unpredictable, intermittent connectivity.

The advent of logical time [19, 22] eliminated the need for physical time synchronization in situations where only causal relationships of events are of interest to the application. However, logical time only captures relationships between "in system" events, defined by message exchanges between event-generating processes. This is not the case for phenomena sensed by the nodes in a WSN; physical time must be used to relate events in the physical world. Logical time is not sufficient in the WSN domain. For example, consider the following applications:

- Object tracking: The size, shape, direction, location, velocity, or acceleration of objects is determined by fusing proximity detections from sensors at different locations.
- Consistent state updates: The current state of an object is most accurately determined by the node that has "sighted" the object most recently.
- Distributed beamforming: beam-forming arrays [33] can perform "spatial filtering," receiving only signals arriving from a certain direction. This depends on the relative time offsets of the array's sensors.
- Duplicate detection: The time of an event helps nodes determine if they are seeing two distinct real-world events, or a single event seen from two vantage points.
- Temporal order delivery: Many data fusion algorithms must process events in the order of their occurrence [27]—for example, Kalman filters.

Another illustrative example is the formation of a TDMA schedule for low-energy radio operation. This is an important application because listening and transmitting are both very energy-expensive operations in a low-power radio. A common technique to conserve precious energy is to turn the radio off, waking up only briefly to exchange short messages before going back to sleep [25, 29].

Consider two nodes that have agreed to rendezvous on the radio channel once every 60 seconds to exchange a short message—say, 8 bits representing the current temperature. Using a 19.2kbit/sec radio such as our testbed's RF Monolithics [4], 8 bits can be transmitted in about $0.5ms$. However, in practice, the radio must be awakened early to account for time synchronization error—so an expectation of a $1ms$ phase error will triple the total amount of time the radio is expending energy listening to the channel. In addition, even assuming perfect synchronization at the start of a sleep period, a typical quartz oscillator on such a sensor will drift on the order of 1 part in 10^5 [32], or $0.6ms$ after 60 seconds. Of course, sending synchronization packets during the sleep period defeats the purpose of sleeping, so we must consider frequency estimation as part of the time synchronization problem.

The examples above demonstrate not only the importance of time synchronization in a WSN, but also one of its difficulties: any resource expended for synchronization reduces the resources available to perform the network's fundamental task. Many current data acquisition systems do not have this constraint, so they often rely on high-energy solutions to the synchronization problem—frequent network synchronization, high stability frequency standards, GPS receivers, and so forth. In a WSN, the impact of such solutions—in terms of energy, cost, and form-factor—can make them untenable.

Another important aspect of the problem domain illustrated by our examples is the heterogeneity of the application requirements over a wide variety of axes. For example:

- *Energy utilization*. Some synchronization schemes require extra, energy-hungry equipment (e.g., GPS receivers). Others may have virtually no energy impact (e.g., listening to extant packets already being transmitted for other reasons).
- *Precision*—either the dispersion among a group of peers, or maximum error with respect to an external standard. The precision might be as fine as microseconds (e.g., coherent signal processing on audio signals) or as coarse as seconds (e.g., tracking a slow-moving human).
- *Lifetime*—the duration for which nodes are synchronized. This might be nearly instantaneous (e.g., to compare views of a single event from multiple vantage points), as long-lived as the network (to track the motion of an object through a sensor field), or persistent forever (e.g., UTC).
- *Scope and Availability*—the geographic span of nodes that are synchronized, and completeness of coverage within that region. The scope might be as small as a pair of nodes exchanging data, or as large as the entire network.
- *Cost and Size*. These factors can make a scheme a nonstarter. It is unreasonable to put a $100 GPS receiver or a $1000 Rubidium oscillator on a disposable sensor node that would otherwise cost $10, or on dust-mote sized nodes.

The exact requirements for WSN time synchronization along these axes can not be characterized in general. The requirements are highly application-domain specific and vary over time in unpredictable ways, since they are influenced by the sensed phenomenon.

Given these new challenges, are traditional time synchronization schemes the best choice for this new domain?

3. TRADITIONAL NETWORK TIME SYNCHRONIZATION

Over the years, many protocols have been designed for maintaining synchronization of physical clocks over computer networks [7, 15, 23, 30]. These protocols all have basic features in common: a simple connectionless messaging protocol; exchange of clock information between clients and one (or a few) servers; methods for mitigating the effects of nondeterminism in message delivery and processing; and an algorithm on the client for updating local clocks based on information received from a server. They do differ in certain details: whether the network is kept internally consistent or synchronized to an external standard; whether the server is considered to be the canonical clock, or merely an arbiter of client clocks, and so on.

Some wireless standards such as 802.11 [16] have similar time-synchronization beacons built into the MAC layer. Work by Mock *et al.* [24] extends 802.11's synchronization by taking advantage of the broadcast property of wireless networks. This technique is notable because it leverages domain knowledge to increase precision; we will argue in Section 4.5 that this is an important design goal. However, these 802.11 methods do not work beyond a single broadcast domain, a serious limitation.

Mills' NTP [23] stands out by virtue of its scalability, self-configuration for creating a global timescale in multihop networks, robustness to various types of failures, security in the face of deliberate sabotage, and ubiquitous deployment. For decades, it has kept the Internet's clocks ticking in phase. Many people in the WSN research community often ask: "Why not use NTP here, too?" At least one research group has moved in this direction, implementing an NTP-like time service over small wireless sensors [11]. But is this truly the best choice? Many of the assumptions that NTP makes, while true in the Internet, are not true in sensor networks. We explore some of these differences below.

3.1 Energy Aware

As explained in Section 1, energy efficiency is a major concern in a WSN. The energy constraints violate a number of assumptions routinely made by classical synchronization algorithms:

- Using the CPU in moderation is free.
- Listening to the network is free.
- Occasional transmissions have a negligible impact.

These assumptions are true in traditional networks and consequently have become fundamental to schemes such as NTP. For example, NTP assumes that the CPU is always available, and performs frequency discipline of the oscillator by adding small but continuous offsets to the system clock. In addition, NTP makes no effort to predict the time at which packets will arrive; it simply listens to the network all the time. And, while it is conservative in its use of bandwidth, it assumes a continuous ability to transmit packets. (It can "free-run" without network access, but requires a significant time with network access restored before it achieves its original accuracy again.)

[25] describes why most of the above assumptions do not hold in a WSN. In a low-power radio, listening to, sending to, receiving from the network all require significant energy compared to the overall system budget. CPU cycles are also a scarce resource; the limited energy mandates the use of slow processors which spend most of their time powered down (awakened by a pre-processor after an event of interest).

3.2 Single-Hop vs. Multi-Hop

Much of the non-Internet ("LAN") work in distributed clock agreement assumes that all nodes in the system can directly exchange messages—or, more precisely, that a single latency and jitter bound is common to all messages in the system. Some methods that exploit the broadcast property of the physical media [24, 31] do not speak to the problem of federating the clocks of multiple (overlapping) broadcast domains.

Sensor networks span many hops; the end-to-end latency is much larger than a single hop. This makes it difficult to apply methods that assume a fully connected or low-latency topology.

3.3 Infrastructure-Supported vs. Ad Hoc

NTP allows construction of time synchronization hierarchies, each rooted at one of many canonical sources of external time in the Internet. The canonical sources ("Stratum 1" servers, in NTP terminology) are synchronized with each other via a variety of "out of band" mechanisms—for example, radio receivers for time signals from the Global Positioning System [18], or the WWVB radio broadcast [3]. This infrastructure provides a common view of a global timescale (UTC) to the Stratum 1 servers throughout the Internet. Consequently, nodes throughout the Internet enjoy being synchronized to a single, *global* timescale while rarely finding themselves more than a few hops away from a *local* source of this canonical time.

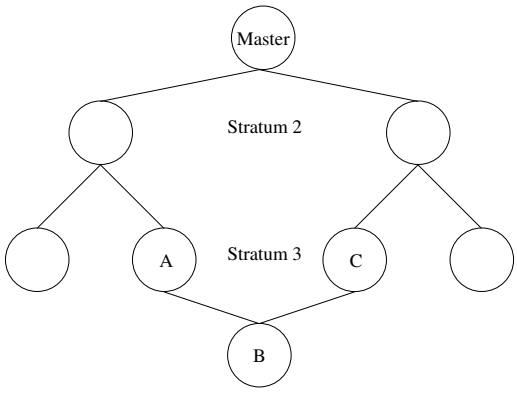


Figure 1: A global timescale can lead to poorly synchronized neighbors, if the neighbors are far from the master clock and have uncorrelated loss due to divergent synchronization paths.

WSNs, on the other hand, may often consist of large-diameter networks *without* an external infrastructure. Often it is not an option to equip sensor nodes with receivers for "out of band" time references. GPS, for example, is expensive both in terms of energy consumption and component cost, since it needs high-performance digital signal processing capabilities. Moreover, it requires a line of sight to the GPS satellites—which is not available inside of buildings, beneath dense foliage, underwater, on Mars, etc.

In this scenario, NTP-style algorithms must create a hierarchy rooted at a single node that is designated as the system's master clock. Even assuming we have an algorithm that automatically maintains such a hierarchy in the face of node dynamics and partitions, there is still a fundamental problem: with a single source of canonical time, most nodes will be far away from it. Nodes that are far away from the master clock will be poorly synchronized to the global timescale.

This is a particularly bad situation in a WSN, where nodes closest to each other are often the ones that need the most precise synchronization—e.g., for distributed acoustic beamforming. Consider the scenario shown in Figure 1. Nodes A, B, and C are close to one another, but far away from the master clock. In a scheme such as NTP, B will choose either A or C as its synchronization source. Either choice will lead to poor synchronization when sharing data with the opposite neighbor. For example, if B synchronizes to C, its synchronization error to A will be quite large; the synchronization path leads all the way to the master and back. As we will discuss in Section 4.2, these constraints suggest that WSNs should have *no global timescale*. Instead, we propose that each node in an WSN maintain an *undisciplined* clock, augmented with relative frequency and phase information to each of its local peers.

3.4 Static Topology vs. Dynamics

Although the Internet suffers from transient link failures, the topology remains relatively consistent from month to month, or year to year. Typically, NTP clients are manually configured with a list of "upstream" sources of time. Although NTP automatically uses statistical means to decide on the best of its given options, it still depends on being configured with some notion of which nodes are peers and which lie upstream.

The network dynamics in WSN prevent such a simple kind of static configuration. Moreover, the need for unattended operation of WSN prevents a manual configuration of individual nodes.

3.5 Connected vs. Disconnected

Node mobility, node failures, and environmental obstructions cause a high degree of dynamics in a WSN. This includes frequent network topology changes and network partitions. Data may still flow through the network despite these partitions, as mobile nodes transport information by physically moving within the network. However, the resulting paths of information flow might have unbounded delays (depending on the movement of the node relaying the information) and are potentially unidirectional, since there might not be any nodes moving in the opposite direction.

This kind of message relaying might seem like an unlikely case. However, in a sparse WSN where sensor nodes are attached to moving objects or creatures (e.g., humans, animals, vehicles, goods) or deployed in moving media (e.g., air, water) this is a major mode of communication [2, 6, 13, 20]. Grossglauser and Tse [14] even show that the communication capacity of a WSN approaches zero with increasing node density unless messages are being relayed in this way.

As we will show below, message relaying is a serious problem for traditional clock synchronization algorithms, since they rely on two important assumptions:

1. Nodes are connected before the time they need to be synchronized.
2. The message delay between two (not necessarily adjacent) nodes to be synchronized can be estimated over time with high precision.

Consider for example Figure 2, which models a water pollution monitoring WSN deployed in a river. At real–time t_1 device 1 detects an oil stain. At t_2 device 2 detects the same oil stain. At t_3 device 2 passes by device 3, a communication link is established, and E_2 is sent to device 3. At t_4 device 1 passes by device 3, a link is established, and E_1 is sent to device 3.

If device 3 wants to determine direction of movement and size of the oil stain, it has to determine whether E_1 happened after E_2 and the time difference between E_1 and E_2. This scenario presents a serious problem for classical clock synchronization algorithms that assume that the device's clocks will be synchronized *a priori* when they sense events E_1 and E_2. However, as shown in figure 2, there is no way for nodes 1 and 2 to communicate for all $t \leq t_3$, which makes clock synchronization of nodes 1 and 2 impossible before E_1 and E_2 are sensed. This violates the first of the above assumption made by classical clock synchronization algorithms.

Even at time t_4, where an unidirectional delayed message path from node 2 to node 1 via node 3 exists, clock synchronization of nodes 1 and 2 seems almost impossible with traditional algorithms. The path is unidirectional and arbitrarily delayed—wreaking havoc with traditional clock synchronization algorithms that assume they can estimate the message delay over time (or, that assume the delay is negligible), thus violating the second of the above assumptions. The highly unreliable communication in WSNs further contributes to arbitrary delays on multihop paths.

4. DESIGN PRINCIPLES FOR WSN TIME SYNCHRONIZATION

Having described the shortcomings of traditional time synchronization schemes in the previous section, we can now begin to formulate requirements and new directions for time synchronization in WSNs. There are not yet any proven solutions for time synchronization in deployed WSNs. However, the authors have developed techniques which might prove helpful in solving this problem [8, 9, 26]. These techniques aim to build a synchronization service that conforms to the requirements of WSNs:

- *Energy efficiency*—the energy spent synchronizing clocks should be as small as possible, bearing in mind that there is significant cost to continuous CPU use or radio listening.
- *Scalability*—large populations of sensor nodes (hundreds or thousands) must be supported.
- *Robustness*—the service must continuously adapt to conditions inside the network, despite dynamics that lead to network partitions.
- *Ad hoc deployment*—time sync must work with no *a priori* configuration, and no infrastructure available (e.g., an out-of-band common view of time).

4.1 Multi-Modal, Tiered, and Tunable

The services provided by various proposals for WSN time synchronization fall into many disparate points in the parameter space we described in Section 2 (energy, precision, scope, lifetime, and cost). Each scheme has tradeoffs—no single method is optimal along all axes. For example:

- Typical GPS receivers can synchronize nodes to a persistent-lifetime timescale that is Earth-wide in scope to a precision of 200ns [21]. However, the receivers can require several minutes of settling time, and may be too large, costly, or high-power to justify on a small sensor node. In addition, the GPS infrastructure is not always available (§3.3).
- Römer's scheme described in [26] achieves 1ms precision, creates an instantaneous timescale with little overhead, and works on unidirectional links. However, the synchronization is localized and rather short-lived.
- Elson's RBS [9] can achieve 1μs precision and sufficient frequency estimates to extend the timescale for several minutes. It synchronizes all nodes within a broadcast domain. However, it requires a bidirectional broadcast medium and several packet exchanges.
- The multihop extension to RBS described in [9] allows the timescale to be extended across multiple broadcast domains, but at the cost of (somewhat) degraded accuracy.

None of these methods can be considered the *best*; each has advantages and disadvantages. The details of a particular application and hardware will dictate the method that should be used in each situation.

Still more options arise when several methods are composed into a multi-modal system. For example, we might equip a small portion of nodes with more expensive high-stability oscillators, and use RBS to allow nearby nodes to estimate their own frequency with respect to the reference [9]. This type of tiered architecture is analagous to the memory hierarchy found in modern computers (registers, memory cache, main memory, disk), where the goal is to build a system that appears to be as fast as the registers, but as large and cheap as the disk.

Ideally, we would like to have a large enough palette of methods available so that we can choose an overall scheme that is both *necessary and sufficient* for the application on all axes. Unnecessary synchronization wastes resources; insufficient synchronization leads to poor application performance. To this end, it is also important that WSN synchronization be *tunable*—providing adjustable parameters that allow a closer match between the type of synchronization needed and that which is provided.

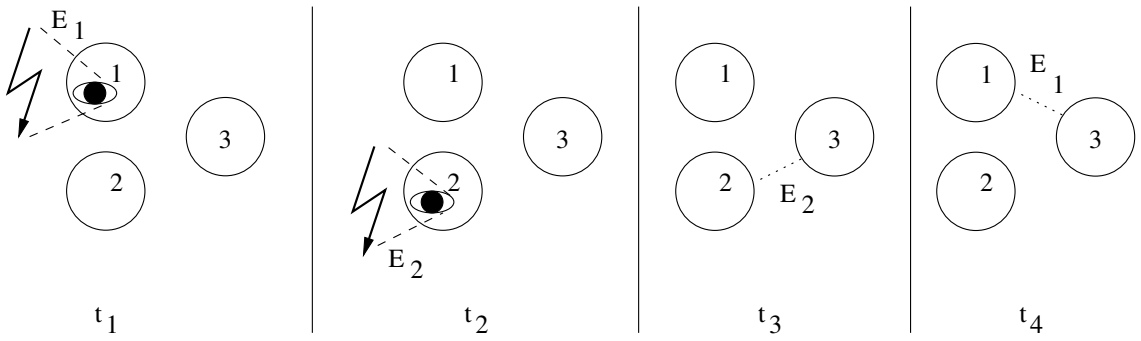


Figure 2: A disconnected network leading to time synchronization problems in a WSN

4.2 No Global Timescale

We argued in Section 3.3 that keeping a global timescale for a large network is only effective when many canonical sources of that timescale are available throughout the network. In the infrastructure-free world of an WSN, where we can not rely on such out-of-band timescale distribution, classical algorithms end up in the situation we illustrated in Figure 1.

Our claim is that the best solution is for each node to keep its own timescale. A node never sets its clock or disciplines its frequency, but rather lets it run at its natural rate. WSN time synchronization schemes—regardless of the underlying method—should only build up a table of parameters relating the phase and frequency of the local clock to other clocks in the system. Local and remote timestamps can then be compared to each other using these parameters for conversion. In fact, time conversion can be built into the packet forwarding mechanism itself. That is, nodes can perform successive time conversions on packets as they are forwarded from node to node—keeping timestamps with respect to the local clock at each hop.

This technique has a number of advantages. First, the synchronization error between two nodes is proportional to the distance between them—not their distance to a master clock, which might be much greater. Second, allowing the local clock to run undisciplined means that each node can enjoy a monotonic clock—a critical feature to many signal processing algorithms. While frequency drift will occur due to the oscillator's instability due to temperature, shock, and voltage variations, there will be no sudden changes in the frequency or phase due to new information arriving at a disciplining algorithm. (Network timesync can produce an estimate of the oscillator's frequency relative to an SI second if needed for data analysis.) Finally, an undisciplined clock requires no continuous corrections to the clock by the CPU or kernel, as are required by algorithms such as NTP. This is important for energy conservation, as we saw in Section 3.1.

4.3 Post-Facto Synchronization

Traditional time synchronization schemes synchronize node clocks *a priori*; clocks are pre-synchronized when an event occurs and is timestamped. As we saw earlier, this causes problems with message relaying and makes it hard to exploit time-variable and unpredictable application knowledge. In contrast, we advocate *post-facto synchronization*, where clocks run unsynchronized at their own natural rates. When timestamps from different clocks need to be compared, they can be reconciled after the fact [8]. This removes the need to predict application requirements in advance; instead, synchronization energy is only expended after an event of interest has occurred. Also, this approach enables support for message relaying, since it does not require network connectivity between event-generating nodes.

Time synchronization is comparable in some sense to routing in ad hoc networks. There, *proactive* routing establishes and maintains routes between nodes in advance, whereas *reactive* routing only establishes routes on-demand between pairs of nodes that want to communicate.

4.4 Adapt to the Application

In Section 4.1, we argued that scalable and energy-efficient WSN time synchronization should be achieved by closely matching the application requirements along axes such as scope, lifetime, and precision. We have also seen a number of techniques that provide service in different parts of this space. However, application requirements vary over time and are in general not predictable, since they depend on the sensed phenomena. Choosing and tuning a necessary and sufficient form of synchronization is a non-trivial problem. To some degree, the application requirements of time syncronization must be built in at design-time. However, dynamics of the application and the environment are likely to dictate that automatic adaptation at run-time is also necessary.

In some cases, the application can explicitly describe its requirements to the synchronization subsystem: the precision required, the peers to which synchronization is needed, and so forth. There are also cases where the synchronization subsystem can deduce application requirements implicitly. For example, data flows might imply the scope and lifetime of needed synchronization.

Once the requirements are known, synchronization should *adapt* to them. For example, the number of synchronization packets sent can be varied, trading energy for precision if dictated by the application. This exemplifies a parameterizable or *adaptive fidelity* algorithm [10]. The synchronization system might even choose from a set of synchronization algorithms with differing characteristics depending on the application requirements.

4.5 Exploit Domain Knowledge

Much of the design of the Internet—and, in fact, the Internet Protocol (IP) itself—is meant to put a consistent interface on top of a heterogeneous and inconsistent tapestry of underlying transport media and protocols. NTP shares a similar philosophy: it makes a reasonable set of "lowest common denominator" assumptions about the environment in which it expects to operate. In the Internet, this is the right choice: it has allowed NTP to become deployed nearly ubiquitously, despite the wide variety of processors, oscillators, network media, node topologies, and cross-traffic it encounters.

The disadvantage of such a design is that it precludes the sys-

tem from taking advantage of any special features that might be available. In a WSN, where we are often trying to squeeze every possible resource from the system, it may not be feasible to give up performance for the sake of generality. It often makes sense for each application to take advantage of whatever special features are available at every layer of the system.

For example, the inherent properties of a some communication media can be leveraged for time synchronization. In 802.11 networks, Reference-Broadcast Synchronization (RBS) has been shown to achieve far better precision than NTP by exploiting the fact that it has a physical-layer broadcast channel [9]. In time-division based MAC layers, some form of synchronization already exists between radios, and can often be accessed by a synchronization process on the CPU [28]. Some radio standards such as Bluetooth [34] provide a separate synchronous communication channel with low delay jitter, which can be used for exchanging synchronization pulses.

Time synchronization can also use domain knowledge about the application. For example, Römer's scheme [26] piggybacks round trip time measurements to ordinary data packets sent by other processes. This achieves time synchronization without imposing any additional load on the network. Similarly, RBS can work by observing extant broadcasts in the system instead of sending its own special packets.

5. CONCLUSIONS

Physical time synchronization is a crucial component of wireless sensor networks. In this paper, we described some of the important applications of synchronized time in a WSN and their characteristics along axes such as energy use, scope, precision, lifetime, and cost. We argue that traditional time synchronization schemes like NTP can not be applied in this new domain, where many assumptions have changed. Unlike in wired networks, energy is finite; infrastructure is unavailable; topologies are no longer static or even connected. Based on our experience with the development of time synchronization schemes for WSNs, we proposed some design principles: use multiple, tunable modes of synchronization; do not maintain a global timescale for the entire network; use post-facto synchronization; adapt to the application, and exploit domain knowledge.

Sensor networking is still a young field; none of these principles have yet been proven in the way that NTP has proven itself in the Internet. However, we believe they provide a useful framework to guide the design of WSN time synchronization as the field evolves.

6. REFERENCES

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless Sensor Networks: A Survey. *Computer Networks*, 38(4):393–422, March 2002.

[2] A. Beaufour, M. Leopold, and P. Bonnet. Smart-Tag Based Data Dissemination. Submitted for publication, June 2002.

[3] R.E. Beehler. Time/frequency services of the U.S. National Bureau of Standards and some alternatives for future improvement. *Journal of Electronics and Telecommunications Engineers*, 27:389–402, Jan 1981.

[4] Alberto Cerpa, Jeremy Elson, Deborah Estrin, Lewis Girod, Michael Hamilton, and Jerry Zhao. Habitat monitoring: Application driver for wireless communications technology. In *Proceedings of the 2001 ACM SIGCOMM Workshop on Data Communications in Latin America and the Caribbean*, April 2001. Available at http://www.isi.edu/scadds/papers/CostaRica-oct01-final.ps.

[5] Alberto Cerpa and Deborah Estrin. ASCENT: Adaptive Self-Configuring Sensor Networks Topologies. In *Proceedings of the Twenty First International Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2002)*, June 23–27 2002. http://lecs.cs.ucla.edu/Publications.

[6] Z. D. Chen, HT Kung, and D. Vlah. Ad Hoc Relay Wireless Networks over Moving Vehicles on Highways. In *MobiHoc 2001*, Long Beach, USA, October 2001.

[7] Flaviu Cristian. Probabilistic clock synchronization. *Distributed Computing*, 3:146–158, 1989.

[8] J. Elson and D. Estrin. Time Synchronization for Wireless Sensor Networks. In *2001 International Parallel and Distributed Processing Symposium (IPDPS), Workshop on Parallel and Distributed Computing Issues in Wireless Networks and Mobile Computing*, San Francisco, USA, April 2001.

[9] Jeremy Elson, Lewis Girod, and Deborah Estrin. Fine-Grained Network Time Synchronization using Reference Broadcasts. In *Proceedings of the Fifth Symposium on Operating Systems Design and Implementation (OSDI 2002)*, Boston, MA, December 2002. http://lecs.cs.ucla.edu/Publications.

[10] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar. Next Century Challanges: Scalable Coordination in Sensor Networks. In *MobiCom 99*, Seattle, USA, August 1999.

[11] Saurabh Ganeriwal, Ram Kumar, Sachin Adlakha, and Mani Srivastava. Network-wide time synchronization in sensor networks. Technical report, Networked and Embedded Systems Lab, Elec. Eng. Dept., UCLA, April 2002.

[12] D. Ganesan, B. Krishnamachari, A. Woo, D. Culler, D. Estrin, and S. Wicker. An Empirical Study of Epidemic Algorithms in Large Scale Multihop Wireless Networks. Submitted for publication, February 2002.

[13] N. Glance, D. Snowdon, and J.-L. Meunier. Pollen: using people as a communication medium. *Computer Networks*, 35(4):429–442, 2001.

[14] M. Grossglauser and D. Tse. Mobility Increases the Capacity of Ad-hoc Wireless Networks. In *INFOCOM 2001*, Anchorage, USA, April 2001.

[15] R. Gusell and S. Zatti. The accuracy of clock synchronization achieved by TEMPO in Berkeley UNIX 4.3 BSD. *IEEE Transactions on Software Engineering*, 15:847–853, 1989.

[16] ISO/IEC. IEEE 802.11 Standard. IEEE Standard for Information Technology, ISO/IEC 8802-11:1999(E), 1999.

[17] J.M. Kahn, R.H. Katz, and K.S.J. Pister. Next century challenges: mobile networking for Smart Dust. In *Proceedings of the fifth annual ACM/IEEE Intl. Conf. on Mobile computing and networking*, pages 271–278, 1999.

[18] Elliott D. Kaplan, editor. *Understanding GPS: Principles and Applications*. Artech House, 1996.

[19] L. Lamport. Time, Clocks, and the Ordering of Events in a Distributed System. *Communications of the ACM*, 21(4):558–565, July 1978.

[20] Q. Li and D. Rus. Sending Messages to Mobile Users in Disconnected Ad-Hoc Wireless Networks. In *MobiCom 2000*, Boston, USA, August 2000.

[21] J. Mannermaa, K. Kalliomaki, T. Mansten, and S. Turunen. Timing performance of various GPS receivers. In *Proceedings of the 1999 Joint Meeting of the European Frequency and Time Forum and the IEEE International Frequency Control Symposium*, pages 287–290, April 1999.

[22] F. Mattern. Virtual Time and Global States in Distributed Systems. In *Workshop on Parallel and Distributed Algorithms*, Chateau de Bonas, October 1988.

[23] David L. Mills. Internet Time Synchronization: The Network Time Protocol. In Zhonghua Yang and T. Anthony Marsland, editors, *Global States and Time in Distributed Systems*. IEEE Computer Society Press, 1994.

[24] M. Mock, R. Frings, E. Nett, and S. Trikaliotis. Continuous clock synchronization in wireless real-time applications. In *The 19th IEEE Symposium on Reliable Distributed Systems (SRDS'00)*, pages 125–133, Washington - Brussels - Tokyo, October 2000. IEEE.

[25] G.J. Pottie and W.J. Kaiser. Wireless integrated network sensors. *Communications of the ACM*, 43(5):551–558, 2000.

[26] K. Römer. Time Synchronization in Ad Hoc Networks. In *ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 01)*, Long Beach, CA, October 2001. www.inf.ethz.ch/vs/publ/papers/mobihoc01-time-sync.pdf.

[27] K. Römer. Robust and Energy-Efficient Temporal-Order Event Delivery in Wireless Sensor Networks. Submitted for publication, March 2002.

[28] I. Rubin. Message Delays in FDMA and TDMA Communication Channels. *IEEE Trans. Communin.*, COM27(5):769–777, May 1979.

[29] K. Sohrabi, J. Gao, V. Ailawadhi, and G. Pottie. Protocols for self-organization of a wireless sensor network. *IEEE Personal Communications*, pages 16–27, October 2000.

[30] T. K. Srikanth and Sam Toueg. Optimal clock synchronization. *J-ACM*, 34(3):626–645, July 1987.

[31] Paulo Verissimo, Luis Rodrigues, and Antonio Casimiro. Cesiumspray: a precise and accurate global time service for large-scale systems. Technical Report NAV-TR-97-0001, Universidade de Lisboa, 1997.

[32] John R. Vig. Introduction to Quartz Frequency Standards. Technical Report SLCET-TR-92-1, Army Research Laboratory, Electronics and Power Sources Directorate, October 1992. Available at http://www.ieee-uffc.org/freqcontrol/quartz/vig/vigtoc.htm.

[33] H. Wang, L. Yip, D. Maniezzo, J.C. Chen, R.E. Hudson, J.Elson, and K.Yao. A Wireless Time-Synchronized COTS Sensor Platform Part II–Applications to Beamforming. In *Proceedings of IEEE CAS Workshop on Wireless Communications and Networking*, Pasadena, CA, September 2002. http://lecs.cs.ucla.edu/Publications.

[34] Bluetooth SIG. www.bluetooth.org.

Routing on a Curve *

Badri Nath and Dragoş Niculescu
Department of Computer Science
Rutgers University
Piscataway, NJ 08854
{badri,dnicules}@cs.rutgers.edu

ABSTRACT

Relentless progress in hardware technology and recent advances in sensor technology, and wireless networking have made it feasible to deploy large scale, dense ad-hoc networks. These networks together with sensor technology can be considered as the enablers of emerging models of computing such as embedded computing, ubiquitous computing, or pervasive computing. In this paper, we propose a new paradigm called trajectory based forwarding (or TBF), which is a generalization of source based routing and Cartesian routing. We argue that TBF is an ideal technique for routing in dense ad-hoc networks. Trajectories are a natural namespace for describing route paths when the topology of the network matches the topography of the physical surroundings in which it is deployed which by very definition is embedded computing.

We show how simple trajectories can be used in implementing important networking protocols such as flooding, discovery, and network management. Trajectory routing is very effective in implementing many networking functions in a quick and approximate way, as it needs very few support services. We discuss several research challenges in the design of network protocols that use specific trajectories for forwarding packets.

Categories and Subject Descriptors

C.2.1 [**Network architecture and design**]: Wireless communication; C.2.2 [**Network protocols**]: Routing protocols

Keywords

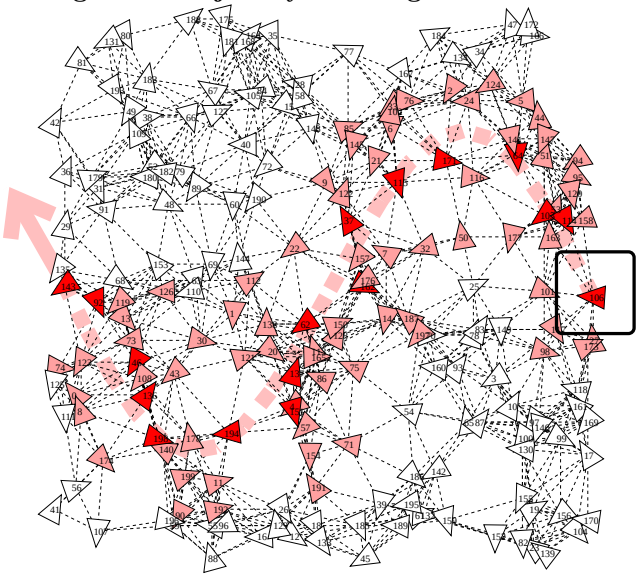
ad hoc networks, trajectory based forwarding, routing

1. INTRODUCTION

Routing packets along a specified curve is a new approach to forwarding packets in large scale dense ad-hoc networks.

*This research work was supported in part by DARPA under contract number N-666001-00-1-8953

Figure 1: Trajectory following a sine curve



The fundamental aspect of considering route paths as continuous functions is the decoupling of the path name from the path itself. The transition from a *discrete view of route path to a continuous view of route path* is important as we move from dealing with sparse networks to dealing with dense networks. The basic idea of routing on curve is to embed the trajectory of the route path in the packet and then let the intermediate nodes forward packets to those nodes that lie more or less on the trajectory. Representing route paths as trajectories is an efficient scalable encoding technique in dense networks. Since a trajectory does not explicitly encode the nodes in the path, it is impervious to changes in specific nodes that make up the topology. We believe that trajectories are a natural namespace to describe route paths when the topology of the network matches the topography of the physical surroundings in which it is deployed which by very definition is embedded computing. Forwarding packets along trajectories can be very effective in implementing many networking functions when standard bootstrapping or configuration services are not available, as will be the case in disposable networks where nodes are thrown or dropped to form a one-time use network.

To effectively forward packets along a trajectory, all that

is needed is a sufficiently dense network and a capability by which nodes can position themselves relative to a coordinate system and estimate distance to its neighbors. The coordinate system could be a relative coordinate system [1], global coordinate system (GPS [2]), or an ad-hoc coordinate system (APS [3]). While an accurate global positioning system such as GPS would be ideal, and some argue that it will be available at a form and cost factor suited for universal deployment, others argue that it is not reasonable to expect GPS to serve all purposes due to non-line of sight and precision requirements. In any case, we believe TBF is an effective mechanism for routing in dense ad-hoc networks. In fact, it can be viewed as a network primitive that serves as a glue between positioning providing services such as GPS, APS[3], or GLS[4] and many applications such as flooding, discovery, routing schemes for path resilience, unicast, multicast and mobile ad hoc routing.

For example, in figure 1 a trajectory representing a sine wave ($x = t$, $y = \sin t$, with the appropriate rotation transformation applied to orient the curve as desired) can be specified in the packet by the source. By using a forwarding technique that delivers the packet to nodes that are close to the trajectory, the packet can be made to traverse the shape of the trajectory specified in the packet as shown in figure . The dashed line is the specified path and the dark shaded nodes are the nodes that forward the packets, while the light shaded ones overhear the packet from the broadcast medium without actually forwarding it. The simulated network is a unit disk graph of 200 randomly positioned nodes.

This technique of forwarding packets along a curve or a trajectory is a generalization of source based routing[5, 6] and Cartesian routing[7, 8]. As in source based routing, trajectories are specified by the source but do not explicitly encode the path on a hop-by-hop basis. As in Cartesian routing, the nodes are selected based on proximity but the proximity is to the trajectory instead of the final destination. TBF combines the best of the two methods: packets follow a trajectory established at the source, but each forwarding node makes a greedy decision to infer the next hop based on local position information, while the overhead of representing the trajectory does not depend on path length. In a network where node positions are known, the packet may be forwarded to the neighbor that is geographically closest to the desired trajectory indicated by the source node. If the destination node is known, the trajectory followed by the packet might be a line, and the method reduces to Cartesian forwarding. A recent work that addresses routing scalability is anchored geodesic packet forwarding [9], where the source specifies a list of anchors between which Cartesian routing is to be used. Other routing solutions in wireless network are complementary to TBF, such as algorithms for guaranteed delivery. The GFG algorithm presented in [10, 11] can be adapted to work with arbitrary trajectories, thus providing guaranteed delivery for routing schemes based on TBF.

Trajectory based forwarding (TBF) has a number of unique features that makes it a candidate as a primitive in dense ad-hoc networks. Here are some distinguishing features of trajectory based forwarding:

- Forwarding based on trajectories decouples the path name from the path itself. Since a trajectory can be specified independent of the nodes that make up the trajectory, routing can be very effective even when the intermediate nodes that make up the path fail, go into doze mode, or move. Reactive routing algorithms (LSR[5], DSR[6]) explicitly encode the path. Proactive route maintenance algorithms (AODV[12]) need to update topology changes. Thus, communication costs increase in proportion to the topology changes induced by mobility. In DSR, any changes to the path results in route discovery, while in AODV, results in propagation of route updates. In TBF, route maintenance is virtually free and unaffected by mobility rate, as the path specification is independent of the names of the nodes involved in forwarding.

- The specification of the trajectory can be independent of the ultimate destination(s) of the packets. In fact, the trajectory specified in the packets need not have a final destination! A packet can be let go along a line or a curve for a desired length. This has implications for implementing networking functions such as flooding, discovery and multicast.

- Trajectories are a natural namespace for specifying paths in embedded systems. Here, packets are expected to take routes that closely mirror the physical infrastructure in which the network is embedded. The topography of the network is a good indication of its topology.

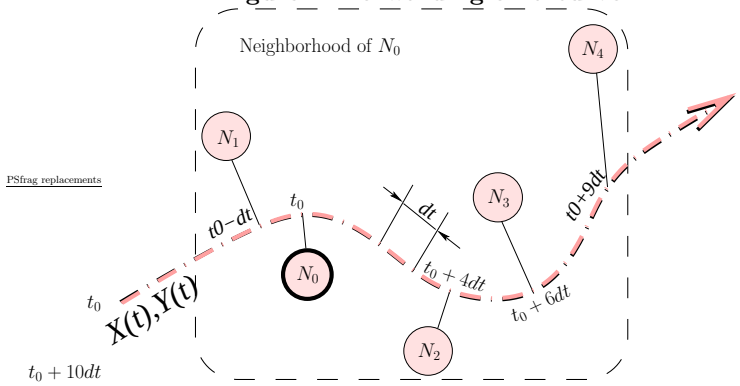
- For packets to follow the trajectory closely, nodes need to have the capability to position themselves in a coordinate system. A system such as GPS or APS would be sufficient for this purpose, but TBF can be made to work even without GPS, as a coordinate system relative to the source can be constructed by positioning only the nodes in the neighborhood of the trajectory. However, the accuracy of the actual path followed will depend upon the errors due to localizations and coordinate transformations.

- Multipath routing has several advantages, such as increased reliability and capacity, but is seldom used because of the increased cost of maintenance. Using TBF, an alternate path is just another trajectory requiring no more maintenance than the main trajectory.

2. FORWARDING ON A TRAJECTORY

The trajectory is usually decided by the source and can be conveniently expressed in parametric form $X(t), Y(t)$. The meaning of parameter t is also decided by the source, as well as the resolution at which the curve will be evaluated, indicated by dt. For the simplicity of the explanation, assume that t indicates the distance along the curve. The neighborhood of a node N_0(figure 2) is defined as the portion of the curve and the nodes that are within a certain distance from N_0, indicated by a dashed rectangle in the figure. In the simplest case, the neighborhood could be the smallest rectangle enclosing all N_0's one hop neighbors. In a network in which node positions are known, the main question is how to choose a next hop that best approximates the trajectory. Assume node N_0 receives a packet with the trajectory indicated by the curve $X(t), Y(t)$ and the value t_0 that corresponds to the point on the curve that is closest to N_0. Using

Figure 2: Forwarding on a curve

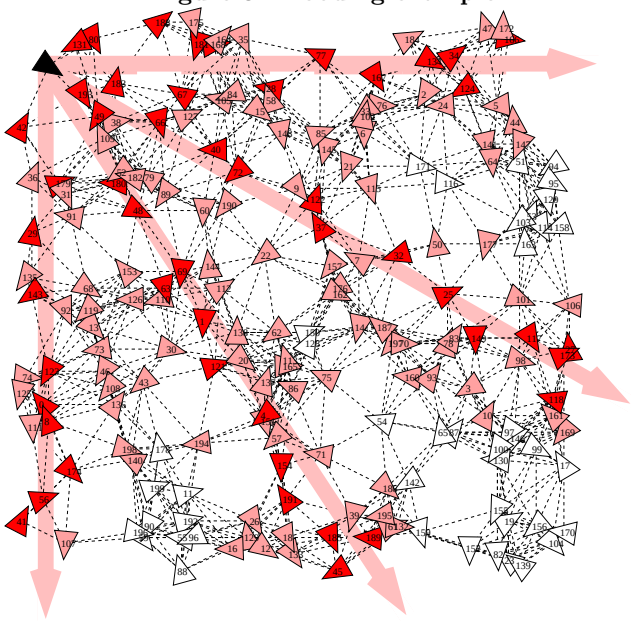


sampling of the curve at dt spaced intervals, indicated by dots in the dashed trajectory curve, N_0 can compute all the points of the curve that reside inside its neighborhood. For all neighbors $N_1..N_4$, their corresponding closest points on the curve are $t_0, t_0 + 4dt$, $t_0 + 6dt$, $t_0 + 9dt$. When referring to curve fitting, these values are called residuals. In fact, the mentioned method computes an estimation of the residuals instead of the true ones, which would require either infinite precision, or usage of higher derivatives of $X(t)$ and $Y(t)$. Since choosing a next hop for the packet should be towards advancement on the trajectory, only portion of the curve with $t > t_0$ is considered. For this reason, node N_1 receives t_0 as the closest point, instead of $t_0 - dt$, which would be closer to the perpendicular from N_1 onto the curve.

Several policies of choosing a next hop are possible:

- node closest to the curve, with the minimum residual. This policy would favor node N_2 and would tend to produce a lower deviation from the ideal trajectory. This strategy should be chosen when it is important for the packets to follow the trajectory closely to possibly determine the state around the trajectory. Since the packet stays close to the trajectory, there is less likelihood for a packet to wander away from the intended trajectory due to errors in localization.
- most advancement on t, choosing N_4. This policy should also be controlled by a threshold of a maximum acceptable residual in order to limit the drifting of the achieved trajectory. It would produce paths with fewer hops than the previous policy, but with higher deviation from the ideal trajectory. This strategy should be favored when delay is an important metric and the packet needs to reach the destination or the perimeter of the network in minimum number of hops.
- centroid of the feasible set, choosing N_3. The path traversed will be more coarse compared to the choice of choosing the node closest to the trajectory and will cover the center of activity of the region without having too many small hops. The centroid is a way to uniformly designate clusters along the trajectory, and a quick way to determine the state of the network along the trajectory.
- randomly choose among the best nodes obtained from above three choices. This is useful when node positions are imperfect, or when it may be necessary to route around unknown obstacles. The randomness in the choice of the next hop can help mitigate the effects of small obstacles.
- in mobile networks a forwarding policy that might provide better results would be to choose the next hop which promises to advance along the trajectory, or one that is expected to have the least mobility in the future. Another interesting choice would be to choose a node that is moving towards the trajectory, rather than the one that is moving away. This strategy is expected to work better when trajectories and neighborhood information are cached, and for a given trajectory, the packet is forwarded to the same node. However, either a periodic evaluation of the position of neighbors relative to the trajectory, or cache invalidation is necessary.

Figure 3: Flooding example



3. BASIC TRAJECTORIES AND APPLICATIONS

In this section we will show how some simple trajectories can be used to implement some basic networking functions such as flooding, discovery, and network management.

Flooding: using TBF, flooding can be replaced with a number of outgoing radial lines that are reasonably close to each other to achieve a similar effect without all the communication overhead involved by receiving duplicates in classical flooding. More generally, a source would indicate the directions and the lengths of the lines that would achieve a satisfactory coverage. The coverage relies on the typical broadcast property of the wireless medium, in which several nodes overhear the packet being forwarded.

In figure 3, the node in the upper left corner broadcasts along

Figure 4: Flooding performance

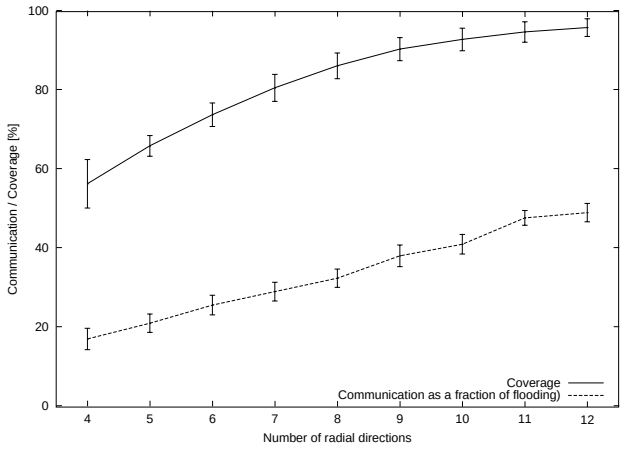
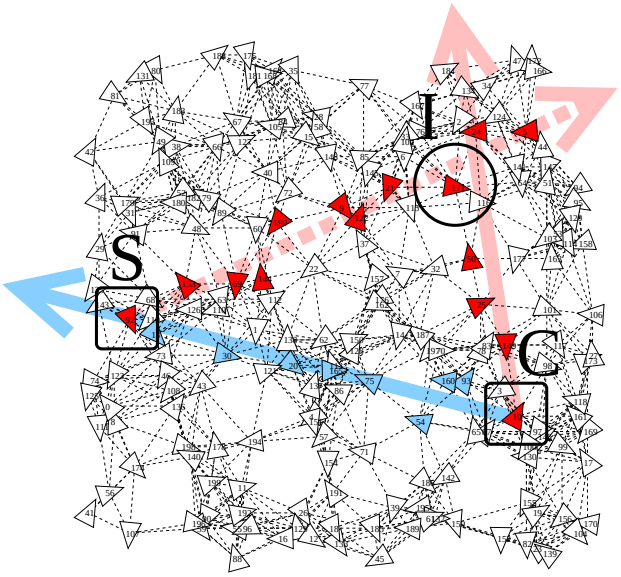


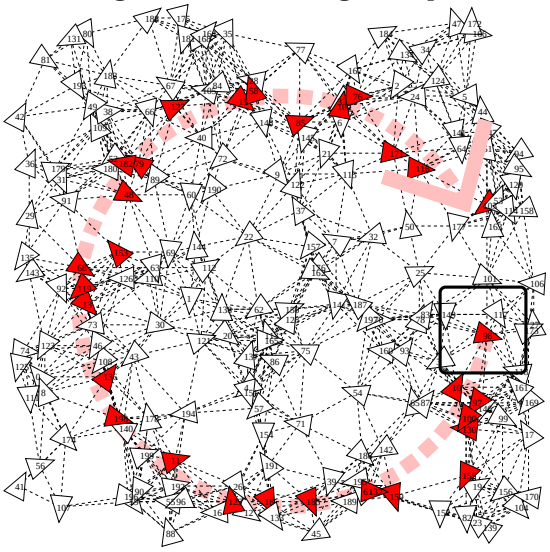
Figure 5: Discovery example



four directions achieving a coverage of 82%. Using another node in the middle of the network(157), and varying the number of spokes, we compare the number of packets used by regular flooding and TBF flooding. In figure 4, we can see that while TBF flooding can achieve almost complete coverage, it only uses half of the communication employed by regular flooding. Thus, flooding along spokes is a quick and dirty way of sending information for purposes of say, configuring sufficient number of nodes to bootstrap the network. Other trajectories such as H-trees or fractals can also be used to provide the required coverage. An interesting research issue is the tradeoff between coverage and the complexity of the trajectory specification and evaluation.

Resource discovery: many algorithms use initial discovery phases based on flooding[6, 13] in order to find a resource or a destination. A replacement scheme using trajectories is as follows(figure 5): servers S advertise their position along arbitrary lines and clients C will replace their flooding phase

Figure 6: Boomerang example



with a query along another arbitrary line which will eventually intersect the server's line. The intersection node I then notifies the client about the position of the destination. The client can then form another trajectory and send requests directly to the server. Discovery and subsequent routing can be done without knowing the position of any of the nodes, as a trajectory can be built in the relative coordinate system of the client. To guarantee that the client and server lines always intersect, it is necessary to send four spokes from both server and client in four cardinal directions (independently and randomly chosen by each). Another necessary condition is that the network be large enough to accommodate the intersection point, and for this, it has to be large enough to fit a circle with diameter CS.

Management: An interesting trajectory is a **boomerang,** where the packet takes a certain route and returns to the sender (figure 6). A circle is a simple example of this trajectory and can be used to secure the perimeter of the network by sending a challenge response token along the trajectory. All nodes who respond properly can be considered to be authenticated. A packet that returns to the source after being routed along a circle or a boomerang path is in effect implementing a self ARQ.

4. RESEARCH CHALLENGES

4.1 Specifying and determining trajectories

There are a number of choices in representing a trajectory: functional, equation, or a parametric representation. Functional representation(e.g. $y(x) = ax + b$) cannot be used to specify all types of curves, such as vertical lines. Equation representation (e.g. $x^2 + y^2 = R^2$) requires explicit solution to determine the points on the curve and cannot easily handle the notion of progress. Parametric representation is ideally suited for the purpose of forwarding. The parameter of the curve is a natural metric to measure the forward progress along the path and can be considered a proxy for the hop count. Given the choice of a parametric form, the next issue is how the trajectories should be interpreted. Trajectories can have several parameters and each node needs

to correctly interpret these parameters. One approach is to have a convention where the nodes know how to interpret the fields given a well known set of trajectories.

A trajectory can also be composed of several simple trajectories. These simple trajectories can be viewed as segments and can be specified along with the appropriate intervals of the parameter over which it is valid. To obtain a continuous trajectory, these intervals need to overlap. A trajectory may involve a complex shape that may not have a simple parametric representation. However, in many cases, this shape can be represented by a simpler set of Fourier components. The more Fourier components in the specification of the trajectory, the better is the reproduction of the trajectory. There is an interesting tradeoff between the accurate reproduction of the trajectory and the overhead of specifying the components and interpreting them. Other flexible and compact ways to encode a complex trajectory are fractal encoding and compiled form encoding. The latter approach, also used in active networking[14], sends in each packet the binary code needed for the parametric evaluation of the curve.

Another important question is how the trajectories are determined. In our initial work on TBF, we assumed that the source knows the trajectories a priori and that is normally derived from the application at hand. This may be the case in many applications of embedded computing where the topology of the network closely mirrors the underlying physical infrastructure in which the network is embedded. As was shown with applications such as flooding and discovery, simple lines or rays are sufficient. However, in other circumstances, the source or the intermediate nodes may have to determine the actual trajectory given the destination or group of destinations. Our initial thinking is to rely on a trajectory mapping service similar to DNS where, given a destination, or a group of destinations, the mapping service returns a trajectory for the source to use. How such a service can be built is a topic for further research.

4.2 Modifying trajectories

Once the source has determined the trajectory to use for routing, it may have to be modified due to network conditions such as obstacles, failures or mobility. The question is how to detect that a modification is needed and who should modify the trajectory. A route failure detection mechanism or an ACK/NACK scheme can be used for detecting the need for a change in the trajectory. A source can then choose a new trajectory for forwarding subsequent packets. Here, the onus of successful delivery is completely on the source. Another choice would be for some authorized intermediate nodes to detect the need for modification and provide a "patch" to the trajectory so that packets can be forwarded around obstacles, failures or local routability conditions such as network congestion. Whatever the reason for providing these patches, the patches should only serve as local detours where the packets eventually get back on track and are forwarded along the original trajectory.

4.3 Implementing network protocols

TBF can be used in implementing many network protocols in ad hoc networks. In static networks, such as sensor networks, packets can be forced by the trajectory to take a certain route around obstacles or congestion. If the destination is mobile and its path is known, it is possible to route towards this path. This means that TBF can naturally use information about trajectories of destinations rather than locations of destinations.

Multipath routing is used for either path resilience[15], or load balancing. Trajectories can easily specify disjoint or braided paths between a source and a destination without the cost of route discovery. Different packets can take different paths, since setting up a route is virtually free.

Routing to multiple destinations has well known uses such as flooding and multicast, but also application specific uses, such as temporary groups which do not justify the cost of setting up as multicast groups. TBF accepts specifying arbitrary trees as trajectories and all nodes along the tree will receive the packet. The tree is the actual physical tree representing the location of recipients. Nodes close to branching points in the tree have the task of duplicating data for the branches downstream. To reduce the overhead of representing the tree in its entirety, the tree representation can be pruned off as the packet is forwarded downstream.

Routing in mobile, ad hoc networks can make use of TBF to forward packets. Since trajectories do not explicitly encode the members of the path, it is less likely to result in a route failure when one more nodes along the path move and there are other nodes close to the path that can forward the packet. Even source mobility is not a problem, as long as the source can determine the new trajectory towards the destination. The important research issue is how to deal with the mobility of the destination. A combination of TBF and location-aided routing[16] may be needed. Another strategy would be to use the knowledge of the trajectory of the mobile node and construct a trajectory that uses a path from the source to a point on the mobile node's trajectory. As long as the destination is moving along the specified trajectory, the packets will reach a mobile destination.

Problems that need further exploration include: who is to determine the trajectory and how (given all destinations and obstacles), how to deal with route failures (for example by patching the trajectory based on local knowledge that is not available globally), and what is the right balance between node centric and position centric addressing.

4.4 Use of positioning techniques

In many networks GPS is not available, but TBF can make use of alternative positioning methods based on heterogeneous capabilities, such as angle measurement, range measurement and compasses. One option is to run a network wide, ad hoc positioning algorithm, such as APS[3]. Another option is to use multimodal sensing to determine local ranges and angles, and eventually to set up local coordinate systems which allow handling of trajectories. This, however, involves successive coordinate systems alignment and is more sensitive to the measurement errors than the first option. For all the mentioned positioning options, the performance of TBF (accuracy, drift) depends on the quality of the positions obtained at each node and therefore, ultimately, on the quality of the measurements.

4.5 Error Management

Errors in forwarding have two main causes: trajectory encoding and measurement error. Errors might be produced by encoding when a complex trajectory is not represented accurately, in order to reduce overhead. Measurement errors affect TBF when multimodal sensing is used for positioning (when GPS is not available). Depending on the hardware available on the nodes, errors accumulate along the trajectory and can be modeled as different random walk processes. If, for example, a compass is available in each node, the actual trajectory might drift left or right of a true linear trajectory, but will not change direction. If only ranges or angles are used for local positioning the trajectory may completely go off course. Characterizing these errors analytically would help in providing confidence areas for points on the achieved trajectory. Another research direction would be to investigate the possibility of correcting the trajectories on course, possibly with the use of a small fraction of GPS enabled nodes.

5. CONCLUSIONS

In this paper, we have outlined TBF – a new paradigm of forwarding in large scale, dense ad-hoc networks. Trajectory based forwarding has a number of features that are ideal for use in embedded networks, sensor networks, disposable networks, and networks that support the paradigm of ubiquitous or pervasive computing. We have presented some techniques for implementing trajectory-based forwarding and have shown the benefits of simple trajectories in implementing networking operations such as flooding, discovery and network management. It is our belief that this approach opens up a new area of research and can draw from the results of research efforts in various other disciplines. We have provided a basis for discussion of a number of research issues that needs to be addressed in the area of networking in an embedded world.

6. ACKNOWLEDGMENTS

We would like to thank members of DATAMAN lab for all their constructive comments.

7. REFERENCES

[1] M. Hamdi S. Capkun and J.P. Hubaux. Gps-free positioning in mobile ad-hoc networks. In *Hawaii International Conference On System Sciences*. HICSS-34, January 3-6 2001. Outrigger Wailea Resort.

[2] B. Parkinson et al. *Global Positioning System: Theory and Application*. Progress in Astronautics and Aeronautics, 1996.

[3] Dragoş Niculescu and Badri Nath. Ad hoc positioning system (APS). In *GLOBECOM*, November 2001. San Antonio.

[4] J. Li et. al. A scalable location service for geographic ad hoc routing. In *6th ACM MOBICOM*, August 2000. Boston, MA.

[5] D. B. Johnson. Mobile host internetworking using ip loose source routing. Technical Report CMU-CS-93-128, Carnegie Mellon University, February 1993.

[6] David B. Johnson and David A. Maltz. Dynamic source routing in ad hoc wireless networks. *Mobile Computing*, 353, 1996.

[7] G. Finn. Routing and addressing problems in large metropolitan-scale internetworks. Technical Report ISI Research Report ISI/RR-87-180, University of Southern California, March 1987.

[8] J. C. Navas and Tomasz Imielinski. Geographic addressing and routing. In *MobiCom'97*, September 26-30 1997. Budapest, Hungary.

[9] Ljubica Blazevic, Silvia Giordano, and Jean-Yves Le Boudec. Self organized terminode routing. In *Cluster Computing*, volume 5, pages 205–218, 2002.

[10] P. Bose, P. Morin, I. Stojmenovic, and J. Urrutia. Routing with guaranteed delivery in ad hoc wireless networks. In *3rd International Workshop on Discrete Algorithms and methods for mobile computing and communications*, August 1999. Seattle, WA.

[11] B. Karp and H.T. Kung. Gpsr: Greedy perimeter stateless routing for wireless networks. In *6th ACM MOBICOM*, August 2000. Boston, MA.

[12] Charles E. Perkins and Elizabeth M. Royer. Ad hoc on-demand distance vector routing. In *2nd IEEE Workshop on Mobile Computing Systems and Applications*, pages 90–100, February 1999. New Orleans, LA.

[13] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: a scalable and robust communication paradigm for sensor networks. In *6th ACM MOBICOM*, August 2000. Boston, MA.

[14] D. L. Tennenhouse and D. Wetherall. Towards an active network architecture. *Multimedia Computing and Networking*, January 1996. San Jose, CA.

[15] Deepak Ganesan, Ramesh Govindan, Scott Shenker, and Deborah Estrin. Highly resilient, energy efficient multipath routing in wireless sensor networks. In *Mobile Computing and Communications Review (MC2R)*, volume 1, 2002.

[16] Y.-B. Ko and N. H. Vaidya. Location-aided routing (lar) in mobile ad hoc networks. In *MobiCom'98*, October 1998.

SIGCOMM Award Nominations

The SIGCOMM Award was initiated in 1989 as a means of honoring computer communication professionals for outstanding lifetime technical achievement in the fields of data and computer communications. The award consists of a plaque and a $2,000 honorarium. The award is presented at the annual SIGCOMM Conference, at which time the awardee is invited to deliver a technical address. In the following are guidelines for submitting a nomination.

(1) Self-nominations are not accepted.

(2) The nominee need not be a member of ACM SIGCOMM.

(3) The nominator must be a member of ACM SIGCOMM.

(4) Nominations must be received by the Chair of the SIGCOMM Award Committee no later than March 31st each year.

(5) Nominations that do not result in an award may be resubmitted/updated in subsequent years.

(6) Previous awardees are not eligible for future nominations.

(7) Members of the Award Committee are not eligible.

(8) Members of the SIGCOMM Executive Committee (Chairman, Vice Chairman, Secretary-Treasurer, Past Chairman, and Editor) are not eligible.

Material to be included in the nomination:

(1) Curriculum Vitae, including publications, of nominee.

(2) Concise statement (one sentence) of the work for which the award is being nominated. This statement will appear on the award plaque.

(3) Description of the nominee's role in the work justifying the nomination.

(4) Letters of recommendation from others discussing the rationale for the nomination and by what means the recommender knows of the nominee's work. It is recommended that at least three letters of recommendation be included in the nomination materials.

(5) Justification for declaring the nominee's work to be a major, lifetime contribution to computer communications.

The nomination should be made in the form of a letter addressed to the Chair of the SIGCOMM Award Committee. The nominator should solicit recommendations from colleagues in the field who are most familiar with the nominee's achievements. It is not recommended to send copies of published papers or books along with the nomination materials. The nominator is responsible for gathering all nomination materials and sending two copies of all materials to reach the Chair of the Award Committee before March 31st. Submissions can be made by email (preferred) or paper (two copies). No late nominations will be considered for the current year. They will be held over until the following year for consideration.

The Chair of the SIGCOMM Award Committee is:

Karen R. Sollins
MIT Lab for Computer Science
200 Technology Square
Cambridge, MA 02139
Voice: +1 617 253 6006
Fax: +1 617 253 2673
Email: sollins@lcs.mit.edu