COMPUTER COMMUNICATION REVIEW

A Publication of ACM SIGCOMM

Volume 33, Number 5 October, 2003
ISSN #: 0146-4833

Contents

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.

SIGCOMM Treasurer's Report August, 2003

Dear SIGCOMM Colleagues,

Some of you may have noticed that my name started appearing in the inside of the front cover of CCR as Secretary/Treasurer in 2002, a position I assumed at the request of the most recent ex-officers (Partridge/Zhang/Delp). Although this term ended recently, I would like to take this opportunity to report to you on the state of SIG finances.

The past 18 months have seen a series of significant changes in the economy, many of which have affected our SIG as well. The challenges of this period, together with a conservative approach for the future, have resulted in some recent fiscal changes which are worth reviewing. Although the SIG experienced significant unexpected expenses in FY2002, recent budget changes bring us back to a more sustainable plan by FY04.

Budget Issues:

In the July 2001 - June 2002 fiscal year (FY2002), deficit spending was over double what was planned. In FY03 the budget was modified to significantly reduce organization costs (nearly halved) and anticipated a reduction in conference expense coupled with an increase in conference revenue. The FY03 organization costs were kept close to budget, but additional conference losses, largely due to Sigcomm 02 but balanced by workshop income, resulted in a loss $12K larger than expected. FY04 increases in dues help turn the organization deficit into a gain, and more conservative conference budgets should help eliminate the conference deficit. Overall, FY03 actual and FY04 planned represent steps in a fiscally viable direction. A summary of our budgets appears below:

	BUDGET FY 2004	**Actual FY 2003**	**Budget FY 2003**	**Actual FY 2002**	**Actual FY 2001**
	7/03-6/04	7/02-6/03	7/02-6/03	7/01-6/02	7/00-6/01
Organization					
Revenue	$111,628	$78,882	$75,060	$106,364	$119,362
Expense	$89,188	$122,746	$119,871	$210,189	$201,240
Net	$22,440	-$43,864	-$44,811	-$103,825	-$81,878
Conferences					
Revenue	$276,936	$275,630	$268,022	$228,165	$302,175
Expense	$272,111	$287,855	$267,178	$307,141	$257,308
Net	$4,824	-$12,225	$844	-$78,976	$44,867
Organization and Conferences					
Revenue	$388,564	$354,512	$343,082	$334,528	$421,538
Expense	$361,299	$410,600	$387,049	$517,329	$458,549
Net	$27,264	-$56,089	-$43,967	-$182,801	-$37,011
Fund Balances					
Beginning	$217,867	$273,956	$273,956	$456,756	$493,767
Ending	$245,131	$217,867	$229,989	$273,956	$456,756

These past two fiscal years were impacted by a number of challenges:

- One-time support of a National Academy of Science workshop to investigate the impact of 9/11.
- Seeding a number of new workshops (HotNets, Latin American, Internet Measurement), each permitted to operate at a small loss until established.
- A policy of distributing paper proceedings of these new seeded workshops to the entire SIG membership (a new cost exceeding 75% of registration fees).
- Conference/workshop losses, where the Latin American workshop came in at 50% over the expected 'seed' cost, and Sigcomm 02 conference lost $37K – which was somewhat offset by revenues of the workshops. We were also impacted by the ratio of student to full member registrations, since student registrations are typically heavily subsidized.
- SIG taxing structure changes. In FY2003, ACM instituted an expense-based overhead tax, so we were taxed on what we spend, not on what we pull in as revenue (as was previously the case). This affects surplus spending, as that money was taxed when earned, and is now taxed again when spent. The SIG also lost expected interest revenue on its surplus, as a result of economic changes.

Our FY04 budget takes these issues into account, and is much more conservative, being based on a number of budget guidelines which have been recently recommended.

Dues Increases:

SIGCOMM has increased its dues for FY04 to:

ACM member	$25
ACM student member	$15
non-ACM member	$35

This increase reflects increases in the overall operating costs of the SIG, notably to cover increased publication costs. This is the first real increase for regular members since 1990, and restores both student and non-member rates to their 1990 levels.

New SIG Budget Guidelines:

In order to continue to hold a number of conferences and workshops, ACM requires each SIG to hold a surplus related to a fraction of planned meeting budgets. Maintaining the SIG surplus is critical to the SIG continuing to sponsor these meetings.

In previous years this surplus was sufficiently large that we had the opportunity to utilize portions to support seeding new workshops and conferences, and supporting unique events for which corporate support would be difficult. Due to the change of the economic

climate, including the challenges of maintaining corporate support for our larger meetings (e.g., the Sigcomm Conference), a number of policies have been instituted to stabilize the SIG surplus as follows:

- *Balance unplanned costs by restructuring discretionary items.*
  Unforseen events are (by definition) difficult to anticipate, but, where possible, support for such events will be balanced against planned investments for the coming year.

- *Workshop proceedings only to workshop participants and by direct purchase.*
  Although the distribution of proceedings of new workshops may significantly increase their impact, it is not feasible to distribute proceedings to all members out of either SIG revenue or surplus at this time. Proceedings will be distributed where the workshop raises supporting funds, typically via corporate sponsorship.

- *Student fees representative of per-person costs.*
  The SIG continues to encourage student participation in workshops and conferences. Again, due to the current economy, student fees are expected to increase substantially to eliminate the gap between per-person revenue and per-person costs.

- *Conferences budget for surplus.*
  Conferences are now expected to budget a modest surplus, both to reduce the liability to the surplus as well as to (hopefully) restock it.

- *Limit new 'seed' workshops.*
  The number of new 'seed' workshops per year has been reduced to 1; seedling workshops are expected to break-even by their third year. IMW-II and HotNets-I both showed profits in 2002. The Latin American workshop for Oct. 2003 is being monitored closely to avoid the overruns of 2001.

The above policies are augmented by codifying some of the more informal policies and procedures of conference budgeting, to streamline the budgeting process, unify policies across SIGCOMM sponsored meetings, and reduce exposure to unforseen costs.

I hope this review has been helpful, and although my term has recently ended, I will continue to work with Martina Zitterbart, our new Secretary/Treasurer, as well as the entire SIGCOMM Executive Committee, to ensure the financial stability of the SIG well into the future.

I look forward to your comments and feedback.

Joseph D. Touch
Outgoing Secretary/Treasurer
ACM SIGCOMM
touch@isi.edu

DCAP: Detecting Misbehaving Flows via Collaborative Aggregate Policing

Chen-Nee Chuah
ECE Department
University of California
Davis, CA 95616, USA
chuah@ece.ucdavis.edu

Lakshminarayanan Subramanian
EECS Department
University of California
Berkeley, CA 94720, USA
lakme@cs.berkeley.edu

Randy H. Katz
EECS Department
University of California
Berkeley, CA 94720, USA
randy@cs.berkeley.edu

ABSTRACT

This paper proposes a detection mechanism called *DCAP* for a network provider to monitor incoming traffic and identify misbehaving flows without having to keep per-flow accounting at any of its routers. Misbehaving flows refer to flows that exceed their stipulated bandwidth limit. Through collaborative aggregate policing at both ingress and egress nodes, DCAP is able to quickly narrow the search to a candidate group that contains the misbehaving flows, and eventually identify the individual culprits. In comparison to per-flow policing, the amount of state maintained at an edge router is reduced from $O(n)$ to $O(\sqrt{n})$, where n is the number of admitted flows. Simulation results show that DCAP can successfully detect a majority (64-83%) of the misbehaving flows with almost zero false alarms. Packet losses suffered by innocent flows due to undetected misbehaving activity are insignificant (0.02-0.9%). We also successfully build a prototype that demonstrates how DCAP can be deployed with minimal processing overhead in a soft-QoS architecture.

Keywords

Misbehaving flow detection, Traffic policing, Flow-level accounting

1. INTRODUCTION

Considerable research has focused on extending the Internet architecture beyond best-effort to provide different classes of services to different applications depending on their Quality of Service (QoS) requirements. Existing proposals range from per-flow mechanisms such as IntServ [11] and RSVP [8] to per-class mechanisms such as Diff-Serv [7] and Clearing Houses [10].

Two integral components of any QoS architecture are: admission control and traffic policing [11]. These two components, in combination with appropriate QoS scheduling policies, enable a network provider to dynamically allocate its shared resources to various customers and satisfy their QoS requirements. While admission control limits the number of flows in the system to avoid depletion of resources, traffic policing is responsible for ensuring that the admitted flows use only their allocated share of network resources.

In this paper, we focus on the traffic policing component and address the question of how a network provider can effectively detect *misbehaving* flows with minimal overhead. We define a flow to be *misbehaving* if it generates traffic in excess to its allocated share. It is crucial to detect and penalize misbehaving flows because they can potentially starve other flows sharing the same physical resources, resulting in degraded performance for legitimate flows. We make two simple assumptions: First, the only resource under contention between the flows is the network bandwidth. Second, we use a predictive service model where a source specifies its bandwidth requirement (during admission control) based on its average rate.

The traditional approach to traffic policing in the context of IntServ, Diff-Serv and ATM networks, is to monitor every admitted flow at the routers [9, 15, 35]. Per-flow policing may incur significant processing overhead at the routers ($O(n)$ where n is the number of flows) resulting in poor scalability. To partially alleviate the problem, the policing can be completely shifted to the edge of an ISP's network.

In this paper, we present an alternative to per-flow policing which trades off detection accuracy for increased scalability. Detection accuracy refers to the probability of detecting misbehaving flows at a router. We propose an aggregate policing mechanism, called DCAP (Detection via Collaborative Aggregate Policing), that has both a good misbehaving flow detection probability and a reduced state and overhead at the routers. DCAP works well under the assumption that the number of misbehaving flows is small compared to the total number of flows in the system. We thereby discount the state required for containing misbehaving flows after they are detected from our analysis.

1.1 Paper Contributions and Overview

In this paper, we propose a detection system, Detection via Collaborative Aggregate Policing (DCAP), that allows a network provider[1] to continuously monitor admitted traffic and detect misbehaving flows in an efficient and scalable manner. The design of DCAP is

[1] A network provider refers to a backbone or regional Internet Service Provider (ISP) that administers its own network domain and provides Internet access to individual and corporate customers or smaller service providers.

driven by three goals: (a) to protect the well-behaved flows against resource depletion due to misbehaving flows, (b) to identify and eventually penalize the misbehaving flows, and (c) to achieve robustness with respect to noise conditions and errors in workload modeling.

Our main contribution is to show how one can leverage distributed, aggregate policing at edge routers to quickly identify a *group* that contains the misbehaving flows. The problem can then be simplified into measuring only the flows within this "candidate" group. The principal observation is that it is relatively harmless to have delay in detecting a misbehaving flow when the overall load is relatively low due to statistical multiplexing. This motivates our approach to aggregate multiple flows for group policing, which eliminates per-flow state management at edge routers. We propose an explicit Flow-Identifier (Fid) assignment scheme to group the admitted flows and police the aggregate group. The fact that each edge router maintains only the aggregate state for each *group* is crucial for the reduction of state from $O(n)$, which would be required if each flow were policed individually, to $O(\sqrt{n})$, where n is the number of admitted flows. Aggregate policing also reduces the per-packet processing overhead. In addition, aggregate policing is more robust to noise conditions and variations of a flow's rate. This is because, the aggregate rate of flows in a group has a much lesser variance (as a fraction of the net aggregate rate) in comparison to the average variance of the rate of a flow (as a fraction of the flow's rate) in the group.

We analyze the performance of DCAP through simulations and characterize the trade-offs between different performance criteria by tuning various parameters of DCAP. Results show that DCAP can detect a majority of misbehaving flows with close to zero false-alarms and low detection time for a variety of source models. We also demonstrate the practicality of our scheme by prototyping our algorithm using a Click router [25].

The rest of the paper is organized as follows. In Section 2, we formulate the tracking of misbehaving flow as an on-line change detection problem, in which one needs to detect the occurrence of misbehaving behavior as soon as possible, but with a low rate of false alarms. We discuss related work in Section 2.2. Section 3 describes the Fid assignment, aggregate policing and misbehaving flow detection scheme within DCAP in details. In Section 4, we illustrate how DCAP can be applied to three different scenarios: (a) within a single ISP, (b) in an overlay network, and (c) across multiple ISPs. We explain our evaluation methodology and present our simulations results in Section 5. Section 6 illustrates the operation of DCAP through a proof-of-concept prototype built on top of a Click router [25]. The prototype demonstrates that the processing overhead introduced by DCAP at an edge router is insignificant. Section 7 summarizes key results and addresses several deployment issues.

2. MISBEHAVING FLOW DETECTION

During admission control, a service contract is negotiated between the service provider and the user. The service contract describes the type and amount of traffic sent by the user, and the expected performance offered by the network provider. A key component required to enforce service contracts is a mechanism to detect misbehaving flows that fail to comply with the allocated rate specified in their service contracts. In this section, we will formulate the misbehaving flow detection problem and discuss previous approaches to this problem.

2.1 Problem Formulation

Misbehaving flow detection (MFD) is an example of on-line change detection problems [6], in which one needs to detect the occurrence of abnormal traffic behavior as soon as possible, with a set of constraints, e.g., without exceeding the tolerable number of false alarms. A *false alarm* happens when a flow is detected as misbehaving when it is not. Let n be the number of incoming flows $f_i(s,d)$ between a specific ingress-egress pair (s,d), each with an allocated rate of $A_i(s,d)$, and m be the number of non-complying flows, where $m \leq n$. The problem is to correctly identify as many of these m flows as possible, i.e., to maximize the probability of *successful detection*. The three intuitive performance metrics for evaluating an MFD scheme are:

1. probability of successful detection, $P_{\rm sd}$ (i.e., a misbehaving flow is not detected),
2. probability of false alarms, $P_{\rm fa}$ (i.e., a flow is detected as misbehaving when it is not), and
3. time to detect, $t_{\rm delay}$

Since the main goal of service contract enforcement is to deliver end-to-end QoS assurance to all admitted flows, the most important criteria is to protect the well-behaved flows that use legitimate amount of resources against misbehaving behavior. We quantify how well the performance of well-behaved flows is preserved in terms of $P_{\rm mis}$, the probability of dropped packets from the complying flows. Identifying the misbehaving flow itself is secondary, as long as the impact from undetected flows' activities on other well-behaved flows, i.e., $P_{\rm mis}$, is negligible. When the overall load is relatively low, misbehaving flows can be harmless even if they are not identified. On the other hand, false alarms may seriously degrade the performance of good flows. Therefore, we can tolerate low $P_{\rm sd}$ but $P_{\rm fa}$ should be close to zero.

In summary, an ideal MFD algorithm with the goal of enforcing service contract should achieve the following (in the order of importance):

1. minimum $P_{\rm fa}$,
2. minimum $P_{\rm mis}$,
3. maximum possible successful detection probability, $P_{\rm sd}$, and
4. small $t_{\rm delay}$,

2.2 Previous Approaches

We make the distinction between two different types of traffic policing:

- *profile-based:* The traffic profile of all the flows being policed is known to the router implementing the policing mechanism.
- *profile-less:* The router has no knowledge of the traffic characteristics of the individual flows.

Our problem falls under the domain of profile-based policing since we have prior knowledge of the allocated rates of the individual flows. We will briefly describe the associated related work for both types of traffic policing.

2.2.1 Profile-based policing

The *traffic profile* of a flow describes the characteristics of the traffic generated by the flow (e.g. peak rate, average rate etc.). Any

profile-based policing mechanism is associated with the goal of identifying flows that violate their specified traffic profiles. profile-based policing is normally used in the context of QoS architecture to detect malicious flows (flows that transmit more than their reserved rate as specified in their traffic profile). The most common example of *profile-based* policing is Token Bucket Filter (TBF) [9], a per-flow policing algorithm which ensures that every flow adheres to its traffic profile. Per-flow policing has been used in the context of IntServ [31], DiffServ [15, 35] and ATM [4, 29, 9] networks. While per-flow policing provides the most accurate accounting information, the required state complexity maintained at each edge router grows linearly with the number of flows, n. Maintaining a counter to measure the traffic sent by millions of concurrent flows in the future will be too expensive (using SRAM) or slow (using DRAM) [14]. Current state of the art policing tools like Cisco's Netflow [1] periodically sample packets and may be too slow, inaccurate and memory intensive. Hence per-flow state maintenance may hinder scalability in future high-bandwidth networks.

Machiraju et al. [26] have suggested an alternative approach where the traffic profile is maintained in the packets rather than the routers themselves. This approach uses the Dynamic Packet State(DPS) concept proposed in [33]. Along with the profile, the service provider also provides a certificate during the admission control phase that acts as a proof to ensure that the end-host does not cheat by modifying its traffic profile. Though this approach removes the necessity of maintaining per-flow state at the routers, it does require a modification of the behavior of end-hosts and the packet header format.

2.2.2 Profile-less policing

Profile-less policing attempts to achieve fairness amongst flows traversing a router and distributes the available bandwidth fairly amongst the individual flows. The most common examples of profile-less policing are the fair queuing algorithms. While many of these algorithms like Weighted Fair Queuing (WFQ) [12, 27, 28] maintain per-flow state at the individual routers to enforce fairness, active queue management techniques like Stochastic Fair Blue (SFB) [17] achieve approximate fairness using minimal per-flow state. SFB is derived from a queue management technique BLUE [18] proposed as an alternative to the RED queuing discipline for early detection of congestion. SFB provides a scalable way to identify and rate-limit non-responsive flows using BLUE, which marks or drops packets based on loss rate and link utilization history.

Estan et al. [14] use a SFB-like approach to identify and provide accurate traffic statistics for the heavy-hitters, i.e., elephant flows that contribute to most the majority of the traffic on a specific router interface. Though we would classify this work under profile-less policing, this technique could be useful for detecting malicious flows that are also heavy-hitters and consume a lot of extra bandwidth.

3. OUR APPROACH: DCAP

In this section, we will describe our solution, Detection via Collaborative Aggregate Policing (DCAP), for the misbehaving flow detection problem.

To reduce the processing and state complexity of per-flow policing, we propose to aggregate flows for group policing. In this process, we do sacrifice a certain level of accuracy in tracking flows' behavior. Aggregate policing allows us to detect a *group* that contains the misbehaving flows. The problem can then be simplified into measuring only the flows within this smaller "candidate" group. However, we still need to preserve the uniqueness of each flow to be able to identify a misbehaving flow eventually. To achieve this, each admitted flow is assigned a flow-identifier, Fid, which is then inserted in the packet header. We assume an application proxy on the client side will be configured to insert the proper Fids before forwarding the packets to the edge router.

A few questions remain to be addressed:

1. How are flows assigned into groups?
2. Can an Fid be devised to represent both the flow itself and the groups it is in?
3. How does one combine various aggregate policers to effectively identify misbehaving flows?

In the following two sections, we describe the mechanisms of our scheme, DCAP, that address theses issues.

3.1 Fid Assignment

There are several possible ways to identify flows, and assign them into different bins for aggregate policing. One simple way is to assign each flow with a random identifier, Fid, which does not require a central database to keep track of which Fids have been assigned. This approach comes with two disadvantages. First, potential Fid collisions (if the number of unique Fids is small) will make it impossible to identify unique flow. Secondly, there is a lack of control and flexibility over how flows are aggregated together. Ideally, we would like to assign flows with similar bandwidth requirements and characteristics into the same group for policing.

3.1.1 Fid Assignment and Releasing in DCAP

We introduce the notion of a *Resource Manager* that explicitly assigns $Fids$ to flows. The Resource Manager (RM) is a logical unit that can be physically placed at the fault monitoring point or policy server in an ISP[2]. The RM maintains the repository of admitted flows, their traffic profiles and service contracts for accounting purposes. Only aggregate states of the reservations are maintained at the edge routers (to both reduce the state and the overhead of policing).

We make two observations. First, though we maintain per-flow state at the RM, this state is accessed infrequently for verification purposes. Hence, the communication between the edge routers and the RM is relatively infrequent. Second, the RM, a single logical unit, in practice is implemented in a distributed fashion across different POPs in an ISP (refer to Section 3.3.1 for more details). Therefore the RM is not a single point of failure.

Every Fid consists of two 16-bit sub-fields: $FidIn$ and $FidEg$. All flows that enter or exit at a particular ER are aggregated into different groups based on their $Fids$. The $FidIn$ and $FidEg$ subfields of a flow identify the groups that the flow belongs to at its ingress and egress ER, respectively. In other words, each of the subfields ($FidIn$ and $FidEg$) serves as group identifiers, while together they form a Fid that uniquely identifies a single flow. Before an Fid is allocated to a flow, the RM has to check whether the Fid has been previously assigned to avoid accidental re-use of the same Fid.

[2]The Resource Manager (RM) should be located very close to the collection of edge routers within a given point of presence(POP). The RM can be implemented as in a distributed manner such that every POP is associated with a *local RM* (Section 3.3.1).

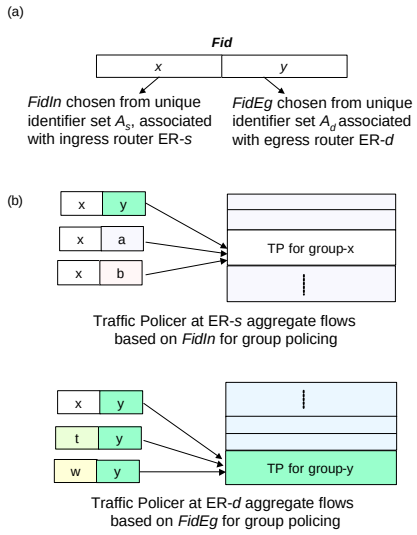

Figure 1: Illustrations: (a) Fid assignments, and (b) aggregate policing at ingress and egress routers.

Each edge router ER_i a set of M unique group identifiers, denoted as $\mathcal{A}_i = \{x_{i1}, x_{i2}, \ldots, x_{iM}\}$, where each member is a 16-bit binary number and is unique across the set $\mathcal{A}_i$. The sets $\mathcal{A}_i$ and $\mathcal{A}_j$ associated with any two ER_i and ER_j are mutually disjoint.

Any Fid of an admitted flow should satisfy the following properties:

1. If the flow is routed from ER_s to ER_d, then $FidIn \in \mathcal{A}_s$, and $FidEg \in \mathcal{A}_d$.
2. No other flow should have the same Fid.
3. Flows with the same $FidIn$ (or $FidEg$) have similar bandwidth requirements.

When a new flow from ER_s to ER_d is admitted, the RM picks a random x_s from $\mathcal{A}_s$ and a random y_d from $\mathcal{A}_d$ such that the Fid with $FidIn = x_s$ and $FidEg = y_d$ satisfies the above properties. This Fid is assigned to the flow, and a new entry with this Fid and its allocated bandwidth is added to the Fid-Repository (FR). The total number of flows that can be uniquely identified in this scheme is $K \cdot M^2$ for a particular ingress ER, where K is the total number of potential egress ERs, each having its own unique set of identifiers. We assume the admitted flow will send a TEARDOWN message to the ingress ER when it terminates. The TEARDOWN message contains the Fid, and its allocated bandwidth. Upon receiving the TEARDOWN message, the RM updates the FR accordingly and releases the Fid.

In the example shown in Figure 1, a new flow that arrives at ingress router ER_s is assigned an Fid with the first subfield, $FidIn$ equals to x, and $FidEg$ equals y. At ER_s, the new flow is aggregated with other flows with the same subfield, $FidIn = x$, for group policing. At ER_d, this flow is grouped with other flows with $FidEg = y$ for policing. Every flow will be policed at both its ingress and egress ER in two distinct groups, thereby increasing the chances of detecting misbehaving flows. Every ER maintains only the aggregate state for each group and hence does not store any per-flow state.

3.1.2 Explicitly Assigned vs User-Selected Fids

In our approach, we attempt to maximize the level of flow aggregation without compromising the uniqueness of $Fids$, thereby minimizing the number of groups to be policed at every edge router(ER). Explicit assignment has two distinct advantages: First, the amount of aggregate policing needed to be performed at the router can drastically be reduced. For example, if there are n flows from an ingress ER to a specific egress ER, an optimal assignment of explicit $Fids$ can be achieved by maintaining only $\sqrt{n}$ groups at each of the two routers. We discuss this optimal assignment in the next section. Secondly, flows with similar bandwidth requirements can be aggregated into a common group to increase the effectiveness of group policing. It reduces the chances of a small bandwidth misbehaving flow hiding in an aggregate containing some large bandwidth flows. This is because it becomes hard to distinguish minor variations of a large bandwidth flow from large-scale bandwidth violations of a small bandwidth flow.

On the other hand, if flows were allowed to assume their own $Fids$, then it would be necessary to maintain a membership function to map the random $Fids$ to a particular group. The cost of performing an online grouping of flows based on these functions would be very high because the $Fids$ are continuously changing. Techniques like Stochastic Fair Blue (SFB) [18] that use random hash functions cannot be applied because they do not provide an inverse mapping from group identifiers to actual $Fids$. Thus, using SFB alone does not provide a direct mechanism to verify whether a suspected flow is truly misbehaving.

3.1.3 Other Challenges

Routing Changes: When a routing change occurs and causes an active flow to change its ingress or egress points, the previously assigned Fid may not match the group identifiers in one of the new ERs or both. Whenever this happens, we can either

1. remark the packets of this flow as best-effort, or
2. contact the RM for re-admission of this flow with the new endpoints.

Further investigation is needed to understand the performance and security issues of both approaches.

Meeting Fid Demands: The typical number of simultaneous flows observed on an ingress link is between 300 and 5000 [19]. It is expected that 5-10% of these will be latency sensitive applications and need Fids. Even if the demand for Fids increases 10 times in future, we need at most $M = \sqrt{5000}$, which is roughly 71 unique identifiers per $\mathcal{A}$ set. Based on the discussion in [32], the total number of ERs within an ISP, K, is in the order of 500.[3] Therefore, the total number of unique identifiers required for the whole ISP ($M \times K$) is roughly 35,500 in this case. Allocating 16-bits ($2^{16} = 65536$) for each subfield should be more than sufficient for producing mutually disjoint $\mathcal{A}_i$ for all routers.

Security Concerns with identifiers: One important security concern with our Fid assignment is the notion of *bogus identifiers*.

[3] K=number of POPs $\times$ number of ERs/pop. Major Tier-1 ISPs has about 25 POPs in the continental USA and each POP consists of a few core routers and 10-20 gateway routers.

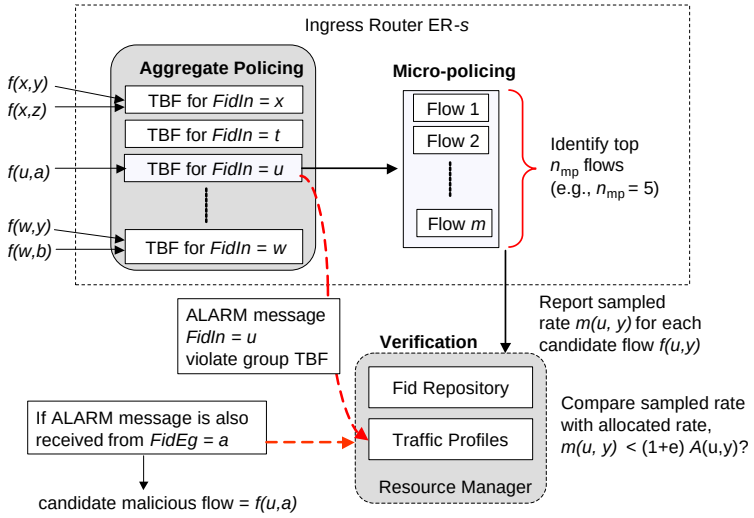

Figure 2: Control blocks of DCAP.

An external misbehaving user/application may attempt to create its own *bogus* Fid which is valid but has not been explicitly assigned by the RM and generate a flow with such an identifier. One possible way of dealing with bogus identifiers is to use the notion of one-way trapdoor functions. A function $f()$ is said to be a one-way trapdoor function if it is one-one and easy to compute while $f^{-1}()$ should be very hard to compute unless given a *trapdoor* [20]. The classic example of a one-way trapdoor function is public-key cryptography. Let us assume that the RM and the ERs within one ISP know a one-way function $f()$ with its corresponding trapdoor, t (not known to the external world). If a flow is supposed to be allocated an identifier, $Fid = FidIn, FidEg$, the actual identifier explicitly assigned to it will be $f(Fid)$ as opposed to just Fid. Upon receiving a packet with a flow identifier, x, the edge router first computes $f^{-1}(x)$ to find the corresponding ingress/egress group identifiers. Since one-way trapdoor functions are very hard to break, this can prevent the problem of bogus identifiers. To avoid replay attacks (using previously specified identifiers), we can just continuously modify the group identifiers at all edge routers. However, there are two basic problems with this approach: (a) computing f^{-1} may sometimes be an expensive operation especially if public-keys are used. (b) 32-bits in a flow identifier may be insufficient to perform these operations. A detailed analysis of security concerns of our approach is outside the scope of this paper.

3.2 DCAP: Detection via Collaborative Aggregate Policing

DCAP is mainly designed for detecting a small number of misbehaving flows among a large group of admitted flows. There are three stages in the detection process:

- Perform aggregate policing at both ingress and egress routers to quickly identify a subgroup that contains the misbehaving flows,
- Guess candidate flows within the group that violate aggregate bandwidth allocation via sampling within a specified time window, and
- Verify whether these flows are truly misbehaving.

Figure 2 highlights the major control blocks of DCAP. The remaining of this section discusses the technical details of each detection phase and example pseudo-codes are included in Figure 3.

3.2.1 *Aggregate Policing*

DCAP deploys traffic policer (TPs) at both ingress and egress ERs for policing the traffic from admitted flows with the matching group identifiers ($FidIn$ or $FidEg$). Each TP is a collection of continuous-state Token Bucket Filters (TBF) [9, 36, 30]). A TBF consists of a counter, which is incremented by the size of each arriving packet, and decrements periodically by a specific rate, as long as the value of the counter is positive. Every group identifier, $x \in \mathcal{A}_i$, is associated with a TBF with two parameters, $r_{\rm tot}$ and $b_{\rm tot}$, where $r_{\rm tot}$ is the total average rate of admitted flows and $b_{\rm tot}$ is the total burst size. When a new flow between ER_s and ER_d with allocated bandwidth $A_{\rm new}$ is admitted and assigned an Fid, the RM updates the TPs at both ingress and egress routers. The total acceptable rate $r_{\rm tot}$ for the TBF with the matching $FidIn$ and $FidEg$ (at ER_s and ER_d, respectively) is increased by $A_{\rm new}$. Packets that violate the associated traffic profile are discarded. Each TP keeps track of the dropped packets and reports the statistics to the RM.

Since the policing at the TP is performed on a group of flows sharing the same 16-bit subfield of Fid, the amount of state information maintained at the ingress ER is proportional to M, the number of unique identifiers in the set, $\mathcal{A}$. Consider an example ISP domain with K edge routers and M=100. Each ER maintains only 100 pieces of state information, but an arbitrary router can admit up to $K \times 10,000$ flows with unique Fids. A per-flow policing scheme would have require each ER to maintain all $K \times 10,000$ states.

3.2.2 *Providing Best Guesses*

```
// pseudo-code excuted by ingress router
PROCEDURE ADC(RECV)
//called when a RECV message arrives
    (A, s ,d) = read(RECV)
    // get the allocated rate, A, ingress and egress point, s and d
    if (measured load < bottleneck link capacity):
        (FidIn, FidEg) = get_Fid()
        // get unique Fid from RM
        r_tot(FidIn) = r_tot(FidIn) + A
        //update TBF w/ FidIn with allocated rate, A
        update(d, A)
        //contact egress router d, to update TBF w/ FidEg

// pseudo-code executed by ingress and egress router
PROCEDURE MONITOR()
    for each k in TBF_LIST:
        if tot_arrival[k] > r_tot[k]:
            send_ALARM(identifier[k])
            MICRO(k)
            //if TBF k overflows, send ALARM to RM
            //enter micro-policing mode

PROCEDURE MICRO(k)
    for each flow in TBF[k]:
        m[flow] =  sample_rate
    list = find_top5_flows()
    //list[flow]=(Fid of flow, measured rate of flow)
    send_RM(list)

// pseudo-code executed by the Resource Manager
PROCEDURE VERIFY(list)
    for each flow in list:
        if m_flow > (1 + e) A_flow:
            penalize(flow)
```

Figure 3: Pseudo-codes for DCAP mechanisms.

As an example, let a flow with $Fid = [f, g]$ be misbehaving. All flows with the same $FidIn = f$ will be policed as an aggregate in the same Token Bucket at the ingress ER, TP_s, regardless of what their $FidEg$ is. If the total allocated rate of $FidIn = f$ is violated, the affected TP reports f to the RM using an ALARM message (Figure 2). However, this information alone is insufficient for pinpointing the exact misbehaving flow, because there can be as many as $K \cdot M$ flows with the same $FidIn$ that could potentially be misbehaving. If the TP at an egress ER $FidEg = g$ also sends an ALARM message, the RM guesses that a flow with Fid that contains both f and g as its subfield is misbehaving, and submits this Fid for verification.

However, a misbehaving flow may not always be detected at both its ingress and egress ERs. To improve the effectiveness of DCAP, we introduce a "micro-sampling" mode. Whenever a group TBF that violates the aggregate rate is identified, DCAP requires the edge routers to sample individual flows within this group for a duration of t_{mp}. At the end of this period, the associated ER identifies n_{mp} largest flows, and report their Fids, along with the sampled rate, to the RM. Normally the potentially misbehaving flows are the ones that transmit at a much higher rate relative to other members.

In this paper, we consider t_{mp}= 5 seconds and n_{mp}= 5 for performance evaluation. With the assumption that the number of misbehaving flows is relatively small, n_{mp}=5 is reasonable, as shown in the ns simulations (Section 5). However, fixing a value of n_{mp} does have its disadvantage. It limits the efficacy of the micro-policing in two scenarios: (a) if the number of large flows is greater than n_{mp}, and (b) if many small misbehaving flows are hiding in the aggregate after discounting the n_{mp} large flows. An alternative solution is to report all the flows that are "disproportionally" larger than the rest in the group. We can leverage the fact that flows with *similar bandwidth* are aggregated for policing in the same group to compute the relative "size" of the flows as the following. Let r_{tot} be the allocated bandwidth to a traffic aggregate that has n admitted flows. We estimate the bandwidth requirement of individual flows as r_{tot}/n. All the flows with the sampled rate exceeding r_{tot}/n by a large margin (say 5%) are potentially misbehaving and will be reported to RM.

3.2.3 Verifying Misbehaving Behavior

For each reported flow with $Fid = [x, y]$, the RM compares the allocated rate, $A(x, y)$ with the measured rate $m(x, y)$ reported in the ALARM message:

$$m(x, y) < (1 + \epsilon) \cdot A(x, y) \quad (1)$$

where ϵ is a hysteresis parameter to absorb transient behavior of bursty traffic. If the condition in (1) is violated, the flow is considered misbehaving. ϵ is typically between 0 and 0.05. A counter associated with this flow is incremented for every such violation of condition (1). To reduce the probability of false alarm, we introduce a second hysteresis parameter, η, which determines the minimum number of violations before a flow is reported as misbehaving. A reasonable range for η is between one and five.

3.2.4 Hiding in the Aggregate

Although group policing allows our architecture to scale, it limits the effectiveness of DCAP because a misbehaving flow can "hide" in the aggregate. This can happen when:

- the aggregate usage of the group is less than the total allocated rate because certain flows are under-utilizing their resources, and
- The misbehaving flow is relatively small compared to other larger, yet legitimate, flows in a misbehaving group.

To address this problem, we introduce redundancy by deploying a traffic policer at every egress point as well. By assigning a unique Fid to every flow, we ensure that no two flows are in the same group in both the associated ingress and egress ERs. Essentially every flow is policed in the aggregate at two distinct points to maximize the number of misbehaving flows that are detected. Secondly, we assign flows with similar bandwidth requirement into the same group (Property 3 of Fids in Section 3.1.1).

3.3 Other Issues

3.3.1 Implementation of the Resource Manager

Although the Resource Manager (RM) is described as a single logical entity within a domain, it can be implemented as a distributed architecture across POPs. Every POP of an ISP usually has a fault monitoring facility to continuously manage the link and router status in its network. The additional functions of the RM can be implemented as additional pieces of software that reside in these monitoring facilities. For example, a *local-RM* of a POP can maintain part of the domain's database, by tracking $Fids$ where at least one of the $FidIn$ or $FidEg$ is a valid group identifier of an edge router within the same POP. For every flow that is admitted, a new entry with its Fid and allocated bandwidth is created in the partial FR databases at both its ingress and egress POPs. Similarly, when a flow stops, we remove the flow's entry from the local RMs in its ingress and egress POPs, respectively.

An alternative model for building a Resource Manager would be to just distribute the state amongst the edge routers as opposed to aggregating it at specific monitoring points within a POP. This alternative model is mainly applicable outside the realm of ISPs where we cannot assume the presence of any POP structure. (refer to section 4.2, for the applicability of this model and DCAP for overlay networks)

While the alternative model may no longer have the state reduction property of DCAP (i.e. the nodes will maintain per-flow state), the nodes still need to perform only aggregate policing. This is because, the *active state* at a node (state used for policing flows on a per-packet basis) is still the aggregate state while the per-flow state is accessed less frequently.

3.3.2 Penalizing Misbehaving Flows

Once a misbehaving flow is detected, several penalty actions can be taken against it: e.g., dropping all the packets of this flow, demoting all its packets to best-effort, or charging more for the connection. Such a penalty would require keeping some state information at the edge router, but only for a very small subset of flows that misbehave (assuming the number of misbehaving flows is small). Our framework can support any of these penalty actions, but for simplicity, we choose packet dropping as the default. We will not address the impact of different penalty actions in detail, since the main focus of this paper is the design of a scalable detection system to identify and isolate these malicious flows. We also do not consider the additional state incurred for containing the malicious flows after they are identified.

4. APPLICABILITY OF DCAP

In the previous section, we described our solution to the malicious flow detection within one ISP's network. In this section, we will describe different scenarios under which our solution can be applied. In particular, we will consider 3 specific scenarios: (a) Single ISP scenario, (b) Policing in Overlay Networks, (c) End-to-end flows traversing multiple ISPs.

Until now, we have been vague about the definition of a flow. The notion of a *flow* in our work can be more generic than just an end-to-end connection between two end-hosts. DCAP places three constraints on the definition of a flow:

1. Every flow is associated with a unique flow-identifier as specified by an ISP (refer to Section 3.1 for more details). At a router, all packets with the same flow-identifier belong to one flow.
2. All packets of a flow are associated with the same ingress and egress routers within the ISP's network.
3. If multiple end-to-end connections have the same QoS requirements and share the same intra-domain path between a specific pair of ingress and egress routers, they can be treated as a single flow if the connections originate from the same *customer* that is accountable for billing purposes (from an ISP's perspective).

4.1 Single ISP Scenario

We will now use the generality in the definition of a flow to show how DCAP can be used for policing within an ISP. Service Level Agreements(SLAs) offered by one ISP to another is normally a complicated agreement, which is carefully worded to address different aspects of performance(latency, bandwidth, loss) at a very coarse granularity. A sample SLA document of a popular ISP is available online [3]. These SLAs are normally static and negotiated on a monthly/yearly basis rather than on very small time granularities.

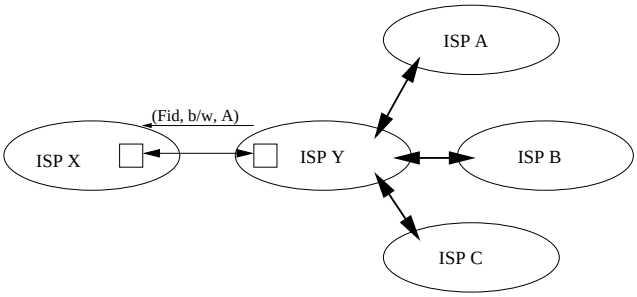

Figure 4: Supporting Dynamic SLAs between ISPs

An ISP can offer to provide transit service between two of its neighboring ISPs. In the simplest case, the transit ISP statically provisions a certain amount of bandwidth between its two neighbors. DCAP provides a scalable way of extending this notion of SLAs to a dynamic reservation model within one ISP without making much modifications to the existing Internet architecture. Figure 4 illustrates a simple scenario where ISP Y has four neighbors A,B,C,X. ISP Y can dynamically provision a certain amount of bandwidth b from ISP X to ISP A and present the edge router of ISP X with an Fid and a corresponding bandwidth b. ISP X can potentially insert this Fid to any packet destined for ISP A (which can be directly determined from the BGP tables in ISP X[4]). Similarly, ISP Y can potentially offer many such $Fids$ of varying bandwidths dynamically to ISP X for each of its other neighbors A,B,C.

DCAP provides a scalable policing model to ensure that the neighboring ISPs do not violate their SLA agreements. However one may think that such a system can be deployed by simply using a per-flow or a single aggregate policing model. We argue that DCAP provides two distinct advantages over both cases in this scenario. First, certain tier-1 ISPs have around $N = 500 - 2000$ ISPs as neighbors. In the worst case scenario, if each pair of these ISPs ($N \times N - 1$) setup an SLA through their common neighbor, per-flow policing may not be a scalable option. Second, between a pair of neighboring ISPs(say X and A), it may be beneficial for ISP Y to handle multiple $Fids$ of varying bandwidths rather than a single Fid of a certain aggregate rate. This offers ISP X the additional flexibility of partitioning these $Fids$ offered by Y amongst its different customers.

4.2 Overlay QoS Provider Scenario

OverQoS [34] and Service Overlay Networks [13] are two recent proposals to provide some form of end-to-end QoS guarantees using an overlay network. These architectures enable a third-party to set up an overlay network and offer customers coarse-level bandwidth guarantees. Traffic policing is also an integral component of these architectures.

The fundamental reason why DCAP would be appropriate for such overlay architectures stems from the assumption that the entire over-

[4]If Y peers with A at multiple locations, the traffic from X to A needs to be diverted along a single egress point of Y to preserve the semantics of a flow.

lay is owned by a single administrator. To deploy DCAP, we assume that every overlay node could be a potential egress point and the Fid allocation is based on the ingress and egress points (entry and exit points within the overlay) of a QoS flow. Here, a flow could either represent a single end-to-end connection or an aggregate pipe for a customer. However, we make one assumption: The ingress and egress overlay node remain the same during the course of a flow. This assumption definitely holds for at least the ingress node since a typical deployment strategy for many overlay-based solutions is to enable the first overlay node as the default gateway of an end-host.

An overlay network may not have specific aggregation points to provide the functionality of a resource manager for a collection of nodes. Hence, this functionality needs to be distributed across all overlay nodes (as described in Section 3.3.1).

4.3 QoS across Multiple ISPs

In order to use DCAP as the traffic policing building block for providing QoS across multiple ISPs, we need to make a few assumptions. First, we require an explicit RSVP-like admission control phase to signal an individual flow's performance requirements to the ingress nodes of all ISPs. The ISPs along the path can locally decide to admit or reject the flow based on the availability of its network resources. Second, an admitted flow is supposed to carry a stack of identifiers (one Fid per ISP along the path). Once a packet of a flow exits an ISP's boundary, the egress router should strip off the corresponding Fid from the top of the stack. Third, when a connection ends, the flow sends its ingress router an explicit message (TEARDOWN) to release the resources.

These assumptions also pose as challenges towards deploying DCAP in a multiple ISP scenario to provide end-to-end QoS. IntServ [11], having been proposed almost a decade ago is far from being deployed. The presence of multiple ISPs raises a fundamental limitation towards deploying end-to-end QoS.

An alternative aspect currently under investigation is whether SLAs between ISPs could be composed together for providing some meaning form of end-to-end QoS guarantees. In Section 4.1, we presented a model of a dynamic SLA that one ISP acts as a transit provider and offers a pipe abstraction between two of its neighboring ISPs. Consider a simple scenario of 4 ISPs, A,B,C and D in a routing path. Assume that dynamic SLAs between A-B-C (i.e. B offering a transit SLA between A and C) and B-C-D with identifiers f_1 and f_2 are already in place. We can compose these identifiers at the transit point between B and C to offer a pipe abstraction between A and D. If such a model can be extended end-to-end, we can potentially offer coarse bandwidth guarantees between various pairs of stub networks. If such a pipe abstraction between stub networks is associated with multiple $Fids$ the stub networks is responsible for allocating these $Fids$ to the individual flows.

5. PERFORMANCE EVALUATION

In this Section, we evaluate the performance and robustness of DCAP via simulation study. We first describe our methodology in Section 5.2, including details on network topology and traffic generation in our simulations. In Section 5.3, we list the four test scenarios, which are designed to address specific issues. The results and detailed discussions of each case are presented in Section 5.4-5.5.

5.1 Simulation Model: Assumptions

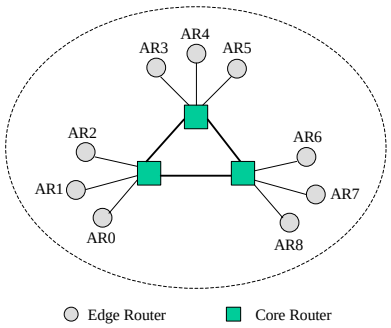

Figure 5: Simulation Topology

To distinguish between QoS flows and non-QoS flows, we introduce two classes of flows: *high-priority* and *best-effort*. We assume that high-priority flows require statistical QoS guarantees while best-effort traffic does not receive any performance assurance. Core routers employ priority queuing to distinguish QoS flows from non-QoS flows. We use a per-flow reservation protocol like RSVP [8] to signal individual flow's performance requirements to the ingress nodes, which locally admit or reject new flows based on the availability of the network resources.

There are typically two types of admission control mechanisms: *parameter-based* or *measurement-based*. Measurement-based admission control (MBAC) algorithms base their decisions on measurements of existing traffic while parameter-based techniques admit flows based on worst-case bounds of different quality metrics (delay, bandwidth, loss rate). Therefore, MBACs are best suited for providing soft real-time service without hard guarantees. Many MBAC algorithms and principles outlined in [22, 23, 21, 24] can be applied in combination with DCAP. For our simulation study and lab prototype reported in this paper, we use the Measured Sum (MS) algorithm: a new flow arriving at edge router ER_s and destined for edge router ER_d with bandwidth requirement A_i is admitted if:

$$A_i + L_{\text{tot}}(s,d) \leq R(s,d)$$

where $L_{\text{tot}}(s,d)$ is the total estimated traffic and $R(s,d)$ is the total capacity reserved for carrying high priority traffic between any pair of ingress-egress routers, (s,d). $R(s,d)$ could be statically configured by an ISP based on traffic predictions or dynamically allocated based on aggregate reservation requests. We assume the former case in this paper.

5.2 Simulation Setup

We use the ns-network simulator to implement the basic mechanisms of DCAP. The TP[5] is implemented as a connector in front of a node, and a time-window estimator is introduced at each input link to estimate the rate of existing flows. The admission control module is created as an NsObject and inserted before the ingress ERs. The various tasks of the RM in our architecture are implemented at the Tcl-level. Our DCAP-patch works for ns-2.1b6.

Since it is infeasible to run large-scale Internet simulations over ac-

[5]We modify the DiffServ module contributed by Sean Murphy, http://www.teltec.dcu.ie/ murphys/ns-work/diffserv/index.html.

tual networks, we use ns to simulate a simple subgraph of an Internet topology shown in Figure 5 that consists of a set of core routers (CRs) and egress routers (ERs). The CRs are fully meshed, while the ERs form stub networks connected to individual CRs. Flows from host networks enter and exit the network domain through edge routers, AR0-AR8, where they are aggregated for policing using the DCAP scheme. All routers support priority scheduling and there is enough buffering for 200 packets at each queue. All control signaling between the RM (not shown on the figure) and the ERs is carried in UDP messages.

We are unable to access real traffic traces from Internet backbone networks because such data is proprietary. Instead, we derive traffic models based on published results and data sets collected by ISPs themselves.

The arrival process of the admission-controlled traffic is modeled as Poisson with arrival denoted as $\lambda_i(t)$. We use the index *t* to indicate the time-of-day dependence of the traffic demand as reported in [16] and [19]. For example, the bandwidth consumption typically peaks between 10 a.m. and 2 p.m. during the day and shows a dip from midnight to 3-4 a.m. To reflect the realistic traffic demand, we introduce $\pm$ 10-15% changes to $\lambda_i(t)$ at a regular interval of 30 minutes. The traffic distributions from an ingress ER to a set of egress ERs are based on a random probabilistic model.

We use four kinds of traffic source models in our experiments:

1. We use EXP1 to model a typical Voice-over-IP source. EXP1 has exponential on and off times with an average of 1.004 s and 1.587 s, respectively. This corresponds to a 38.53% talk-spurt cycle, as recommended by ITU-T specification for conversational speech [5]. The peak transmission rate is 64 kbps, and the average is 24.8 kbps.
2. EXP2 also has exponential on and off times, but with an average of 100 ms and 900 ms, respectively. The peak rate is increased to 248 kbps while keeping the average rate the same as EXP1, leading to a burstier source.
3. CBR is a constant bit rate source of 64 kbps.
4. PARETO source has Pareto on and off times but the average rate is the same as EXP1.

EXP1, EXP2 and CBR have exponential lifetimes with an average of 300s. The flow lifetimes of PARETO sources follow a lognormal distribution with average of 300 s. The aggregation of Pareto sources is known to exhibit long-range dependencies [37]. For all four cases, packets are 320 bytes in length.

In our simulations, we assume that each new flow will request for bandwidth r that is equal to its average rate, regardless of its source model. The network does not have a priori knowledge on the peak rate or characteristics of the flows.

5.3 Test Scenarios

As discussed in Section 2.1, there are different performance criteria in evaluating DCAP as a misbehaving flow detection mechanism. In particular, we are interested in two events: successful detection and false alarms. The probability of successful detection $P_{\rm sd}$ is approximated as the fraction of misbehaving flows that are actually detected. Similarly, the probability of false alarm $P_{\rm fa}$ is the fraction of normal flows that are incorrectly reported as misbehaving. Since the flows are policed as an aggregate, misbehaving flows can cause packets from complying flows to be dropped. The probability of a packet being incorrectly dropped, denoted as $P_{\rm mis}$, quantifies the impact of misbehaving flows on end-to-end performance seen by other member flows.

We study the trade-offs between these different metrics by tuning the parameters of DCAP. Following the discussion in Section 2.1, it is more desirable to sometimes miss the detection of a misbehaving flow (hence lower $P_{\rm sd}$) than to wrongly penalize the good flows and hurt their performance. Therefore, the objective of DCAP is to maximize $P_{\rm sd}$, while keeping $P_{\rm fa}$ and $P_{\rm mis}$ to near zero.

To examine the robustness of DCAP, we simulate four extreme cases:

Case 1: False Positive Detection
This is a control experiment with zero misbehaving flows to determine the optimum choice of DCAP parameters to reduce the number of false alarms.

Case 2: Homogeneous Flows:
This arrangement is similar to Case 1, but now a small fraction, γ, of the flows are misbehaving. All the flows generate traffic with the same average rate and have similar characteristics.

Case 3: Mixing Elephants and Mice
Flows generally do not have similar bandwidth requirements in a real network. Previous studies show that 20% of the flows (known as elephants) contribute to 80% of the Internet traffic. It is important to understand how DCAP copes with such scenario.

In this test scenario, we assign Fids such that one large flow (the elephant) and many simultaneous small flows (mice) are grouped together for policing. The allocated rate of the elephant flow $A_{\rm l}$ is 10 times larger than the allocated rate of a small flow, $A_{\rm s}$. All of the small flows are compliant, and only the large flow misbehaves.

Case 4: Hiding in the Aggregate
Again, we consider a mixture of one large flow and many simultaneous flows like Case 3. However, the large flow is compliant this time, and a fraction γ of the small flows are misbehaving. This is the case in which the small misbehaving flows may survive the group policing without being detected by hiding in the group aggregate (Section 3.2.4) if we only perform regular sampling.

We repeat each experiment using four different source models: EXP1, EXP2, CBR and PARETO. For each scenario, the simulation was repeated 10 times with different random seeds, and the average $P_{\rm sd}$, $P_{\rm fa}$, and $P_{\rm mis}$ was computed. Each simulation ran for 1000s. All experiments were performed under high load with 20% blocking probability. A misbehaving source requests allocation for r kbps but sends traffic at a higher rate, randomly chosen between $1.1 \cdot r$ and $1.2 \cdot r$ kbps (10-20% violation). The average and peak rate for each source model is the same as described in Section 5.2.

5.4 Results and Discussions

Case 1: False Positive Detection

The experiments in Case 1 are intended for understanding the limitations of the DCAP and its performance sensitivity with respect to different choices of design parameters. Ideally, none of the flows should be reported as "misbehaving", but the transient behavior of

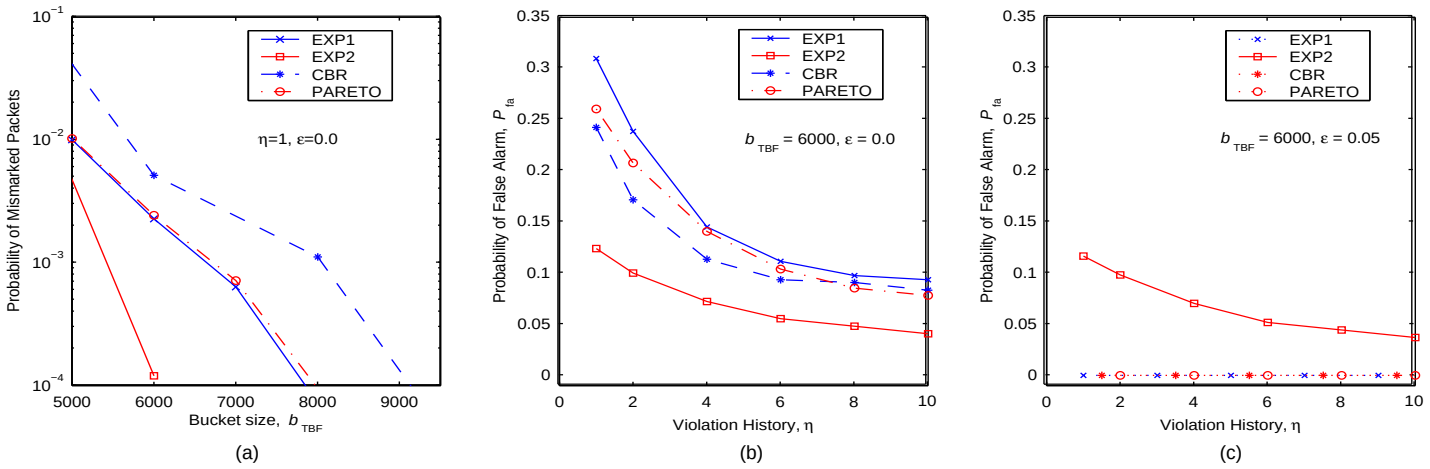

Figure 6: Case 1: Zero Misbehaving Flows.

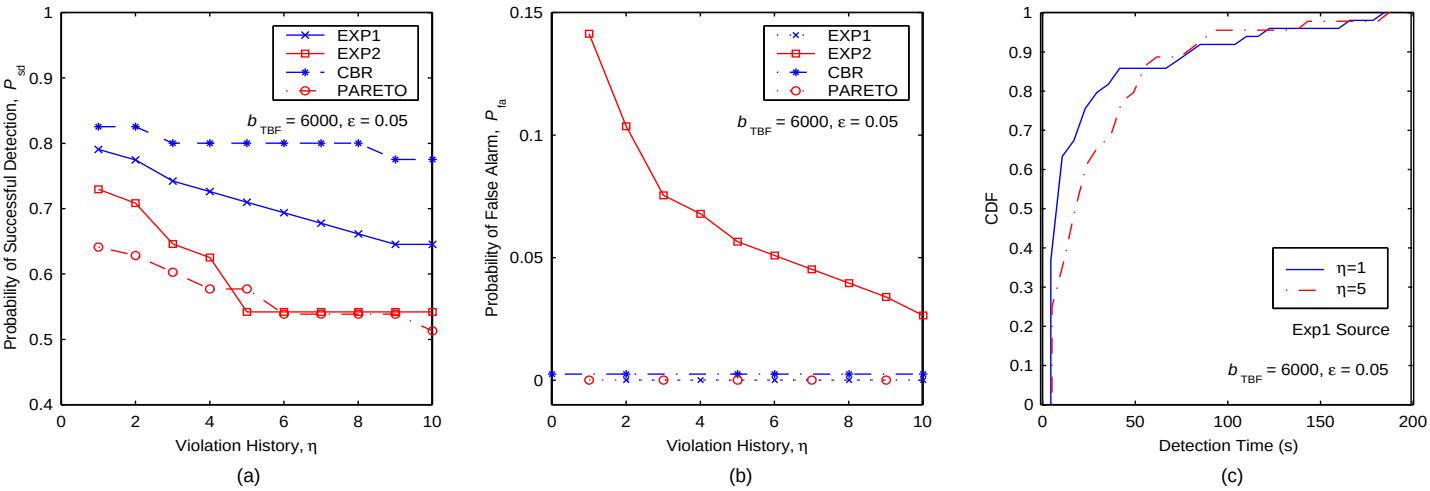

Figure 7: Case 2: Many small homogeneous flows; a small fraction, γ =0.1, misbehave.

bursty traffic could momentarily overflow the Token Bucket Filters (TBFs) and be interpreted as misbehaving, leading to a "false alarm". Figure 6a shows how the choice of bucket size, $b_{\rm TBF}$ at the Traffic Policers (TP) affects the probability of mis-marked packets, $P_{\rm mis}$. A smaller value of $b_{\rm TBF}$ is more effective in detecting misbehaving flows, but there should be enough tokens to allow the legitimate packets to pass, and keep the $P_{\rm mis}$ low. Except for the CBR traffic, $P_{\rm mis}$ is below 0.01 for other source models. When the $b_{\rm TBF}$ is set to 6000 using a fixed leaky rate, all the flows (including CBR traffic) can be policed with losses $P_{\rm mis}$ less than 0.005. The two hysteresis parameters η and ϵ determine under what conditions a flow is reported as "misbehaving" (Section 3.2.3), but have no effect on $P_{\rm mis}$.

We can relax the condition for DCAP by increasing ϵ and η, and this helps to reduce the number of false alarms. Figure 6b and 6c study how $P_{\rm fa}$ varies as a function of η for ϵ = 0.0 and 0.05. For a 0% tolerance level in DCAP (i.e., ϵ=0), $P_{\rm fa}$ decreases gradually as η is increased. However, we notice that $P_{\rm fa}$ drastically decreases for all the source models when the tolerance level is increased to 5%. This indicates that $P_{\rm fa}$ is more sensitive to ϵ than η. For the rest of the experiments, we choose ϵ=0.05 and $b_{\rm TBF}$ = 6000.

Case 2: Homogeneous Flows Scenario

Increasing η causes a delay in reporting misbehaving flows and may adversely impact the effectiveness of DCAP. We examine this issue in Case 2. We set the value of γ (fraction of misbehaving flows) to be 0.1. Figure 7a and 7b show the variation of $P_{\rm sd}$ and $P_{\rm fa}$ as η is increased from 1 to 10. The effect of η on $P_{\rm sd}$ for the CBR source is minimal. For the other source models, $P_{\rm sd}$ decreases sharply when η is increased and the rate of decrease varies across the source models. From Figure 7b, we can infer that only in the case of the EXP2 source model is $P_{\rm fa}$ sensitive to the value of η. With η = 1, we can detect most of the misbehaving flows with EXP1 (79%), CBR (83%) and PARETO (64%) sources with virtually zero $P_{\rm fa}$. In the case of EXP2, there is a trade-off between maximizing $P_{\rm sd}$ and minimizing $P_{\rm fa}$ as we choose the value for η. This indicates that burstier sources are more difficult to detect. The observed $P_{\rm mis}$ is between 0.02-0.79%.

We also measured the detection time for each correctly identified misbehaving flow and plotted the distributions in Figure 7c. With

Table 1: Case 3: One large misbehaving flow and many small complying flows. η = 5, b_{TBF} = 6000, ϵ = 0.05.

Source	EXP1	EXP2	CBR	PARETO
P_{sd}	1.0	1.0	1.0	1.0
P_{fa}	0.0077	0.027	0.012	0.0013
P_{mis}	0.003	0.00021	0.0072	0.0037

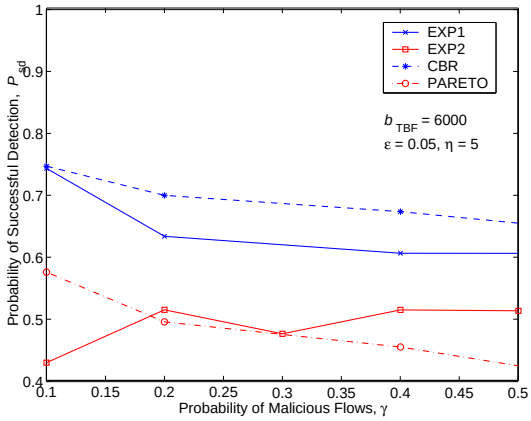

Figure 8: Case 4: One large flow (A_{l}) and many small flows (A_{s}); γ of small flows misbehave.

η = 1, the average detection time is 26.9 seconds, which is less than 1/10 of the average duration of a flow. 90% of the flows are detected within 78.9 seconds. When we increase η to 5, the average detection time increases to 33.8 seconds, which is still reasonably fast. The 90^{th}-percentile detection time is 80.4 seconds in this case.

The simulations in Case 2 show the basic results of DCAP hold across different source models. Although long range dependent traffic like PARETO is harder to detect, we can achieve a reasonable success rate (0.64) with zero false alarms. The presence of burstier sources, EXP2, pose challenges to the DCAP scheme, and we need to choose the value for η carefully to maximize P_{sd} while keeping P_{fa} reasonably small.

Case 3: Mixing Elephant and Mice

We have so far considered only homogeneous flows with the same rates. In the next two cases, we consider an extreme case where one large flow and many small flows are aggregated together for policing (Section 5.3). We repeat our experiments for four different source models. In Case 3, only the large flow is misbehaving. The results are summarized in Table 1. For all source models, we always successfully detect the misbehaving large flow and P_{fa} is at most 0.027.

Case 4: Hiding in the Aggregate

Case 4 addresses the scenario where misbehaving small flows "hide" in the aggregate with another large flow. The probability of a small flow being misbehaving is γ. Intuitively, we suspect that detection is harder in this case, because the misbehaving flows can "steal" the idle bandwidth allocated to the large flow. Since the traffic policer can only enforce the total allocated rate, the misbehaving flows may not be detected. Figure 8 shows the P_{sd} achieved for different values of γ and Table 2 summarize P_{sd}, P_{fa} and P_{mis} for γ = 0.1 and 0.5. Surprisingly, we notice that the P_{sd} achieved with γ = 0.1 for

Table 2: Case 4: One large flow and many small flows. γ of small flows misbehave. η = 5, b_{TBF} = 6000, ϵ = 0.05.

Source Model	EXP1	EXP2	CBR	PARETO
$\gamma = 0.1$				
P_{sd}	0.74	0.43	0.75	0.57
P_{fa}	0.00066	0.011	0.0	0.0
P_{mis}	0.0032	0.00016	0.0047	0.0028
$\gamma = 0.5$				
P_{sd}	0.61	0.51	0.67	0.39
P_{fa}	0.00067	0.025	0.0	0.0
P_{mis}	0.0030	0.00047	0.0088	0.0022

EXP1, CBR, and PARETO sources are fairly close to the results in Case 2, where there is no large flow. But for EXP2, the success rate is significantly smaller (P_{sd}=0.43 in Case 4 as supposed to 0.54 in Case 1). When γ increases, the success rate P_{sd} decreases for EXP1, CBR and PARETO source models. With EXP2, P_{sd} fluctuates as we vary γ, and is actually higher at γ=0.5 than γ=0.1. This is because the active cycle of EXP2 is very short (10%), and can easily go undetected if it coincides with the idle period of the large flow. However, when the fraction of misbehaving flows increases, there is an increased likelihood that some of the misbehaving flows will synchronize or overlap in their active cycles, leading to overflow of the TBF at the traffic policer. When the aggregate rate is violated, all the flows sharing the same subfield ($FidIn$ or $FidEg$) will be monitored individually (micro-monitoring) and the misbehaving flow can be correctly identified. The probability of false alarms P_{fa} and mis-marked packets P_{mis} are negligible in this case across different values of γ and source models.

5.5 Further Sensitivity Analysis

So far, we have been considering flows with homogeneous source characteristics in our simulations. The next experiment uses a random mixture of the four different source models (EXP1, EXP2, CBR and PARETO), each with different peak rates, idle times and burst times. Each arriving flow chooses among these source models at random. We repeat the experiment in Case 2, with η=1 using heterogeneous flows (HET), and compare the results with Case 2 where homogeneous flows are used. Results are summarized in Table 3. With HET sources, the success rate P_{sd} is lower than all the other homogeneous source models, but the differences in P_{fa} and P_{mis} are negligible. It is difficult to tune the hysteresis or TBF parameters to optimize the overall performance since the source characteristics are not known a priori.

We repeat the Case 2 experiment using the EXP1 source with the following modifications:

a. DCAP without micro-policing mode, and

b. DCAP deployed at ingress ERs only.

Results show that only 23% of misbehaving flows are detected in (a), and 53% in (b), which is significantly lower than 79%, when DCAP is deployed at both ingress and egress ERs (Figure 7).

6. IMPLEMENTATION

In this section, we provide a brief description of a DCAP prototype implemented on top of the Click modular router [25]. We use this implementation to evaluate certain performance metrics

which could not be accurately quantified through simulations. One such important metric is the overhead incurred at an edge router by adding DCAP control functionalities. The current implementation works on Linux 2.2.16 and 2.2.17 kernels.

6.1 Overview of Prototype

Click is a Linux-based software router, and is assembled from packet processing modules called elements. Individual elements implement simple router functions like packet classification, queuing and scheduling. We extend the Click router to support three additional elements: the traffic policing (TP) unit, the reservation agent(RA), and DCAP agent. The RA is responsible for directing flow requests to the Resource Manager (RM) and forwarding responses from the RM to the client. The TP is implemented as a set of token bucket filters to police admitted flows. When a specific group of aggregate traffic exceeds its allocated threshold, an alarm is sent to the DCAP agent, which then handles the micro flow policing and verification process to identify the misbehaving flows as described in Section 3.2.

The communication between the Click router and the Resource Manager is performed through SNMP. In order to enhance the throughput of the Click router, we reduce the number of context switches required for processing the control packets from the RM by batching the messages from the RM to the Click router.

6.2 Experimental Setup

Using our DCAP prototype, we measure the performance overhead of adding the Reservation and Monitoring agents in an edge router. To quantify this overhead, we compare the maximum throughput obtained from our implementation to a basic implementation of Click which did not contain any of the monitoring tools (hereafter referred to as default Click). This experiment helps provide insights as to whether it is practical to deploy DCAP.

For evaluation purposes, we set up our own cluster of machines over a 10.0.0.0/24 network. The throughput measurements depend on the configuration of the network testbed and the machine implementing the Click router. For studying the performance, we restrict ourselves to the communication between the RM and one edge router (implemented on top of Click). The machine running the Click router has the following configuration: Pentium-III 650 Mhz, 3com 3c905 Ethernet controller. We use one other machine with a similar configuration as the RM. We used four other machines as sources and sinks of traffic. All these machines are connected to a backbone router using 100 Mbps connections. The router is a Bay Networks Accelar-1100B router with the capacity to support 16 100 Mbps Ethernet ports. The traffic statistics was periodically sent to the RM every 100ms. We modified Mgen [2], a publicly available traffic modeling software, to generate traffic for our experiments.

Table 3: Comparisons between heterogeneous and homogeneous source models: γ = 0.1, b_{TBF} = 6000,ϵ = 0.05.

Source Model	HET	EXP1	EXP2	CBR	PARETO
$\eta = 1$ P_{sd}	0.55	0.79	0.73	0.83	0.64
P_{fa}	0.0	0.0	0.14	0.0025	0.0
P_{mis}	0.0030	0.0028	0.00016	0.0079	0.0031

Table 4: Average Performance Reduction vs number of flows

Number of flows	Reduction in Throughput(%)
≤ 300	≤ 0.1
350	1.5
400	2.8
450	1.6

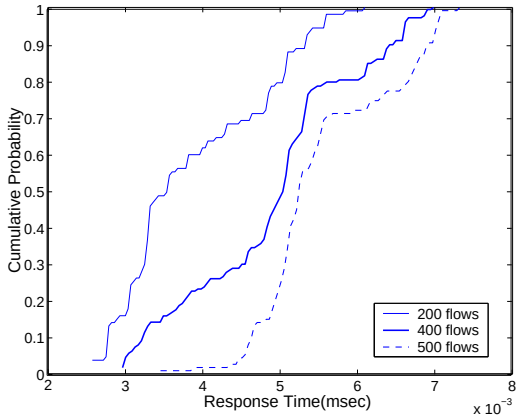

Figure 9: Response Time of Flow Allocation for varying loads

6.3 Experimental Results

In our first experiment, we measured the maximum throughput of our implementation at different loads and compared it to a default Click router. A basic flow in our setup has a bandwidth of 80 kbps and a packet size of 1024 bytes. As the number of flows increases, the amount of policing and state needed at the edge router also increases.

Table 4 shows the average reduction in the throughput of the access router for varying number of flows. We make two observations. First, the overhead incurred due to DCAP, i.e., the percentage of throughput degradation from default Click, is small. The maximum throughput difference observed in all our experiments is 5% and the average degradation is much smaller than this. Second, we notice that the throughput of the Click router saturates in our system at around 400 flows. Beyond 500 flows, we observed packet drops in the network interface.

In the second experiment, we measured the response time for flow allocation at different loads. We achieved a particular load in the system by maintaining a constant number of active flows and sending dummy flow requests to the Click router from the Traffic generator. There are three stages in the process of obtaining a response for a flow request: the RA in Click forwards the request to the RM, the RM performs admission control on the flow, and the RM sends the response to the requesting entity through the RA.

In Figure 9, we plot the cumulative distribution of the response time at three different loads: 200 flows (small load), 400 flows (saturation point) and 500 flows (high load). From the graph, we can observe that the mean response time increases as the load increases and the CDF shifts to the right. In all our experiments, we observe a minimum response time of 2.5 ms and a maximum of 7.2 ms. Our results indicate that the standard deviation of our response time is high. This can be attributed to the batching of responses at the RM and timer-based processing of flow requests at the Click

router. However, this variance is tolerable since the mean response time is small.

7. CONCLUSIONS

Although detection of misbehaving flows has been recognized as an important aspect of resource control, a practical and scalable way of implementing it has not been studied in great detail. This paper proposes a new scheme called DCAP (Detection via Collaborative Aggregate Policing) for policing incoming flows and detecting misbehaving behavior without requiring per-flow state maintenance at any edge routers. By aggregating flows for group policing, DCAP only requires $O(\sqrt{n})$ state maintenance at edge routers (where n is the number of flows), which is substantially better than previous approaches. Extensive simulations show that DCAP is effective and robust across a variety of source models and extreme cases. For the EXP1 source (VoIP type), DCAP can successfully detect 79% of misbehaving flows with almost zero false alarms and about 0.3% incorrectly-dropped packets. However, further study is needed to improve the detection of bursty misbehaving sources. Our approach has significant practical value since DCAP can be implemented with simple per-packet operations in a high-speed line card on a router. Our prototype demonstrates that DCAP adds minimal processing overhead to edge routers.

7.1 Future Work

As part of the future work, we will address the various practical issues of deploying DCAP in the existing Internet.

Changes to Routers: To deploy DCAP, no changes are required in the core routers, while the policing and monitoring need to be added to the edge routers. From our Click implementation, we infer that the modifications needed to add the extra DCAP mechanisms in an edge router is minimal. The entire implementation consists of 4463 lines of C++ code. Per-packet operations of DCAP can also be implemented in hardware.

Alternate Accounting Mechanism: In this paper, we perform regular sampling during the "micro-policing" phase of DCAP to track the bandwidth usage of individual flows within a sub-group. In future, we will consider alternative accounting mechanisms, including the scheme proposed in [14], to track the top flows (also known as heavy hitters) that contribute the most to the traffic within a group.

Security Issues: In Section 3.1.3, we briefly specified some of the security concerns (like bogus identifiers) with our solution. While we have made some in-roads towards addressing some of these concerns, a more detailed analysis of security issues is part of future work.

8. ACKNOWLEDGMENTS

We are grateful to Jennifer Rexford, Scott Shenker, Eddie Kohler, and Ion Stoica for their insightful feedback, and to Nina Taft, Supratik Bhattacharyya, and Dina Papaqianaki for sharing their knowledge on current IP-backbone topology and traffic models.

9. ADDITIONAL AUTHORS

Additional authors: George J. Lee (Massachusetts Institute of Technology - Department of Electrical Engineering & Computer Science, email: `gjl@mit.edu`).

10. REFERENCES

[1] Cisco Netflow. http://www.cisco.com/warp/public/732/Tech/netflow/.

[2] The Naval Research Laboratory (NRL), Multi-Generator (MGEN) toolset. http://manimac.itd.nrl.navy.mil/MGEN/.

[3] Verio service level agreement: Terms and definitions. http://www.verio.com/company/policies/sla/tsandcs.cfm.

[4] ITU-T: Recommendation I. 371, Traffic control and congestion control in b-isdn, 1993.

[5] ITU-T Recommendation P.59, Artificial conversational speech, 1993.

[6] M. Basseville and I. V. Nikiforov. *Detection of Abrupt Changes: Theory and Application*. Prentice Hall, April 1993.

[7] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weise. An architecture for differentiated services. RFC 2475, IETF, December 1998.

[8] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin. Reservation protocol (RSVP) version 1 functional specification. RFC 2205, IETF, September 1997.

[9] M. Butto, E. Caverolla, and A. Tonietti. Effectiveness of the leaky bucket policing mechanism in atm networks. *IEEE J. Selected Areas in Communications*, 9(3):335–342, April 1991.

[10] C.-N. Chuah, L. Subramanian, R. H. Katz, and A. D. Joseph. Qos provisioning using a clearing house architecture. In *Proc. of IFIP 5th Intl. Workshop on Quality of Service*, pages 115–124, June 2000.

[11] D. Clark, S. Shenker, and L. Zhang. Supporting real-time applications in an integrated services packet network: Architecture and mechanism. In *Proc. ACM SIGCOMM*, August 1992.

[12] A. Demers, S. Keshav, and S. Shenker. Analysis and simulation of a fair queuing algorithm. *Internetworking: Research and Experience*, 1:3–26, January 1990.

[13] Z. Duan, Z. L. Zhang, and Y. T. Hou. Service overlay networks: Sla, qos and bandwidth provisioning. In *Proc. International Conference on Network Protocols*, November 2002.

[14] C. Estan and G. Varghese. New directions in traffic measurement and accounting. In *Proc. ACM SIGCOMM Internet Measurement Workshop*, November 2001.

[15] K. C. F. Baker and and A. Smith. Management information base for the diffserv architecture. Internet draft, IETF, May 2002. draft-ietf-diffserv-mib-16.txt.

[16] A. Feldmann, A. Greenberg, C. Lund, N. Reingold, J. Rexford, and F. True. Deriving traffic demands for operational ip networks: Methodology and experience. *IEEE/ACM Trans. Networking*, 9(3):265–279, June 2001.

[17] W. Feng, D. Kandlur, D. Saha, and K. Shin. Stochastic fair blue: A queue management algorithm for enforcing fairness. In *Proc. IEEE INFOCOM*, April 2001.

[18] W. Feng, D. D. Kandlur, D. Saha, and K. G. Shin. Blue: A new class of active queue management algorithms. In *Proc. IEEE INFOCOM*, April 1999.

[19] C. Fraleigh, C. Diot, B. Lyles, S. Moon, P. Owezarski, K. Papagiannaki, and F. Tobagi. Design and deployment of a passive monitoring architecture. In *Passive and Active Measurement Workshop*, April 2001.

[20] S. Goldwasser and M. Bellare. Lecture notes on cryptography. http://www.cs.ucsd.edu/users/mihir/papers/gb.html.

[21] M. Grossglauser and D. Tse. A framework for robust measurement-based admission control. In *Proc. ACM SIGCOMM*, pages 237–248, September 1997.

[22] S. Jamin, P. Danzig, S. Shenker, and L. Zhang. A measurement-based admission control algorithm for integrated services packet networks. *IEEE/ACM Trans. Networking*, 5(1):56–70, February 1997.

[23] S. Jamin, S. Shenker, and P. Danzig. Comparison of measurement-based admission control algorithms for controlled-load service. In *Proc. IEEE INFOCOM*, April 1997.

[24] E. W. Knightly and J. Qiu. Measurement-based admission control with aggregate traffic envelopes. *IEEE/ACM Trans. Networking*, 9(2):199–210, April 2001.

[25] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek. The click modular router. *ACM Trans. on Computer Systems*, 18(4), November 2000.

[26] S. Machiraju, M. Seshadri, and I. Stoica. A scalable and robust solution for bandwidth allocation. In *Proc. of IFIP 5th Intl. Workshop on Quality of Service*, May 2002.

[27] A. K. Parekh and R. G. Gallager. A generalized processor sharing approach to flow control in integrated services networks: The single-node case. *IEEE/ACM Trans. Networking*, 1(3), June 1993.

[28] A. K. Parekh and R. G. Gallager. A generalized processor sharing approach to flow control in integrated services networks: The multiple node case. *IEEE/ACM Trans. Networking*, 2(2), April 1994.

[29] E. P. Rathgeb. Modeling and performance comparison of policing mechanisms for atm networks. *IEEE J. Selected Areas in Communications*, 9(3):325–334, April 1991.

[30] S. Shenker, C. Partridge, and R. Guerin. Specification of guaranteed quality of service. RFC 2212, IETF, September 1997.

[31] S. Shenker and J. Wroclawski. General characterization parameters for integrated service network elements. RFC 2215, IETF, September 1997.

[32] A. Sridharan, S. Bhattacharyya, C. Diot, R. Guerin, J. Jetcheva, and N. Taft. On the impact of traffic aggregation on routing performance. In *Proc. of Intl Teletraffic Congress*, September 2001.

[33] I. Stoica, S. Shenker, and H. Zhang. Core-stateless fair queuing: Achieving approximately fair bandwidth allocations in high speed networks. In *Proc. ACM SIGCOMM*, pages 118–130, September 1998.

[34] L. Subramanian, I. Stoica, H. Balakrishnan, and R. H. Katz. Overqos: Offering qos using overlays. In *First Workshop on Hot Topics in Networking(HotNets)*, October 2002.

[35] B. Tietelbaum and R. Geib. Internet2 qbone: A test bed for differentiated services. In *Proc. of INET*, June 1999.

[36] J. S. Turner. Managing bandwidth in atm networks with bursty traffic. *IEEE Network*, pages 50–58, September 1992.

[37] W. Willinger, M. Taqqu, R. Sherman, and D. Wilson. Self-similarity through high-variability: Statistical analysis of ethernet lan traffic at the source level. In *Proc. ACM SIGCOMM*, pages 100–113, August 1995.

Guidelines for Interdomain Traffic Engineering

Nick Feamster
Laboratory for Computer Science
Massachusetts Institute of Technology
Cambridge, MA 02139
feamster@lcs.mit.edu

Jay Borkenhagen
AT&T IP Services
AT&T Labs
Middletown, NJ 07748
jayb@att.com

Jennifer Rexford
Internet and Networking Systems
AT&T Labs – Research
Florham Park, NJ 07932
jrex@research.att.com

Abstract

Network operators must have control over the flow of traffic into, out of, and across their networks. However, the Border Gateway Protocol (BGP) does not facilitate common traffic engineering tasks, such as balancing load across multiple links to a neighboring AS or directing traffic to a different neighbor. Solving these problems is difficult because the number of possible changes to routing policies is too large to exhaustively test all possibilities, some changes in routing policy can have an unpredictable effect on the flow of traffic, and the BGP decision process implemented by router vendors limits an operator's control over path selection.

We propose *fundamental objectives* for interdomain traffic engineering and specific guidelines for achieving these objectives *within the context of BGP.* Using routing and traffic data from the AT&T backbone we show how certain BGP policy changes can move traffic in a predictable fashion, despite limited knowledge about the routing policies in neighboring AS's. Then, we show how operators can gain greater flexibility by relaxing some steps in the BGP decision process and ensuring that neighboring AS's send consistent advertisements at each peering location. Finally, we show that an operator can manipulate traffic efficiently by changing the routes for a small number of prefixes (or groups of related prefixes) that consistently receive a large amount of traffic.

Categories and Subject Descriptors

C.2.2 [**Computer-Communication Networks**]: Network Protocols–*Routing Protocols*; C.2.3 [**Computer-Communication Networks**]: Network Operations–*Network management, Network monitoring, Public networks*; C.2.5 [**Computer-Communication Networks**]: Local and Wide-Area Networks–*Internet*

General Terms

Measurement, Performance

1. Introduction

Operating a large IP backbone requires continuous attention to the distribution of traffic over the network. Equipment failures and changes in routing policies in neighboring domains can trigger sudden shifts in the flow of traffic. Flash crowds caused by special events and new applications can also cause significant changes in the load on the network. Network failures and traffic fluctuations degrade user performance and lead to inefficient use of network resources. Network operators adapt to changes in the distribution of traffic by adjusting the configuration of the routing protocols running on their routers. Additionally, routing configuration changes are often necessary after deploying new routers and links. Developing effective techniques for adapting routes to the prevailing traffic and topology has been an active area of research and standards activity during recent years [1, 2, 3, 4]. Previous work on traffic engineering has focused predominantly on *Interior* Gateway Protocols (IGPs), such as OSPF, IS-IS, and MPLS, which control the flow of traffic within a single Autonomous System (AS).

In practice, though, most traffic in a large backbone network traverses multiple domains, making interdomain routing an important part of traffic engineering. We motivate the need for interdomain traffic engineering with three examples:

- *Congested edge link:* The links between domains are common points of congestion in the Internet. Upon detecting an overloaded edge link, an operator can change the interdomain paths to direct some of the traffic to a less congested link.
- *Upgraded link capacity:* Operators of large IP backbones frequently install new, higher-bandwidth links between domains. Exploiting the additional capacity may require routing changes that divert traffic traveling via other edge links to the new link.
- *Violation of peering agreement:* An AS pair may have a business arrangement that restricts the amount of traffic they exchange; for example, the outbound and inbound traffic may have to stay within a factor of 1.5. If this ratio is exceeded, an AS may need to direct some traffic to a different neighbor.

The state of the art for interdomain traffic engineering is extremely primitive. The IETF's Traffic Engineering Working Group, which has focused almost exclusively on intradomain traffic engineering, recently noted that interdomain traffic engineering "is usually applied in a trial-and-error fashion. A systematic approach for inter-domain traffic engineering is yet to be devised" [1]. Operators make manual changes in the routing policies without a good understanding of the effects on the flow of traffic or the impact on other domains.

Ultimately, this ad hoc approach to interdomain traffic engineering must evolve into mature, well-tested guidelines and mechanisms. This paper is a first step in that direction. Recent previous work has presented a high-level overview of interdomain traffic engineering [5] and described the traffic data that must be measured to perform interdomain traffic engineering [6]. In addition, several commercial products help large campus and corporate networks balance load over connections to multiple upstream providers [7]; however, these products do not address the challenges of traffic engineering for large ASes in the core of the Internet. Our work is the first to propose *fundamental objectives* for interdomain traffic engineering, as well as specific guidelines for service providers to achieve these objectives within the context of BGP. We argue that

the guidelines we present are practical by characterizing traffic and routing data from a large, tier-1 IP backbone.

Neighboring ASes use the Border Gateway Protocol (BGP) to exchange routing information to provide end-to-end connectivity between hosts in different domains [8, 9, 10]. Each BGP advertisement announces reachability to a destination prefix that represents a block of IP addresses. Each advertisement includes a list of the ASes in the path, along with several other attributes. The routers in each AS apply local routing policies that manipulate these attributes to influence the selection of the best route for each destination prefix and to decide whether to propagate this route to neighboring ASes. Operators affect the flow of traffic by tuning the local routing policies that affect the selection of the best path for a destination prefix. Choosing the appropriate configuration is difficult since it depends on the network topology, the IGP parameters, the BGP advertisements from neighboring ASes, and the current traffic patterns. Our work focuses on the impact of BGP policies on the flow of traffic *leaving* an AS at the egress points that connect to neighboring domains. Some traffic engineering tasks necessitate changes to how traffic *enters* the network. However, controlling how traffic enters the network in a predictable way requires coordination with neighboring domains [1]. The results of our analysis of outbound traffic can be applied by the neighboring ASes to influence how traffic enters the network.

Interdomain traffic engineering is significantly more complicated than intradomain traffic engineering. While IGPs select paths based on link metrics, such as static weights or dynamic load information, BGP advertisements do not explicitly convey any information about the resources available on a path. BGP routing policies are complex and depend on a variety of factors, such as the commercial relationships with neighboring ASes [11]. The selection of the best path for each prefix depends not only on the routing policies but also on the advertisements sent by neighboring domains. Operators have, at best, indirect influence on BGP path selection. In fact, changing the BGP policy in one AS may alter the advertisements propagated to neighboring domains, which may inadvertently affect how traffic enters the AS. The constraints that BGP imposes on making "good" routing decisions makes moving to a radically different interdomain routing paradigm desirable, but extremely difficult in practice. Rather than proposing a new routing protocol, our analysis identifies ways to support traffic engineering within the existing BGP framework.

Router vendors support a wide variety of configuration commands that provide significant flexibility in specifying BGP policies. Selecting the right policy changes for a particular traffic-engineering task is challenging, especially for service providers that have many connections to neighboring domains. Our study focuses on developing traffic engineering techniques that achieve the following objectives:

- *Achieving predictable traffic flow changes:* Some routing changes have effects that are difficult to predict in advance, due to the routing policies in other domains. Our analysis identifies approaches for tuning policies in ways that have predictable outcomes and limit the changes seen by neighboring domains.

- *Limiting the influence of neighboring domains:* Certain practices, such as sending inconsistent advertisements at different peering locations, can have a significant impact on the path selection process. Our analysis shows how operators can check for these practices and use BGP policies that limit their effects.

- *Reducing the overhead of routing changes:* Changing the routing policy may trigger new advertisements that impose a load on the routers and a delay for converging to a new set of routes. Our analysis shows that operators can limit overhead by focusing on the small number of prefixes (or groups of prefixes) that consistently receive a large amount of traffic.

Although this paper primarily describes how to achieve these objectives within the context of BGP, these objectives are applicable to interdomain traffic engineering *in general*. We discuss our results for these three objectives after a brief background section on the BGP protocol and traffic engineering tools and an overview of our measurement data from AT&T's IP backbone.

2. BGP Traffic Engineering

This section presents an overview of BGP and the attributes associated with route advertisements. We briefly describe tools that could allow operators to adjust the routing configuration to the prevailing traffic.

2.1 Border Gateway Protocol

Internet routing operates at the level of address blocks, or prefixes. Each prefix consists of a 32-bit address and a mask length; for example, $192.0.2.0/24$ represents the 256 addresses ranging from $192.0.2.0$ to $192.0.2.255$. An IP router constructs a forwarding table that is used to select the output interface for each incoming packet, based on the longest-matching prefix for that destination address. Routers in different ASes use BGP to exchange update messages about how to reach different destination prefixes. A router sends an *announcement* to notify its neighbor of a new route to the destination prefix and sends a *withdrawal* to revoke the route when it is no longer available. Each advertisement includes a number of attributes about the route, including the list of ASes along the path to the destination prefix. Before accepting an advertisement, the receiving router checks for the presence of its own AS number in the AS path to detect and remove routing loops.

A router may receive routes for the same prefix from multiple neighboring ASes. The router applies *import policies* to filter unwanted routes and to manipulate the attributes of the remaining routes. Ultimately, the router invokes a *decision process* to select exactly one "best" route for each destination prefix among all the routes it hears. The router then applies *export policies* to manipulate attributes and decide whether to advertise the route to neighboring ASes. In addition to exchanging BGP messages with neighboring domains, an AS may use internal BGP (iBGP) to distribute routing information among its routers. Ultimately, every router must select a single best route for each prefix among the advertisements from the various external BGP (eBGP) and iBGP neighbors.

BGP advertisements can include numerous attributes [9], and the BGP decision process implemented by router vendors has several steps, which proceed in order and sequentially eliminate candidates for the best route [12, 13, 14]. To simplify the discussion, we focus on five main steps in the selection process:

1. *Highest local preference*: Prefer routes with the highest local preference, assigned by the import policy and conveyed to other routers via iBGP.

2. *Shortest AS path*: Prefer routes with the shortest AS path length, as conveyed in the BGP advertisement.

3. *eBGP over iBGP*: Prefer routes learned via eBGP over routes learned via iBGP, since leaving the AS directly is preferable to traveling through the AS.

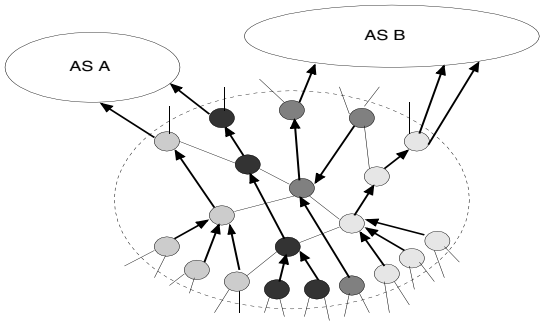

Figure 1: Flow of traffic from ingress routers to the egress links. Each node represents a router within the AS. Routers with the same shading have the same closest egress point.

4. *Lowest IGP metric*: Prefer routes with the smallest intradomain (IGP) metric to reach the next hop. This enables each router to select its "closest" exit point.

5. *Lowest router ID*: Prefer the route learned from a router with the lowest identifier, as conveyed during BGP session establishment. This step breaks ties between routes that are equally good after the previous steps have been applied.

We primarily focus on how operators can assign local preference to influence the first step of the BGP decision process; router configuration languages provide operators with flexibility in assigning local preference based on the destination prefix, the AS path, and other BGP attributes. Our observations also apply to other BGP attributes, such as the origin type and the multiple-exit discriminator (MED), as discussed in the Appendix.

2.2 Traffic Engineering Tools

The construction of the forwarding table at each router depends on the complex interaction of BGP routing policies, the distribution of update messages via iBGP, and the IGP parameters. Over time, each router receives eBGP messages from neighboring domains and iBGP messages that report the best routes seen at other routers in the AS. The routers also participate in an IGP that affects their selection of the best path, as well as the route through the domain to reach the BGP next hop. Figure 1 shows a collection of routers that select different routes toward a destination prefix reachable via ASes A and B. Each router selects a route with the "closest" egress point, based on the IGP weights. Modeling the impact of interdomain routing on the flow of traffic in the network requires a way to separate the roles of BGP policies and IGP parameters in the construction of the forwarding table. It also requires a way to capture how the asynchronous exchange of eBGP and iBGP messages affects the selection of the best path at each router.

Operators can use tools to predict the influence of changes to the BGP policies and IGP weights on the flow of traffic[1], as shown in Figure 2. The first module [17] captures the first three steps of the BGP decision process that do not depend on the IGP weights. For each prefix, this produces a set of egress points, where the final selection of the closest egress point may vary at different routers inside the AS, as shown in Figure 1. The second module [4] captures the selection of the closest egress point, based on the IGP cost and the router ID tie-break (steps 4-5) for each router in the domain; this module also identifies the IGP path(s) associated with the minimum cost. Together, the two tools predict the how traffic would flow through the AS for each ingress point and destination prefix. By combining this information with traffic measurements from the ingress points [18], the tools can predict how a change in routing configuration would influence the load on each link in the domain.

However, to use these tools effectively, operators must first be able to identify good candidate changes to the routing configuration. BGP is a policy-based routing protocol that provides substantial flexibility in matching and assigning the attributes in the advertisement messages. This is important for two main reasons. First, the search space of changes to BGP policies and IGP weights is extremely large—far too large to explore exhaustively. Second, BGP permits operators to make ineffectual or even harmful changes in an attempt to shift traffic from one path to another. Making these kinds of changes in the operational network can cause significant degradation in user performance, and trigger unnecessary routing updates throughout the Internet. Experiments with routing changes should be conducted outside of the network, using accurate tools to predict the effects. Still, it is important to avoid spending valuable time exploring innumerable changes to BGP policy in the tool. In this paper, we identify effective and efficient ways to tune the BGP import policies for traffic engineering.

3. Measurement Data

Effective traffic engineering requires an understanding of the network paths and traffic volume associated with each destination prefix. This section describes the collection of BGP routing tables and flow-level traffic measurements from the routers that connect the AT&T backbone to other large providers.

3.1 BGP Routing Tables

Ideally, the operator would have a complete, up-to-date snapshot of all of the BGP updates heard from eBGP neighbors, which would enable the operator to precisely determine how a change in import policies would affect the routing decision made by each router. However, acquiring a timely view of all of the BGP update messages in the network may be difficult. Ideally, IP routers would be able to provide a continuous feed of all of the routes (both best and alternate paths) as they arrive, but this feature is not universally available. An alternate approach is to extract the set of paths from the BGP routing table (the Routing Information Base) from each router at the edge of the network. A simple script can connect to each router and issue a command to dump the current routing table (e.g., "show ip bgp" in Cisco IOS). Figure 3 shows an example line in a BGP routing table. The entry lists a single route for prefix 38.138.55.0/24 that was learned via iBGP (the "i" before the prefix) and has a next-hop IP address of 192.168.0.10. The routing table entry includes other attributes such as the multiple-exit discriminator (MED) value (2130), local preference (100), AS path (1 701 17031), and the origin type ("i" for IGP). The ">" symbol indicates that this is the router's "best" route for this prefix.

Using routing tables to extract all paths to a prefix imposes two limitations on the quality of the data. The first limitation concerns the *consistency* of the data. Dumping the entire routing table imposes a load on the router, making it impractical to collect these tables very frequently. In fact, since routing table dumps do not occur instantaneously, the state of the table may change during the dump itself; most router implementations avoid this problem by defer-

[1]The use of these tools rests on the assumption that the inputs are relatively stable. The operator controls the import policies and the IGP weights, and topology changes occur only in response to unexpected failures and planned maintenance/upgrades. Although the BGP updates from other ASes change over time, the BGP routes for most prefixes stay the same for weeks at a time [15]; the BGP routes for the most popular prefixes are especially stable [16]. In addition, we envision that operators would not need to change BGP policies all that frequently—only in response to significant changes in the topology or traffic demands.

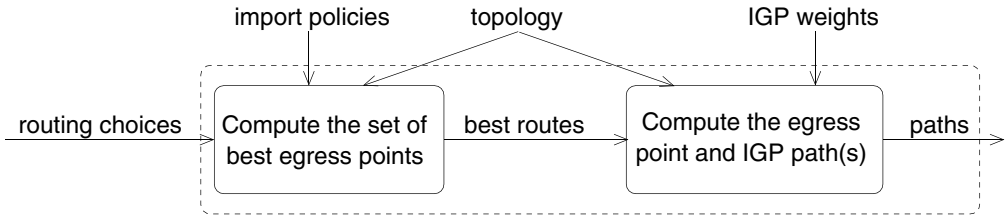

Figure 2: Predicting the impact of BGP policies and IGP weights on the flow of traffic.

```
   Network          Next Hop            Metric LocPrf Weight Path
*>i38.138.55.0/24   192.168.0.10          2130    100      0 1 701 17031 i
```

Figure 3: Example BGP routing table entry for prefix 38.138.55.0/24

ring changes in the routing table until the dump is complete. Table dumps may not occur at exactly the same time across all routers, thus causing occasional inconsistencies in the network-wide view of the routing choices. The significance of these issues depends on how often routing changes occur relative to the frequency of the routing table dumps. Given that many routes are stable for days or weeks at a time [15, 16], these types of inconsistencies are likely uncommon. However, a live feed of BGP updates from each router would provide precise information about the routes available to each router at a particular time and would eliminate this concern entirely.

Relying on routing tables can adversely affect the *completeness* of the routing information. The routing table represents the collection of routes *after* the import policies have been applied. Hence, the table does not include any routes filtered by the import policy. Since we do not try to model changes in the filtering policy, this is not a significant limitation. Each routing table entry includes attributes such as local preference, origin type, and MED *after* manipulation by the existing import policy. This does not preclude experimenting with different import policies that change the assignment of local preference or origin type, or that reset the MED value. Finally, routing table entries such as the one shown in Figure 3 do not include the community values included in the BGP advertisement. As such, these BGP tables are not useful for experimenting with import policies based on communities. Despite these shortcomings, the routing table data is sufficient for evaluating policies that set local preference based on the prefix and AS path.

We collected BGP routing tables from routers that connect the AT&T backbone to other large providers and extracted the routes for each prefix. We focused on the routes learned via eBGP and ignored the routes that were propagated from other routers via iBGP. To focus on routes that traverse the peering links, we excluded prefixes that are reached directly by connections to customers of the AT&T backbone. Suppose a prefix has routes learned from both customers and peers. If the customer route has a high local preference, then we do not include any of the routes for this prefix in our analysis, since traffic to this prefix should travel via the customer link(s) rather than peering links. On the other hand, if the customer route has a low local preference (indicative of a backup route), then we include the routes learned from peers, since traffic to this prefix should travel via peering links rather than customer links.

3.2 Flow-Level Traffic Measurements

The influence of changes in BGP import policies depends on the amount of traffic that moves to new routes. Our analysis draws on daily summaries of the traffic leaving the AT&T backbone via peering links. The data were collected using Cisco's Netflow feature [19]. Netflow produces a single measurement record for each "flow"—a group of packets that match in key IP and TCP/UDP header fields and appear close together in time. Each Netflow record includes the start and finish time of the flow, the number of bytes and packets, the source and destination IP addresses, the mask length for the longest-matching prefix in the forwarding table, the TCP/UDP port numbers, and several other fields. The routers that connect AT&T to other large providers are configured to run Sampled Netflow [20], which performs one-out-of-N sampling of the packets before constructing the flows. This reduces the packet handling overhead and the number of flow records, at the expense of a reduction in accuracy.

The routers in each Point-of-Presence (PoP) were configured to send the measurement records to a dedicated collection machine. Each collection server was configured to aggregate the flow-level records to compute the volume of traffic for each destination prefix on an hourly time scale. Each flow-level record was associated with a destination prefix based on the destination IP address and the mask length. The collection server was configured to aggregate the measurement records separately for inbound and outbound traffic. Each Netflow record includes identifiers for the input and output links that carried the traffic for the packets in the flow. These links can be classified as edge and core links, based on a snapshot of the network topology. Outbound traffic travels from a core link to an edge link, whereas inbound traffic travels in the opposite direction.

The collection server corrected for the influence of one-out-of-N sampling at the router by multiplying the resulting traffic volumes by N. In addition, the collection server applied stratified sampling to reduce the processing overhead [21]. This sampling scheme focuses on a subset of the records based of the number of bytes associated with the flow. Records for large flows are always included in the aggregation. Smaller flows are included with a probability proportional to their size; the aggregation applies an appropriate correction factor to account for the effects of sampling. Together, the two forms of sampling make it possible to collect and analyze measurement data on a large number of high-speed links. Adjusting for the effects of sampling produces an unbiased estimator of the volume of traffic destined to each prefix. The estimates have very low variance, except for destination prefixes that receive an extremely low volume of traffic. In the next three sections, we analyze *daily* totals of outbound traffic volumes. We also avoid drawing conclusions about the amount of traffic associated with prefixes that have low (and, thus, potentially inaccurate) traffic volumes.

The Netflow measurements were collected throughout the day on March 1, 2002 and the BGP routing tables were dumped at approx-

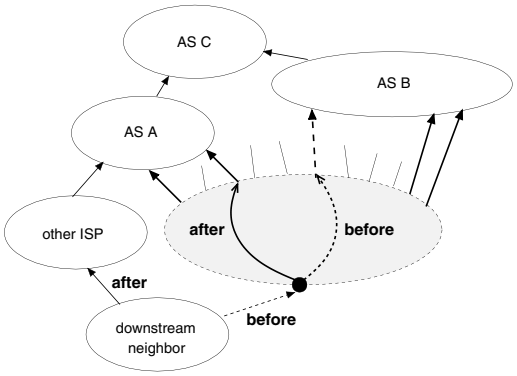

Figure 4: Neighbor's behavior upon receiving a new route.

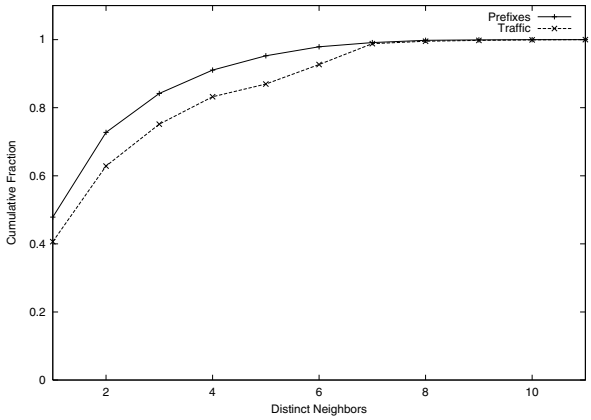

Figure 5: Cumulative distribution of the number of next-hop ASes among the shortest AS paths for a prefix. Prefixes that have advertisements from only one next-hop are ideal candidates for BGP traffic engineering, since policy changes can be made without changing inter-AS traffic flow.

imately 2 a.m. EST on the same day. Additionally, we collected the same data on April 1, 2002 to verify the results of the analysis. We parsed and preprocessed the data and stored the results in a MySQL database. One database table stores the set of eBGP-learned routes for each destination prefix. Each entry in this table includes the date of the BGP dump, the associated router, the advertised prefix, the AS path, and whether or not this route was a "best" route for that destination prefix. A second table stores daily summaries of the outbound traffic. Each entry in this table includes the date when the measurements were collected, the associated router, the destination prefix, and the number of bytes sent outbound to the destination prefix via that router.

4. Achieving Predictable Traffic Flow Changes

Effective traffic engineering relies on policy changes that have a *predictable* influence on the flow of traffic through the network, which is inherently difficult for two main reasons. First, modifying the import policies may cause the AS to change its choices for best routes, and thus send new routing advertisements *to* neighboring domains, which may in turn affect where and whether traffic enters the AS from these neighbors. Second, small changes in the advertisements sent *by* neighboring domains may cause unintended changes in the selection of the best routes for a destination prefix. In this section, we show that careful modification of import policies can control these effects and thus improve predictability of changes to the flow of traffic.

4.1 Avoid Globally-Visible Changes

When adjusting routing policies, operators should minimize the impact on the behavior of downstream neighbors. If a policy change causes neighboring domains to change their behavior (e.g., by selecting a different best route for a prefix), the amount of traffic entering the AS from these neighbors may be unpredictable. Suppose that a particular edge link is congested and the network operator assigns a lower local preference value to some of the routes traversing the congested link. The new import policy will remove these routes from the set of possible best routes for these prefixes, thus causing some routers to direct traffic for these destination prefixes to a different route via a different egress link. Moving the traffic reduces the load on the congested link. However, the affected routers might advertise a new route to their eBGP neighbors, such as downstream customers, potentially causing significant changes in the volume of inbound traffic.

In the example shown in Figure 4, ASes A and B both advertise paths to destinations in AS C. Initially, there are five "best" routes—two via AS A and three via AS B. Routers on the west coast route via AS A and routers on the east coast route via AS B. Suppose that the leftmost link to AS B is congested (as illustrated by the dashed line), and the import policy for this egress point is modified to assign a lower local preference to routes originating from AS C. After this change, some routers might switch from a route via the leftmost link to AS B to a route via the rightmost link to AS A. These routers would advertise the new best path to downstream neighbors. Depending on the neighbor's routing policies, the new advertisement might cause the neighbor to select a different next-hop AS (i.e., another ISP) for reaching this prefix. This could result in an unpredictable decrease in the volume of traffic entering the domain at this router. Similarly, the routing change could trigger an increase in traffic if other neighbors preferred the (A, C) route over the (B, C) route.

To prevent the effects of routing changes from propagating to neighboring domains, a network operator should only adjust routing policies for prefixes for which every potential best route has the same BGP attributes (except for the next-hop IP address, of course). This approach still gives an operator significant flexibility, because the operator can route traffic for that destination via *any subset* of these advertised routes without affecting the BGP advertisements seen by neighboring ASes. Depending on the BGP implementation, downstream ASes may not even receive a new BGP advertisement, since none of the attributes conveyed to eBGP neighbors has changed (this feature is called *non-transitive attribute filtering*).

For our data, 47.8% of the prefixes have shortest AS paths with a single next-hop AS, as shown in Figure 5; these prefixes contribute over 40% of the outbound traffic. For these prefixes, reducing the local preference at one peering location would shift traffic to another egress link to the *same* peer. In some cases, an operator may need to move traffic from one next-hop AS to another. As shown in Figure 5, a reasonable fraction of prefixes and traffic have shortest paths with two next-hop ASes (e.g., if these two ASes share a common, multi-homed customer like AS C). This is useful for moving traffic between two neighboring ASes without having to select routes with different AS path lengths. (The network may have routes to two ASes via the *same egress router*. In this case, it is possible to move traffic between egress links without changing the traffic flow *within* the AS.) Although this type of routing change requires sending a new advertisement to some downstream ASes,

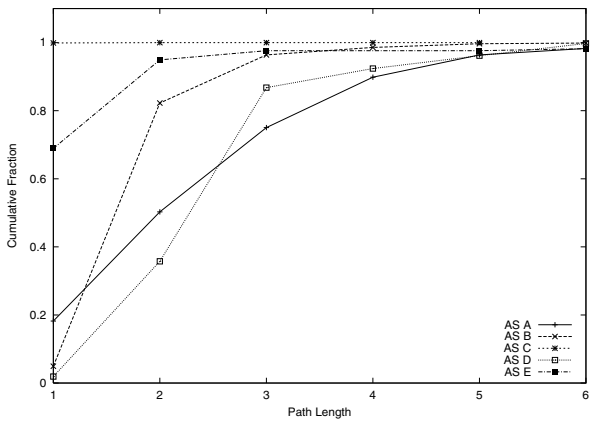

Figure 6: Cumulative fraction of outbound traffic vs. AS path length for various next-hop ASes. Assigning a lower local preference to all traffic destined for certain AS and path length can have vastly different effects depending on which ASes advertisements are affected.

advertising a route with the *same AS path length* reduces the likelihood that a downstream AS selects a best path through a different provider.

4.2 Limit Reaction to Minor Changes

Router configuration languages provide significant flexibility in assigning local preference values to routes. For example, these languages allow an operator to assign local preference based on the destination prefix or regular expressions on the AS path. However, the import policy has only an *indirect* affect on the path selection process. Changes in the advertisements sent by neighboring domains may cause the existing import policies to assign a different local preference value and shift traffic to or from a particular edge link. For example, suppose a neighboring domain D advertises a three-hop AS path "D B C" to reach a particular destination, and then later changes to the path "D A C"; this may occur for traffic engineering reasons, similar to the example shown in Figure 4. An import policy that sets local preference based a specific AS path "D B C" would assign a different value for a route with the path "D A C", which may cause an unintended shift in the traffic associated with the destination prefix.

Network operators can design import policies that are robust to small changes in BGP advertisements by avoiding policies that make such fine-grain distinctions between different AS paths. For example, suppose a network has several high-bandwidth links to AS D and one low-bandwidth link. Then, the import policy for the low-bandwidth link could be configured to assign a lower local preference value to certain routes based just on the origin AS (e.g., C) or even the AS path length. For example, the import policy could assign a small local preference to all destination prefixes with three-hop AS paths. This would divert traffic for destination prefixes with a three-hop (shortest) AS path to other egress points that have shortest paths with three AS hops. This approach is simple and does not depend on the exact sequence of ASes in the path. However, the specific effects of this technique depend on how traffic is distributed over different lengths of AS paths. This may vary across different next-hop ASes.

A network operator might expect to see differences in the distribution of traffic over AS path lengths and may have to consider per-AS traffic patterns when designing policies that are based on AS path length. Figure 6 shows the cumulative distribution of outbound traffic carried by best paths of different lengths. Each curve corresponds to the traffic traversing a different next-hop AS, identified by A, B, C, D, and E; for example, nearly all traffic to AS C follows a one-hop path, and nearly 70% of the outbound traffic to AS E travels over a one-hop AS path (where AS E is the next-hop AS). In contrast, the majority of traffic traveling via the other three ASes travels on AS paths of length two or three. These differences stem from the various roles ASes in the Internet can play, as well as historical and network-specific artifacts (e.g., a single ISP network might consist of multiple ASes). In some cases, an AS hosts a large number of services and directly-connected customers that do not have their own AS numbers. This type of network sends traffic over paths with a single AS hop, as shown in the plot for AS E. In other cases, an AS is a transit provider for a large number of tier-2 providers or multi-homed institutions. Outbound traffic to these types of networks is likely to traverse paths of different lengths, as shown in the plots for ASes A, B, and D.

5. Limiting the Influence of Neighboring ASes

The routing choices for each prefix depend on the routing advertisements heard from neighboring domains. The common practice of AS prepending (i.e., repeating an element in the AS path before readvertising to make the path appear longer) limits the ability to spread traffic over a large number of egress points in different parts of the network. In addition, although BGP import policies can reassign some attributes (such as origin type and MED), other attributes, such as the AS path, depend on the policies applied in other ASes. Inconsistencies in the routes advertised via different eBGP sessions with the same next-hop AS can reduce an operator's control over traffic flow. In this section, we quantify these effects and suggest techniques for increasing control over the flow of outbound traffic.

5.1 Limiting the Influence of AS Path Length

Even if advertisements are consistent across eBGP sessions to the same next-hop AS, path length has a significant influence on the comparison of routes via different ASes. AS prepending increases the length of the AS path by repeating an AS number multiple times to artificially make a path look longer. Consider an AS 100 that connects to AS 200 and AS 300, as in Figure 7. AS 100 may send a one-hop route to AS 200 and a two-hop route to AS 300 to encourage traffic destined to AS 100 to traverse a route via AS 200. An AS that connects to these two ASes would receive routes $(200, 100)$ and $(300, 100, 100)$, perhaps at different locations in the network. If both routes are assigned the same local preference, the AS would direct all of the traffic to the $(200, 100)$ paths. Alternatively, the operator could assign lower local preference to the $(200, 100)$ path, which would force all of the traffic to use the $(300, 100, 100)$ path. Using both paths (via different egress points in the network) is not possible in general.

We investigated the frequency of AS prepending in the BGP routing data from the AT&T network. Approximately 32% of the routes in the BGP tables had some amount of AS prepending. Figure 9 shows the distribution of the amount of prepending in these paths. The majority of the paths were extended by one or two hops; four paths were extended by as many as 16 hops. AS path prepending contributes to the diversity of AS path lengths, as shown by the cumulative distribution plot in Figure 10. The majority of prefixes have AS paths of a single length, and the majority of traffic is associated with these prefixes. However, about 40% of the prefixes have paths with different lengths. Most of these prefixes have paths with just two or three unique lengths. The different lengths stem

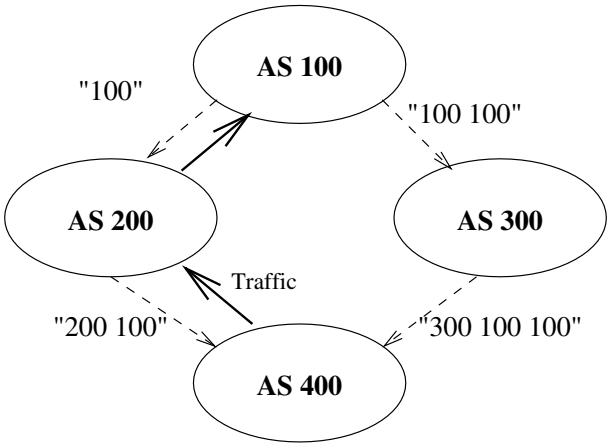

Figure 7: Example of AS path prepending. AS 100 can make a path look arbitrarily longer to downstream networks (e.g., AS 400) by prepending its AS to the path one or more times.

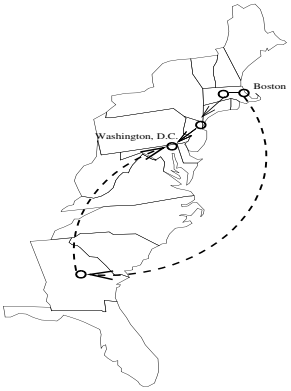

Figure 8: Shortest AS path length does not always reflect shortest network distance. A shorter path from Boston to Washington, D.C. that traverses two intermediate ASes on the way may be shorter than a path with one intermediate AS that that does not have a geographically proximal exchange point.

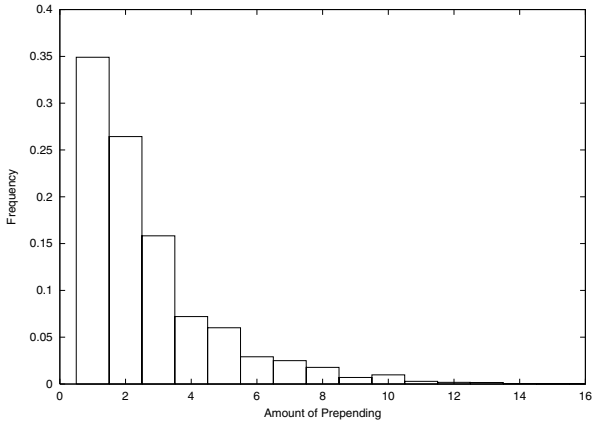

Figure 9: Frequency of AS prepending of different lengths for the 32% of all advertised routes that include some amount of prepending. Twelve advertised paths were extended by at least 14 hops.

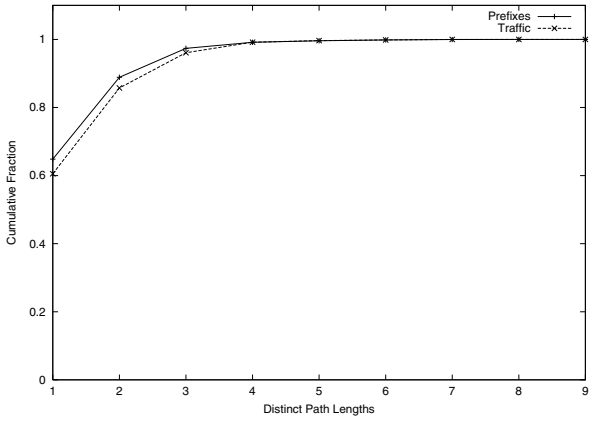

Figure 10: Cumulative distribution of the number of unique AS path lengths. Prefixes that have multiple AS path lengths limit flexibility in selecting a set of best routes.

from a mixture of AS prepending and routes with a different number of unique ASes in the path. In either case, the different lengths limit flexibility in selecting a set of best routes, since the second step in the BGP decision process forces all best paths to have the same length.

While small differences in AS path length restrict routing choices significantly, they are also not often indicative of the best route to a particular prefix. As shown in Figure 8, a path from Boston to Washington, D.C. that crosses two intermediate networks with conveniently-located exchange points is preferable to a path that has fewer AS hops, but requires the packets to travel to a distant exchange point[2]. Similarly, a path with fewer AS hops may traverse a network that is experiencing high latency or loss or contains many intra-AS router hops. Forcing all best paths to have the same AS path length may be unnecessarily restrictive. Figure 6 shows that, for many ASes, the majority of traffic travels over shortest AS paths of length 2 or 3. Furthermore, almost no traffic traverses AS paths of length 4 or longer.

Consequently, it may be effective to allow the set of best paths to include AS paths with small differences in length (e.g., having one 2-hop path and one 3-hop path to a prefix, rather than allowing only the 2-hop path). Coarse-grained AS path length categorization can be achieved by *disabling* step 2 of the BGP decision process and instead assigning local preference ranges based in part on AS path length. For example, a network operator could assign a range of local preference values to one-hop paths, another range to paths of length 2 or 3, and so on. This ensures that AS path length has an influence on the decision process without imposing the strict requirement that all best paths for a prefix must have the same length.

5.2 Consistent Advertisements from Neighbors

BGP update messages from neighboring ASes have a significant impact on the flow of traffic through a network. A neighbor AS can exert influence on how traffic leaves a network by sending inconsistent routing advertisements over different eBGP sessions. For example, suppose that a network connects to AS A at locations on the east and west coast. If AS A advertises a prefix only on the east

[2]Network operators in Europe face these challenges continually. These operators typically tag transatlantic routes with a particular community value and assign a different local preference value accordingly.

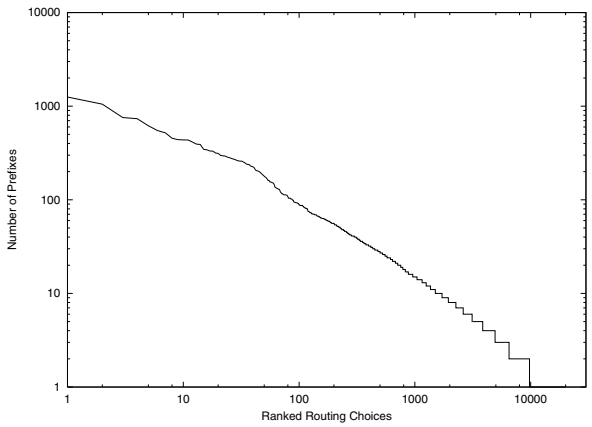

Figure 11: Distribution of the number of prefixes with the same routing choices (log-log scale).

coast, then this would force the other network to carry all of the outbound traffic for this prefix to the east coast. Alternatively, AS A might advertise the path with a different AS path length or origin type at different locations. Advertising inconsistent routes can have a significant and unpredictable influence on the flow of traffic by limiting the number of possible egress points; in addition, this practice is often a violation of peering agreements.

We analyzed the routes in the BGP tables to identify paths of different AS path lengths from the same next-hop AS for the same destination prefix. All but two peers, which local routing policy indicated were special cases, advertised consistent AS path lengths for more than 99% of advertised prefixes. However, our preliminary evaluation shows many instances where a peer advertises a prefix at some peering points but not others. Some inconsistencies can likely be explained by the asynchrony in downloading the BGP tables from the routers; in our ongoing work, we are trying to better understand the nature of these inconsistencies.

6. Reducing the Overhead of Routing Changes

Traffic engineering involves moving a portion of the traffic in the network from one link to another. The BGP import policies can select this traffic based on the prefixes and the attributes in the route advertisements. An operator could conceivably configure import policies to manage traffic on a per-prefix basis. In this section, we first argue that simpler import policies that focus on *groups* of related prefixes, such as prefixes with the same routing choices or the same origin AS, can achieve traffic engineering goals with a relatively small number of policy changes. Then, we argue that import policies should focus on the routes to *popular* destinations to move large amounts of traffic with a small number of routing changes. Finally, we discuss how operators can focus on prefixes (or groups of prefixes) with *stable* traffic volumes over time.

These three techniques reduce the overhead of routing changes in several ways. Moving groups of prefixes that carry more traffic eases management overhead by reducing the number of changes that an operator must make to achieve a task, and setting policies based on groups of prefixes with stable traffic volumes reduces the likelihood that an operator will have to be constantly adjusting routing policies to achieve a certain traffic engineering task.

6.1 Group Related Prefixes

Because a typical default-free BGP routing table contains routes for more than 100,000 prefixes, exploring all possible combinations of import policies is computationally intractable. Furthermore, import policies that are tailored to every prefix at every router would be extremely complicated to configure and expensive for the router to apply. Such fine-tuned policies might not remain appropriate following a shift in traffic or a change in the neighbors' routing advertisements.

Many prefixes have the same attributes across all eBGP advertisements from neighboring domains (the *routing choices* in Figure 2). For example, a single institution, such as a company or university campus, may announce a dozen different destination prefixes from a single location. These prefixes tend to have identical routes in a BGP table at an arbitrary point in the Internet[3]. Because many prefix advertisements have the same characteristics, a network operator can effect policy changes for a significant number of prefixes simply by changing policies based on characteristics in the routing advertisements (e.g., AS path properties), rather than on the specific prefix.

To identify groups of related prefixes, we propose a canonical representation of the *routing choices* announced by neighboring domains. Most of the steps in the BGP decision process depend on the import policy or IGP weights, except for the step based on AS path length. We classify each prefix based on the routers where the routes for that prefix were learned, as well as the AS paths that were learned for each prefix. If several prefixes are advertised to the same set of routers and, at each router, the routes for those prefixes have the same AS paths, we say that those prefixes have the same *routing choices*. This concept facilitates comparisons between different destination prefixes and can be useful for predicting the impact of changes in import policy, since many computations can be performed once per group of prefixes.

In our data, we find a total of $20,086$ unique representations of routing choices. Figure 11 shows the distribution of the number of prefixes associated with each set of routing choices, starting with the set with the largest number of prefixes. A set of routing choices is associated with five destination prefixes on average. However, in some cases, many more prefixes are associated with a particular routing choice. $2,142$ destination prefixes had exactly the same set of routing choices, and 88 sets of routing choices were associated with 100 or more prefixes. Because many prefixes have the same routing choices, a network operator can affect the routes for a large group of prefixes by selecting import policies based on the attributes in the routing advertisements, rather than on each specific prefix. For example, a network operator can manipulate the traffic for a group of prefixes by assigning local preference to these advertisements based on their common attributes, such as AS path characteristics.

6.2 Focus on Popular Destinations

Defining independent import policies even for $20,000$ unique routing choices is still an unreasonable requirement. Fortunately, the bulk of the traffic is concentrated in a small fraction of routing choices. The bottom curve in Figure 12 shows the cumulative distribution of the proportion of traffic destined to most popular prefixes. For example, traffic destined for the top 1% of the prefixes is responsible for about 20% of the outbound traffic volume. The top 10% of prefixes accounts for approximately 70% of the traffic. These results are consistent with the trends seen in earlier traffic measurement studies [18, 24, 25]. The results are more dramatic

[3]Previous work has made similar observations [22, 23]. However, this work did not consider the *volume* of traffic associated with these groups of prefixes, and focused on grouping the routes from a *single* BGP routing table, rather than constructing an *AS-wide* view of routing choices across multiple edge routers.

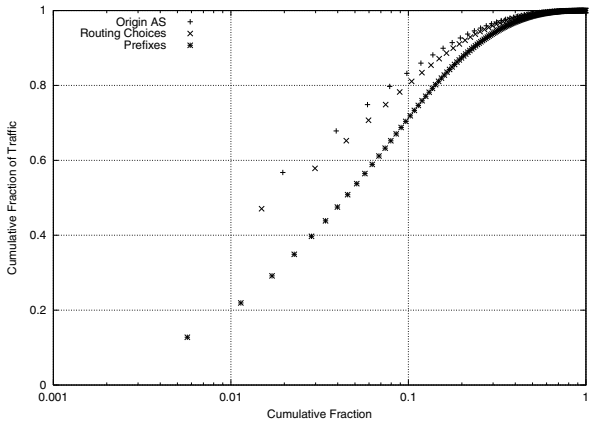

Figure 12: Cumulative distribution of traffic for by individual prefixes, prefixes grouped by common origin AS, and prefixes grouped by common routing attributes.

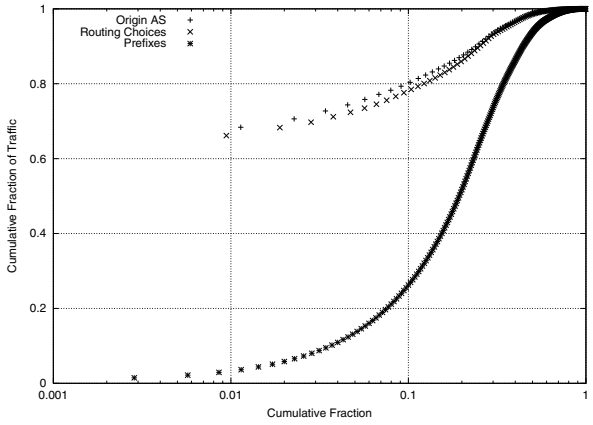

Figure 13: Cumulative distribution of traffic for one next-hop AS at one router. A small number of groups of prefixes with common routing advertisements are responsible for the majority of outbound traffic.

when we group prefixes with the same routing choices, as shown by the middle curve in Figure 12. For example, 10% of the sets of routing choices contribute more than 80% of the traffic. Grouping traffic by origin AS—the AS that originates the BGP announcement and receives the traffic—produces similar results, as shown by the top curve. The top 10% of origin ASes are responsible for approximately 82% of the outbound traffic.

By focusing on the small fraction of prefixes that carry the majority of the traffic, an operator can manipulate a large volume of traffic with a small number of routing changes. For example, an operator who wishes to reduce the load on an outgoing link might assign a smaller local preference value to the route advertisements associated with one or more popular prefixes at that router, thus shifting traffic destined for these prefixes to a different egress point. That is, each ingress point that is sending traffic to these destinations prefixes via this outgoing link would start sending the traffic via the next closest egress point with a "best" route. Rather than moving traffic for individual prefixes, the import policy modifications based on route advertisement attributes can move the traffic associated with popular *groups* of related prefixes. Figure 13 shows the distribution of traffic for a single egress point (a particular next-hop AS at one router). Compared to Figure 12, the bottom curve in Figure 13 shows a more even distribution across the destination prefixes, since each egress point carries traffic destined to some subset of prefixes. Nevertheless, the top curves show that a few *groups* of prefixes carry most of the traffic.

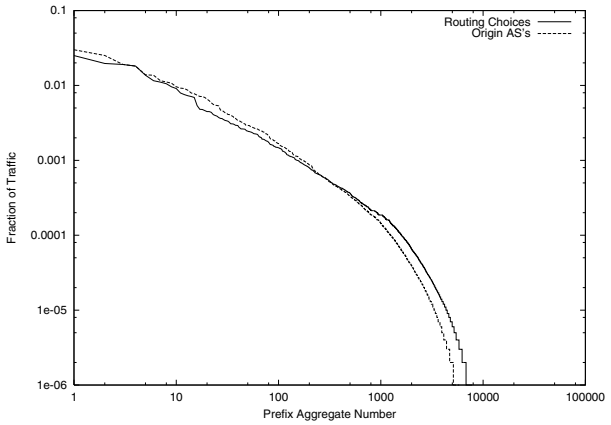

Figure 14: Proportion of traffic for each group of prefixes for one next-hop AS at one router. Each group of prefixes carries a different proportion of total outbound traffic at that router, providing network operators flexibility in shifting arbitrary amounts of traffic.

A relatively simple change in import policy can move a significant amount of traffic to or from a particular egress link. However, the appropriate amount of traffic to move may depend on the current link loads. Typically, an operator selects a set of prefixes to shift based on the current traffic distribution. Figure 14 shows the proportion of traffic traversing a particular egress point associated with each origin AS and each set of unique routing choices. Knowledge about traffic distributions for each origin AS and each set of prefixes with common routing attributes allows the operator to identify groups of prefixes associated with a certain proportion of the traffic and devise changes to import policy that manipulate an appropriate traffic volume.

6.3 Move Stable Traffic Volumes

Section 6.2 describes how to shift traffic by making import policy changes that affect the routes taken to groups of destination prefixes. The effects of these types of changes depend on the volume of traffic traveling to the destination associated with these routes. Even if the aggregate utilization of the link is relatively stable, the contribution of individual prefixes or origin ASes can be highly variable. Figure 15 shows the cumulative distribution of origin ASes experiencing a particular change in traffic between April 1, 2002 and April 8, 2002. The bottom curve shows the results for all origin ASes. For example, the point $(1, 0.7)$ on the lower line indicates that 70% of all origin ASes experience less than a 100% fluctuation in traffic from week-to-week; the remaining 30% of the origin ASes experience *more* fluctuation. This amount of variation would make it difficult to use traffic measurements from one day to drive traffic engineering decisions on another day. In particular, the prediction tools described in Section 2.2 would not make accurate estimations about how much traffic would be affected by changes in import policies.

Fortunately, by focusing on the groups of prefixes that carry significant portions of traffic, a network operator can make the effects of BGP policy changes more predictable. This is illustrated in the

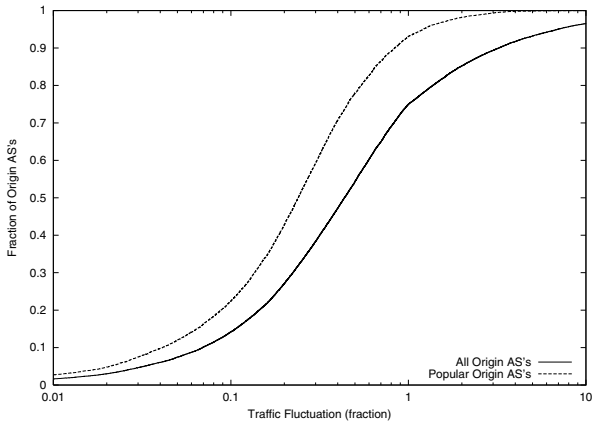

Figure 15: Cumulative distribution of origin ASes experiencing a particular fraction change in traffic from week-to-week. The graph shows this characteristic both for all origin ASes and for "popular" origin ASes—those that receive at least 0.01% of the total outbound traffic. Popular origin ASes tend to be more stable: only 20% of all popular origin ASes experienced more than a 50% traffic fluctuation from week to week, even though 45% of all origin ASes experienced such a fluctuation.

top curve in Figure 15, which focuses on the origin ASes that receive at least 0.01% of the total traffic ("popular" origin ASes). The graph shows that popular origin ASes tend to have more stable traffic volumes: only 20% of all popular origin ASes experienced more than a 50% traffic fluctuation from week to week, even though 45% of all origin ASes experienced such a fluctuation. Graphs for other pairs of days one week apart show similar trends, which are consistent with earlier studies that show that aggregation results in more stable traffic loads over time [25]. Thus, network operators should focus their attention on changing routes for prefixes and groups of prefixes that are responsible for larger fractions of the traffic. Fine tuning by moving small amounts of traffic may prove rather difficult in practice.

Nevertheless, the degree of stability varies across the popular destinations—certain destinations have remarkably stable traffic volumes, as shown by the left portion of the two curves in Figure 15. Just over 7% of all origin ASes have a traffic volume that fluctuates less than 5% between the two dates. This amount of fluctuation is arguably small enough to enable the traffic engineering tools to make accurate predictions of the volume of traffic that would move from one route to another. Tracking measurement data over time would allow an operator to identify the specific prefixes (and groups of prefixes) with relatively stable traffic volumes. The operator can focus on routes for these destinations when trying to move traffic from one link to another.

7. Conclusion and Future Work

BGP is a flexible interdomain routing protocol that scales to the large number of ASes in today's Internet. However, BGP was not designed with traffic engineering in mind. The attributes available in BGP advertisements, the restrictions in the BGP decision process, and the constraints imposed by configuration languages all limit an operator's ability to tune routing policies to the prevailing traffic patterns. A network operator can achieve certain traffic engineering goals by making changes to the BGP import policies running on its routers. Using BGP policies to shift traffic requires extreme care: changes should result in predictable and stable changes in traffic flow and minimize the possibility of affecting inbound traffic volumes. In particular, our analysis suggests that operators should make policy changes based on large groups of prefixes (e.g., groups of prefixes that have a common origin AS, or other common attributes), limit policy sensitivity to AS path changes by assigning policies based on AS path regular expression matches, and assign local preference based on ranges of AS path lengths, rather than using AS path length as an absolute metric.

The techniques we suggest can be used together to solve real traffic engineering problems. For example, suppose an operator realizes (say, via SNMP data) that a particular edge link is congested. First, fine-grain measurement data (such as Netflow) can be used to identify the destination prefixes responsible for the bulk of the traffic traversing this link; historical measurement data could be used to determine which of these prefixes have stable traffic volumes. Next, the operator could analyze the routing data to focus on the popular, stable prefixes that have a single "best" AS path across all of the egress points. Then, the operator could consider modifying the import policy at the congested router to assign a lower local preference to some of these destination prefixes to divert this traffic to the other egress links. Rather than assigning local preference directly to each prefix, the operator could inspect the routing data to select a suitable regular expression on the AS path attribute. Finally, the operator could test this policy using the prediction tool in Figure 2 to check how the proposed change would affect the flow of traffic in the network. In fact, ultimately, the traffic engineering tools could evolve to automate many of these steps by identifying specific destination prefixes and import policy changes for the operator.

Interdomain traffic engineering using BGP policies presents many interesting avenues for future work:

- *Traffic stability:* The amount of traffic traveling to each destination prefix varies over time. Effective traffic engineering relies on understanding how traffic stability varies with the level of aggregation and over time. Section 6.3 makes a few initial observations about the stability of traffic volumes for prefixes and groups of prefixes, but a better understanding about traffic stability could enable operators to make traffic engineering changes with higher confidence. The notions of "operational constancy" and "predictive constancy" [26] may be helpful in identifying which kinds of fluctuations in traffic volume might affect traffic engineering decisions.

- *Inbound traffic:* In this paper, we have focused on the influence of BGP import policies on *outbound* traffic; however, a complete solution should consider inbound traffic as well. Since an operator has limited control over how traffic enters the network (using crude techniques such as AS prepending), we believe that neighboring ASes should coordinate to gain a greater level of predictability with respect to how traffic enters each network. We are considering ways for neighboring ASes to cooperate without revealing their network topologies and routing policies [27].

- *Performance objective:* Traffic engineering involves tuning routing policies based on a target performance objective. The commercial relationships between ASes impose constraints and costs based on the volume of traffic exchanged with neighboring domains. In addition, the distribution of traffic after network failures may also play a role in evaluating possible changes to the routing configuration. Drawing on earlier work on IGP optimization, our ongoing work considers new objective functions that capture the constraints of both in-

tradomain and interdomain routing, including the influence of peering agreements.

- *End-to-end performance:* Changes in BGP policies affect the *end-to-end* path from a source to a destination which, in turn, influences performance. We are investigating ways to collect information about the performance properties of the rest of the path to help weigh the benefits of different changes in BGP policies and IGP weights. For example, active measurements that identify congestion problems in other ASes would lend insight into which policy changes would improve end-to-end performance.

These ongoing research efforts can draw on and extend the insights from the analysis of routing and traffic data we have presented.

Appendix

In this appendix, we discuss lower-level details related to the configuration of import policies on BGP-speaking routers. First, we describe configuration options that operators should enable to make the BGP decision process deterministic and reduce the overhead of making changes in import policies. Then, we discuss the influence of other BGP attributes (besides local preference and AS path) on the decision process.

A. Router Configuration Options

The traffic engineering framework in Figure 2 of Section 2.2 depends on the ability to predict how BGP import policies affect the selection of the best route. We discuss configuration options that an operator should enable to ensure that the BGP decision process has a deterministic outcome. Then, we describe other configuration options that enable network operators to modify an import policy without resetting the BGP session.

A.1 Deterministic BGP Decision Process

Some router vendors have an additional step in the BGP decision process that occurs between the "lowest IGP metric" and the final "lowest router ID" steps. This additional step prefers the "oldest" route—the route that was received the earliest among the ones still in consideration. Including this step has the desirable effect of favoring the more stable routes over the routes that change frequently. However, this makes the outcome of the BGP decision process dependent on the *order* the router receives the advertisements, making it impossible to predict the selection of the best route from a static snapshot of the routing choices. Disabling age-based tie-breaking forces a deterministic selection based on the smallest router ID. Other BGP features, such as route flap damping [28], can help reduce the likelihood of selecting unstable routes.

The MED attribute is another potential source of non-determinism. As discussed above, the comparison of MED values applies only to routes learned from the same next-hop AS. As a result, the comparison between routes is not necessarily transitive—route r_1 being "better" than route r_2 and route r_2 being "better" than r_3 does *not* necessarily imply that route r_1 is "better" than r_3. This can make the selection of the best path dependent on the *order* of the comparison between paths, as illustrated by a detailed example in [29]. Router vendors recommend enabling the "bgp deterministic-med" option for deterministic path selection in the presence of MEDs.

A.2 Avoiding BGP Session Resets

A router applies the import policy to filter and manipulate BGP advertisements as they arrive from a BGP neighbor, as part of constructing the Routing Information Base (RIB). After a change in the import policy, the router needs to apply the new import policy to the existing routes learned from the BGP neighbor. However, the RIB only stores the routes as they appear *after* import processing under the *old* policy, and the old import policy may have filtered some routes and manipulated the attributes of others. Applying a new import policy could conceivably require the router to *reset* the session with the BGP neighbor in order to receive a fresh copy of each advertisement. This introduces substantial overhead on both routers and causes temporary routing instability that could spread to other parts of the Internet.

To avoid this problem, operators can configure their routers to store a local copy of each received advertisement. Enabling the "soft-reconfiguration" feature on inbound routes allows the router to apply the new import policy without disrupting the BGP session with the neighbor [30]. Enabling soft reconfiguration has the additional advantage of allowing operators to inspect or dump a copy of the received routes (e.g., using the "show ip bgp received-routes" command on a Cisco router). Dumping the received routes is useful for diagnosing routing problems and provides a more complete view of the routing choices learned from neighboring domains than the RIB does. However, the soft-reconfiguration feature has the disadvantage of consuming additional memory on the router. The relatively new "route refresh" option [31] in BGP is a viable alternative, if the neighbor's router supports it. This feature allows a router to signal a BGP neighbor to send a fresh copy of each advertisement without resetting the BGP session.

B. BGP Attributes and the Decision Process

To simplify the discussion, Section 2.1 presented a view of the BGP decision process that omitted the influence of two BGP attributes, origin type and multi-exit discriminator (MED). The origin type identifies how the origin AS learned about the route—within the AS (e.g., static configuration), EGP (a now-defunct distance-vector protocol), or injection from another routing protocol. These origin types are known as IGP, EGP, and INCOMPLETE. After considering AS path length, the BGP decision process prefers IGP routes over EGP routes, and EGP routes over INCOMPLETE. The MED attribute is an integer value set by an eBGP neighbor to encourage the recipient to pick a particular egress point for traffic. After considering the origin type and before considering "eBGP vs. iBGP," the decision process selects routes with the lowest MED value. The default behavior in most routers is to compare MED values only across routes with the same next-hop AS.

We have focused on how local preference assignment influences the first stage in the BGP decision process. After local preference, AS path length influences the selection of the "best" routes. Because origin type and MED affect the next two stages of the decision process, these attributes may *remove* some of the routes from consideration, reducing the set of "best" routes. In some cases, an eBGP neighbor may require the AS to accept these attributes as they appear in the advertisement messages. For example, a neighbor may use the MED attribute to override "hot potato" routing, where an AS can select the "closest" egress point based on the IGP path costs. Two AS's may have an agreement in advance to send and accept the MEDs. Alternatively, an operator may choose to ignore these two attributes by *resetting* their values in the import policy. Operators sometimes choose to reset the origin type and MED values to prevent an eBGP neighbor from using these attributes to affect the outcome of the decision process.

Alternatively, operators can *reassign* origin type or MED values in the import policy to influence route selection. This complements the influence of local preference on the decision process by giving an operator control *after* the selection of the routes with the shortest

AS path. For example, if an AS has multiple BGP sessions with a neighboring domain and wants to shed some traffic from an egress point, an operator can assign a higher MED value (or less preferable origin type) to some prefixes at this egress point. An operator can also override the default behavior of limiting MED comparison to routes with the same next-hop AS. For example, Cisco IOS has a "always-compare-MED" command that causes the BGP decision process to compare MED values across all routes, irrespective of the next-hop AS. For all of these techniques for configuring import policies, network operators can draw on the insights in the main body of our paper to decide *which* traffic they should move between egress points.

Network operators can employ a variety of other techniques to influence the decision process. Section 5.1 described how a neighboring domain might employ AS prepending in the export policy to inflate the AS path length. An operator can use this technique in the *import* policy to influence the selection of best routes, effectively eliminating some routes in the "AS path length" step. Operators might also use the BGP community attribute to "program" a wide variety of policies for path selection. A community is an opaque string that is assigned to a route by an import or export policy. The import policy could tag a route with a community string to label whether the route was learned from a peer or a customer, or based on the geographic location. A network operator can use these tags to affect the assignment of other BGP attributes or the decision of whether to export the route to certain neighboring AS's. For example, an operator could use the tags to instruct routers in Europe to assign lower local preference values to routes learned in the United States in order to minimize the use of slow (and often expensive) transatlantic links.

Acknowledgments

We thank Tim Griffin for many very helpful discussions and Carsten Lund for providing the aggregated Netflow data. Thanks also to Dave Andersen, Hari Balakrishnan, Randy Bush, Steve Garland, Joel Gottlieb, Jaeyeon Jung, Carsten Lund, Aman Shaikh, Alex Snoeren, Iljitsch van Beijnum, Jia Wang, and the anonymous reviewers for very helpful feedback.

C. References

[1] D. O. Awduche, A. Chiu, A. Elwalid, I. Widjaja, and X. Xiao, "Overview and principles of Internet traffic engineering." Request for Comments 3272, May 2002.

[2] D. O. Awduche, J. Malcolm, J. Agogbua, M. O'Dell, and J. McManus, "Requirements for traffic engineering over MPLS." Request for Comments 2702, September 1999.

[3] D. O. Awduche, "MPLS and traffic engineering in IP networks," *IEEE Communication Magazine*, pp. 42–47, December 1999.

[4] A. Feldmann, A. Greenberg, C. Lund, N. Reingold, and J. Rexford, "NetScope: Traffic engineering for IP networks," *IEEE Network Magazine*, pp. 11–19, March 2000.

[5] B. Quoitin, S. Uhlig, C. Pelsser, L. Swinnen, and O. Bonaventure, "Interdomain traffic engineering with BGP," *IEEE Communications Magazine*, pp. 122–128, May 2003.

[6] J. Wepman and J. Abley, "Inter-domain Traffic Engineering: Principles, Applications, and Case Studies." `http://www.nanog.org/mtg-0202/te.html`, February 2002. Tutorial at NANOG 24. Miami, FL.

[7] Z. Kerravala, N. Maynard, and A. Phull, "Intelligent routing: The high IQ Internet," July 2002. The Yankee Group. `http://www.sockeye.com/pdf/Yankee_IntellRouting_July02.pdf`.

[8] Y. Rekhter and T. Li, "A Border Gateway Protocol." Request for Comments 1771, March 1995.

[9] S. Halabi and D. McPherson, *Internet Routing Architectures*. Cisco Press, second ed., 2001.

[10] J. W. Stewart, *BGP4: Inter-Domain Routing in the Internet.* Addison-Wesley, 1998.

[11] G. Huston, "Interconnection, peering, and settlements," in *Proc. INET*, June 1999.

[12] "BGP Best Path Selection Algorithm." `http://www.cisco.com/warp/public/459/25.shtml`.

[13] "How the Active Route Is Determined." `http://arachne3.juniper.net/techpubs/software/junos42/swconfig-routing42/html/protocols-overview4.html#1045417`.

[14] "Foundry Switch and Router Installation and Configuration Guide, Chapter 19, Configuring BGP4." `http://www.foundrynet.com/services/documentation/SRguide/FoundryManual_BGP4.html`.

[15] C. Labovitz, A. Ahuja, and F. Jahanian, "Experimental study of Internet stability and wide-area network failures," in *Proc. International Symposium on Fault-Tolerant Computing*, June 1999.

[16] J. Rexford, J. Wang, Z. Xiao, and Y. Zhang, "BGP routing stability of popular destinations," in *Proc. Internet Measurement Workshop*, November 2002.

[17] N. Feamster and J. Rexford, "Network-wide BGP route prediction for traffic engineering," in *Proc. Workshop on Scalability and Traffic Control in IP Networks, SPIE ITCOM Conference*, August 2002.

[18] A. Feldmann, A. Greenberg, C. Lund, N. Reingold, J. Rexford, and F. True, "Deriving traffic demands for operational IP networks: Methodology and experience," *IEEE/ACM Trans. Networking*, vol. 9, June 2001.

[19] "Cisco Netflow." `http://www.cisco.com/warp/public/732/netflow/index.html`.

[20] "Sampled Netflow." `http://www.cisco.com/univercd/cc/td/doc/product/software/ios120/120newft/120limit/120s/120s11/12s_sanf.htm`.

[21] N. Duffield, C. Lund, and M. Thorup, "Charging from sampled network usage," in *Proc. Internet Measurement Workshop*, November 2001.

[22] A. Broido and K. Claffy, "Analysis of RouteViews BGP data: Policy atoms," in *Workshop on Network-Related Data Management*, May 2001.

[23] T. Bu, L. Gao, and D. Towsley, "On Characterizing BGP Routing Table Growth," in *Proc. IEEE Global Internet*, (Taipei, Taiwan), November 2002.

[24] W. Fang and L. Peterson, "Inter-AS traffic patterns and their implications," in *Proc. IEEE Global Internet Symposium*, December 1999.

[25] N. Taft, S. Bhattacharyya, J. Jetcheva, and C. Diot, "Understanding traffic dynamics at a backbone POP," in *Proc. Workshop on Scalability and Traffic Control in IP Networks, SPIE ITCOM Conference*, August 2001.

[26] Y. Zhang, N. Duffield, V. Paxson, and S. Shenker, "On the constancy of Internet path properties," in *Proc. Internet Measurement Workshop*, November 2001.

[27] J. Winick, S. Jamin, and J. Rexford, "Traffic engineering between neighboring domains." `http://www.research.att.com/~jrex/papers/interAS.pdf`, July 2002.

[28] C. Villamizar, R. Chandra, and R. Govindan, "BGP Route Flap Damping." Request for Comments 2439, November 1998.

[29] "How BGP Routers Use the Multi-Exit Discriminator for Best Path Selection." `http://www.cisco.com/warp/public/459/37.html`.

[30] "BGP Soft Reset Enhancement." `http://www.cisco.com/univercd/cc/td/doc/product/software/ios120/120newft/120t/120t7/sftrst.htm`.

[31] E. Chen, "Route refresh capability for BGP-4." Request for Comments 2918, September 2000.

Link Layer–Based TCP Optimisation for Disconnecting Networks

James Scott *
Intel Research Cambridge
15 JJ Thomson Avenue
Cambridge CB3 0FD, UK
james.w.scott@intel.com

Glenford Mapp †
Middlesex University
Bounds Green Road
London N11 2NQ
g.mapp@mdx.ac.uk

ABSTRACT

This paper discusses a link layer approach to improving TCP performance in the face of periodic network disconnections. Network disconnections are encountered in many scenarios, including being out-of-range in a wireless network, during network handoff, and also in the case of Networked Surfaces, a novel LAN technology which provides the motivation for this work.

A "smart link layer" employing repetition of selected packets at reconnection time is shown to improve TCP's utilisation of a disconnecting network to nearly 100%. This solution is also demonstrated in the context of a Networked Surface prototype, improving TCP performance for both bulk transfers and interactive traffic.

The smart link layer solution is lightweight, requiring little processing and buffering only one packet per TCP connection. It is therefore easily retro-fitted to existing TCP-capable devices, without modifying the internal operation of those devices.

Keywords

TCP, Disconnection, Link Layer, Mobile Networking

1. INTRODUCTION

The Internet Protocol Suite, in particular TCP/IP, has been a runaway success. However, these protocols were conceived when all data communications was carried over wired links. Those days are long gone and new environments such as mobile telephony and Wireless LANs are now becoming ubiquitous.

In these new settings some of the original design assumptions of TCP no longer hold true with respect to the handling of errors such as lost packets. This unfortunately results in a large performance degradation in these environments. This is because TCP assumes all packet loss is due to network congestion, whereas in these settings it may be due to a number of factors including momentarily high link error rates and handoffs of mobile devices between adjacent base stations.

By assuming network congestion is the cause of these errors, TCP does the wrong thing: it drastically reduces its transmit window and deploys the slow start algorithm. This results in unutilised network bandwidth and applications experiencing increased network latency. This behaviour has also been observed in networks where devices can be disconnected, even when the disconnected interval is near the human threshold of noticeability (under 1 second), as well as longer durations (e.g. changing a network cable; 1 minute). Part of the motivation for this work was the development of a novel LAN technology called Networked Surfaces [21] which exhibits disconnections; details of this technology are discussed further in Section 3. Such disconnections may also be found when using wireless networking with signal fading (e.g. due to being on the limit of the range available), in wireless handoff scenarios, or in other situations where the network access path changes (e.g. when a device is removed from a wired docking station and starts to use a wireless network).

1.1 Related Work

Past attempts to address this problem can be neatly divided into two distinct camps.

The first group does not attempt to change or modify the TCP protocol, instead using methods such as injecting, removing or delaying TCP packets based on a superior understanding about what is happening at the link layer. Snoop [6] looks at this problem in the context of a wireless lossy link on the periphery of a wired network. It requires that base stations have large memory and processing power to store network packets while handoffs take place. The base station also gives local acks and suppresses duplicate acks. The "Delayed Dupacks" scheme [23] looks at the same problem but in the context of a reliable link layer protocol which acknowledges each packet and performs fast retransmission. The system allows the TCP receiver to delay acks by a set amount and does not send them at all if a new packet arrives prior to the timeout. TULIP (Transport Unaware Link Improvement Layer) [18] also attempts to recover from retransmission losses before TCP coarse-grain timeouts occur. AIRMAIL (Asymmetric Reliable Mobile Access In Link Layer) [2] uses similar ideas.

The second group focusses on modifying how TCP works. Caceres and Iftode [8] were one of the first to examine the problem

*Much of this work was done whilst at the Laboratory for Communication Engineering, which is part of the University of Cambridge.
†Much of this work was done whilst at AT&T Laboratories Cambridge.

of disconnected periods affecting TCP, which they found to occur during mobile handoff. They propose a system augmenting TCP so that, on reconnection, a mobile host would retransmit a number of duplicate acks, and so that a fast retransmit mode is automatically entered. Other research focuses on the use of proxies to try to isolate the effects of disconnection to a single link. I-TCP [3] does TCP proxying at such a gateway and modification must be made to the non-ideal segment (meaning the wireless side) to improve performance. The disadvantage is that end-to-end TCP semantics are not upheld, in that acks are sent for data which has not actually reached the final endpoint. M-TCP [7] also uses a proxy approach but maintains end-to-end semantics. It does so by using delayed acks and by placing TCP in persist mode to avoid losing packets during handoffs.

Other methods of modifying TCP behaviour use "flags" or control messages to trigger appropriate responses from TCP. Schemes based on this general theme including Explicit Loss Notification (ELN) [5], Explicit Bad State Notification (EBSN) [4], Explicit Link Failure Notification (ELFN) [13], Route Failure Notification (RFN) [9], as well as an ICMP-based solution [12].

It is also appropriate to summarise the recent efforts of standards bodies, in particular the IETF, with regard to this problem. The Performance Implications of Link Characteristics (PILC) Working Group is looking at how the IP Protocol Suite works with different types of link layers. The latest draft document from this group [16] attempts to characterise links and come up with best-practice suggestions for system administrators. The disconnection problem, described by that group as recovery from subnetwork outages, is addressed by recommending that packets are not discarded during an outage and an interface (such as those described above) be provided to allow IP and the higher layers to be notified once the link has been restored. If this is not feasible, it is recommended that the link layer retains one or more of the packets which could not be transmitted during the disconnected period, and retransmits these packets on reconnection. A similar approach was decided upon when the research presented in this paper was begun, in March 2000, and experimental results concerning the performance of this solution are presented below.

Other recent work in the IETF includes the development of the TRIGTRAN framework [10]. This proposal, like the PILC one, involves a mechanism to alert the transport layer about changes in individual links along the network path from source to destination. Under this scheme, hosts may request notification when trigger events such as Connectivity Interrupted, Connectivity Restored and Packets Discarded by Subnet occur. However, this work is very recent and is currently lacking in experimental support.

1.2 Paper Structure

This paper describes the implementation and experimental evaluation of a "smart link layer" to solve the problem of TCP performance degradation during disconnected periods, as motivated by the introduction of a new type of LAN named Networked Surfaces, which is prone to disconnections. Section 2 presents the TCP performance degradation problem in detail. Section 3 then describes Networked Surfaces and presents the motivation behind the work as a whole, as well as the impetus for making the specific design choices present in the smart link layer. Section 4 presents a discussion of the design space for the smart link layer approach, and identifies a number of candidate algorithms for experimental evaluation. Section 5 evaluates these algorithms on a testbed using a simulated channel and analyses the performance characteristics of the algorithms, and Section 6 goes on to determine the effectiveness of the best-performing algorithm on the Networked Surface platform, under both bulk transfer and interactive traffic patterns. Finally, Section 7 compares the techniques presented in this paper with those in the literature, and Section 8 concludes the paper, including a discussion of future work.

2. THE EFFECT OF DISCONNECTION ON TCP

TCP regards all packet losses as indications of congestion. While working well for wired infrastructure, this has caused many problems when combined with wireless access, in which channel errors causing dropped packets are more common. Distinguishing and coping with such losses in order to make TCP fully utilise a lossy channel has been the subject of much research, as described in the previous section.

However, there are many differences between disconnection and lossy channels, which means that the same solutions may not work well for both cases. Firstly, in the lossy channel case, it is obvious that discovering the loss and retransmitting quickly is desirable. However, with disconnection, retransmissions are not useful until the link is re-established. On the contrary, transmitting and retransmitting packets for a disconnected device is guaranteed to be a waste of network bandwidth.

Secondly, the timescales for channel losses and disconnections are very different, with the former operating on a packet-by-packet basis and in the microseconds range, while disconnections may last anywhere between milliseconds and minutes, depending on the cause. A short disconnection may be due to a Mobile IP handoff occurring, while a longer disconnection may be due to a modem connection failing and having to be reconnected. Disconnections typically last longer than a TCP timeout, while potentially being shorter than the lifetime of a TCP connection.

In order to illustrate the detailed effects of disconnection on a running TCP connection, a simple experiment was conducted in which a file was transferred over a disconnecting link. The TCP "trace" occurring in this experiment is shown in Figure 1. As the figure shows, the sender does not react to disconnection, and continues sending (pointlessly) until its window is full. It then waits for acks, but times out before any ack arrives and retransmits the first packet. Retransmission occurs two more times, with an increasing timeout period each time — this is because TCP assumes that the lack of response is due to the network being congested, and so tries to back off to let the network recover. When reconnection occurs, TCP does not immediately restart, instead continuing to wait until its next timeout. When this happens, the packet gets through, causing an ack to be received and further packet transmission to resume. Note that over 1.5s of connected time was wasted by TCP in this case.

3. NETWORKED SURFACES

The motivation behind this work is the introduction of a new type of network, known as Networked Surfaces [21]. This network is based on the use of physical surfaces such as desks to perform networking. Devices such as laptop computers and PDAs can acquire network connectivity by simply being placed on top of such a surface, in any position and at any orientation. Networked Surfaces can also provide power to devices such as mobile phones, they can

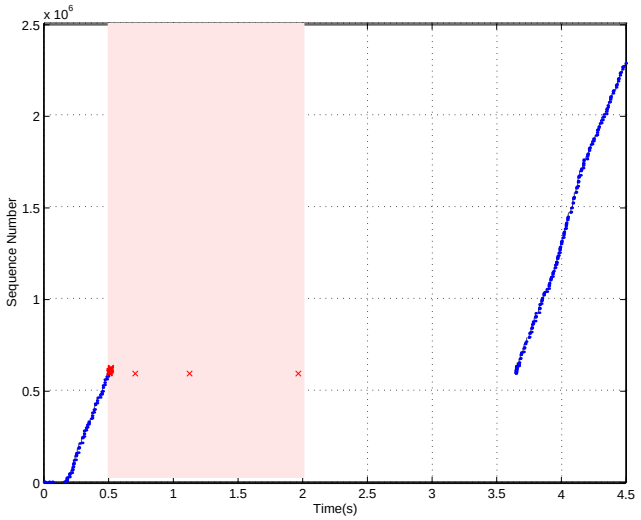

Figure 1: TCP File Transfer with Disconnection

TCP traces in this paper are presented as follows.

The vertical lines are transmitted segments, the dots are acks returning. In zoomed-out plots (such as the one above), these are hard to distinguish individually, and appear as sloped line.

The shaded portions are periods of disconnection. Segments dropped during these disconnected periods are shown with crosses.

Segments and acks inserted by the smart link layer (to be described) on reconnection are shown with circles and plus marks, respectively.

support low-speed devices such as keyboards and sensors, and they can locate devices to within a few centimetres and a few degrees.

The vision behind Networked Surfaces is that they provide the best of both worlds between wired and wireless paradigms. As with wired devices, networking is provided at a high bandwidth (5Mbit/s in the first prototype) and does not have to be shared with other devices in a physical space, and electrical power is provided. At the same time, the inconvenience and hassle of carrying and connecting cables is avoided, thus providing a very user-friendly environment for mobile computing users.

Networked Surfaces operate by using electrically conductive pads on the surface and the base of the device. When a device comes into contact with a surface, a handshaking procedure causes the various conducting paths formed to be assigned to functions such as ground, power, and networking buses. Disconnection is detected by the custom link layer protocol used, and when it occurs, the connected pads are returned to a disconnected state, ready for a new connection. It is important to note that Networked Surface NIC functionality is intended to be added to existing devices, including devices which are not reprogrammable, and which therefore have hardcoded TCP/IP stacks. Even when devices are reprogrammable, e.g. with notebook computers, it is not normally expected that installation of a NIC's software driver would entail modification of the TCP/IP stack of the device.

One key issue in the usability of Networked Surfaces is the fact that devices are susceptible to occasional disconnections, since any movement of a device may cause the connected pads to lose contact. When this happens, the pads must all undergo disconnection and then re-execute the handshaking protocol before data transfer can resume, a process typically taking between 200ms and 500ms. Movement may happen because a user is actively operating the device (e.g. typing); they may therefore be directly inconvenienced by the lack of connectivity (e.g. typing into a remote terminal). The optimisation of TCP performance in these circumstances is therefore important to the usefulness of Networked Surfaces. Note that Networked Surfaces do *not* suffer from high bit error rates; the 5Mbit/s prototype network offers a bit error rate of 10^{-10}. They also do not suffer from link-layer packet loss, as the link layer protocol used avoids losses due to collisions. For more information see [20].

In considering the implementation of a solution to this problem, a number of constraining factors are noted. Firstly, the solution must be applicable to a variety of devices using Networked Surfaces, which, as stated above, may not be internally modifiable. Even when programmable, the device may have limited CPU, memory and/or battery life, so minimal use of resources is important. The second factor to be considered is that the solution must also run on the device acting as IP-level gateway between the Networked Surface and other networks. A solution which demands high per-connection processing and memory requirements may therefore limit scalability of a Networked Surface to supporting many devices, and should be avoided. Finally, it is important to be able to communicate with unmodified corresponding hosts, as otherwise the solution would be impractical for reasons of deployability.

These factors (in particular the first two) dictate that a solution operating externally to TCP/IP is required. Such a solution, based in the link layer and known henceforth as the "smart link layer," is discussed in the next section, where the requirements above will have a strong role in guiding the design decisions made. Further discussion on advantages and disadvantages of a link layer approach when compared to solutions modifying TCP may be found in Section 7.

4. DESIGN OF THE SMART LINK LAYER

The "smart link layer" augments a traditional link layer design with limited awareness of transport-layer functionality, so that methods can be applied in order that TCP connections which have stalled during a disconnected period are promptly "kick-started" on reconnection, i.e. the flow of data is promptly resumed.

To illustrate the placement of the smart link layer in the network, a network connection involving a disconnecting link is depicted in Figure 2. This diagram shows the general case in which the disconnection is occurring on an unspecified link somewhere in the network path between the end-to-end TCP connection. In practice, it is expected that most disconnections will occur in the edge links, e.g. on the access network for a mobile device, such as a Networked Surface–enabled PDA. It is also possible that disconnections may be present on more than one link of the network, for example during peer-to-peer transmissions between two Networked Surface devices.

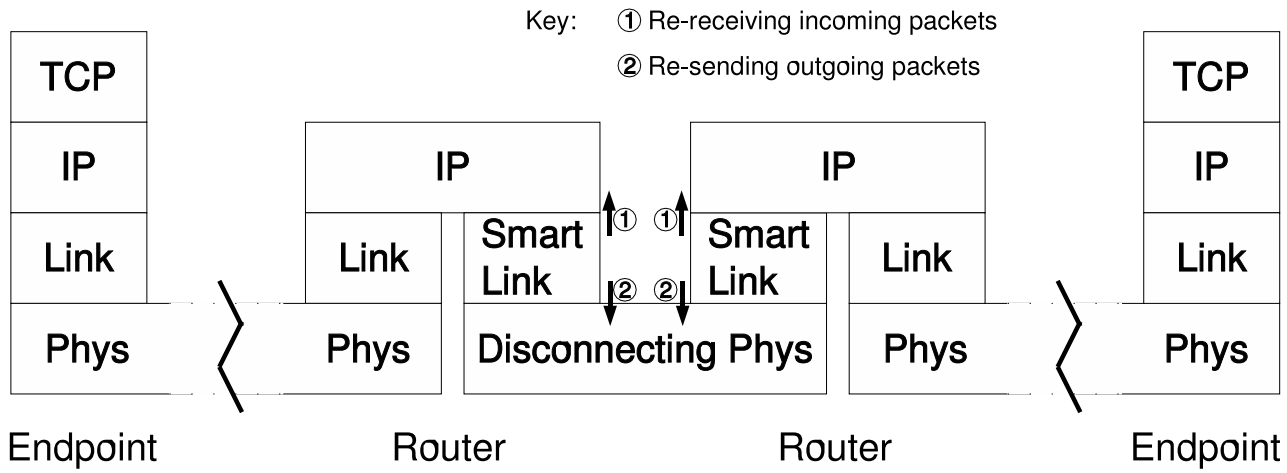

Figure 2: Network Connection including Disconnecting Link

For the purposes of the smart link layer solution, the key locations in the network path are at either end of the disconnecting link; these are represented as "Smart Link" in Figure 2. The smart link layer operating at these points may modify the network traffic in a number of ways. Network packets in transit for a given connection may be recorded, dropped or modified (the latter being more computationally expensive than the other two as checksums must be recalculated). Packets may also be inserted at these points in one of two ways; as shown in the diagram, inserted packets may either be placed in the outgoing queue for the disconnecting link (denoted hereon as "re-sending" packets), or in the incoming queue as if they had just been received on that link (denoted hereon as "re-receiving" packets).

One key advantage of a smart link layer solution is that disconnection and reconnection of the link may be automatically detected at these points and may be used to trigger events. This is not true, in general, for an end-to-end protocol, which may not be able to easily determine link states for individual links on the network path used. In order for the smart link layer to force a reaction from the end-to-end TCP engines at reconnection time, the obvious method is to insert one or more packets into the network at the moment when reconnection is detected.

The design space for solutions involving link layer insertion of packets on reconnection is discussed below. The priority motivating the particular designs discussed is the minimal use of resources such as processing and memory, for the reasons described in the previous section.

4.1 Parameters for Packet Insertion

Irrespective of whether inserted packets are re-sent or re-received, there are a number of other issues to be solved. In particular, there is the issue of how the inserted packets are constructed, and the issue of how many packets are inserted.

For the issue of the construction of inserted packets, there are two possibilities. These packets may either be copies of packets that have already passed through the network (and were recorded by the link layer), or they may be constructed afresh by the link layer, presumably using information gathered from monitoring previous traffic. While the latter approach gives the maximum flexibility to the smart link layer, allowing it to choose precisely the content of its inserted packets to potentially force the quickest TCP recovery, it is also resource-intensive in that the inserted packets must be constructed and checksummed by the link layer itself.

In contrast, although the use of pre-transmitted packets provides less flexibility, it is also much simpler to implement. Relatively little "knowledge" of TCP/IP is required to be duplicated at the link layer, and the computation required to construct valid TCP/IP packets is avoided. In addition, the use of copying assumes less about the particular TCP implementation being used, in that any unknown options or parameters present in packets are simply passed on without modification. For these reasons, this research focuses on the use of copied packets retransmitted at reconnection time.

The next consideration is the number of packets that should be repeated on reconnection. While it is possible to have a policy such as link layer retransmission of all unacknowledged data on reconnection, this would require a large amount of buffer space, may result in a large waste of bandwidth (as packets may be retransmitted needlessly), and may interfere with TCP's own retransmissions. For these reasons, only one packet per TCP connection is buffered. Under this policy, a quick "back of the envelope" calculation shows that the overhead is not burdensome; an access router attached to a disconnecting link would only need 3kb of storage per TCP connection; with 100 edge nodes each using 10 active TCP connections, this would result in a requirement of 3Mb of extra RAM to implement smart link layer functionality.

The one-packet policy still leaves the possibility for duplicates of the buffered packet to be inserted on reconnection. Duplicate packets may be useful as they can cause the receiving TCP state machine to be forced into a fast retransmit mode [15], since the packets will cause duplicate acks to be received. The benefit of repeating packets is therefore examined experimentally in this research.

4.2 Re-receiving Packets

Re-receiving means that the packets are inserted into the incoming queues of the hosts on either side of the disconnecting link. While such packets will not traverse the disconnecting link, they may be

transmitted across other parts of the end-to-end network. The advantage of this approach is that, in the likely case that the disconnecting link is at the periphery of the network, one of the TCP correspondents will receive its kick-start very quickly, as no network latency will be incurred. A key disadvantage of re-reception is that such packets are by definition never going to provide the receiving TCP engines with any data they have not seen before. These packets are therefore confined to repeating old data, old acks, and/or old window advertisements.

The choice of data to re-receive is determined by that data which is most likely to cause TCP to immediately send out more data, and initiate recovery mechanisms to re-establish data flow as quickly as possible. Since these mechanisms are governed largely by the reception of acks, the best information to re-receive is obviously the highest ack already received. To achieve this, it suffices to compare each packet passing through the link layer with the packet in the buffer, and replace the buffered packet if the new packet has a higher ack number.

Finally, in order to ensure that idle connections are not needlessly kick-started, it is sensible to only re-receive on reconnection if there was a send attempt on that connection during the disconnected period. (Other methods of monitoring connection activity, such as idle timers, could also be used.)

4.3 Re-sending Packets

The smart link layer is also capable of re-sending packets at reconnection time, by inserting them in the outgoing queue for the disconnecting link. Re-sending has the disadvantage that some network latency is bound to be incurred before either TCP engine receives its kick-start. However, the advantage is that the re-sent packets may include new data, new acks, and/or new window updates.

Unlike in the re-receiving case, where the choice of which packet to buffer is simple (as there is no possibility of providing new data, merely repeating old data), care must be taken in the choice of a buffered packet for re-sending. In order to facilitate the quick restarting of TCP traffic, there are two obvious criteria for buffering, and two more subtle criteria. The obvious criteria are that the packet should have the highest acknowledgement number sent thus far, for the same reasons as for re-reception. The packet should also have the lowest unacknowledged sequence number, as this is the next "in-order" data that the remote TCP engine is expecting, and is therefore most likely to promote quick recovery of the TCP traffic flow. It must be noted that this criterion relies on the ability to monitor the current acknowledgement number, which must be found by scanning packets going in the opposite direction. This would not work if the acknowledgements take a different network path to the data, however, this is unlikely to be the case, as the disconnecting link would most often be on the access network for one of the endpoints.

More subtly, the buffered packet should be chosen as the longest length packet, and the one advertising the largest receive window. These criteria are relevant when considering packets retransmitted during the disconnected period. The former criterion relates to Nagle's algorithm [17], which states that only one packet which is smaller than the MTU should be unacknowledged at any time; other data should be queued at source rather than sent as further small packets. A corollary of using this algorithm is that, when timeout and retransmission occurs on small packets, the retransmitted packet may be longer than the original packet, as the retransmitting TCP will put as much data as possible in this packet, up to the MTU. A larger packet is obviously a better choice for buffering at the smart link layer, as most or all of the outstanding data can be sent immediately at reconnection, rather than incurring multiple round trip times for the data to be transferred.

The criterion of having the largest receive window advertisement is also related to retransmitted packets. During a disconnection period, it is likely that a host which has received but not processed some number of packets will be able to conduct some or all of this processing and therefore be able to send larger receive window advertisements in subsequent retransmissions.

In summary, for the re-sending algorithm, a packet is placed in the per-connection single packet buffer, if it:

1. Has a larger acknowledgement number than the current buffered packet, or
2. Has the same acknowledgement number but an older unacknowledged sequence number, or
3. Has the same acknowledgement and sequence numbers, but a longer length, or
4. Has the same acknowledgement number, sequence number and length, but advertises a larger receive window.

Finally, in order to avoid disturbing idle connections with this policy, re-sending on reconnection only occurs if there is unacknowledged data outstanding.

5. TESTBED EXPERIMENTS

To analyse the effect of the various link layer methods described above, an experimental setup using a simulated network channel was constructed. This allowed tests to be run using real traffic, but with precise control over the network connectivity, and also provided a platform for implementation and bug-fixing of the smart link layer algorithms.

5.1 Experimental Setup

To simulate a disconnecting channel, the Linux "ethertap" driver was used. This allows network packets to be routed to a user-level program, which simulates the lossy channel, and implements the send and receive components of the smart link layers. This setup is illustrated in Figure 3.

To implement the simulated channel, a two-state Markov model was used, with one state having 100% reliability and the other state having 0% reliability. The mean time spent in each state was configurable to allow different channel characteristics to be simulated. This is illustrated in Figure 4. The period spent in each state was modelled by uniform random distributions, between half and one-and-a-half of the desired means; this ensured that the results were not subject to interaction between TCP timers and the channel timing.

In addition to implementing this model, the simulation program was also made to reverse the IP addresses and TCP port numbers of all packets, thereby "mirroring" packets so that the local TCP/IP

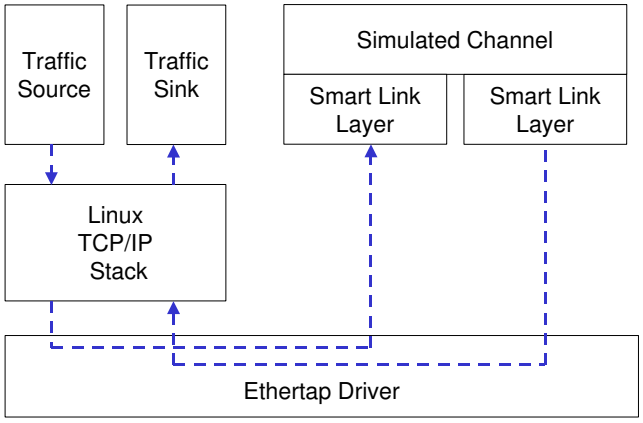

Figure 3: Ethertap Experiment Setup

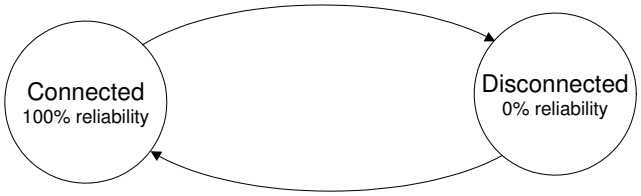

Figure 4: Simulated Channel Markov Model

stack treats the packets as incoming rather than outgoing. This allows networking tests to be performed on a single machine. Using this arrangement, networks with various connection and disconnection patterns can be simulated, and various smart link layer algorithms can be tested. One potential disadvantage of this arrangement is that the sender, channel model and receiver are all competing for CPU and memory bandwidth, however, this is not a factor in the tests below, since the behaviour being monitored is the time taken to recover from disconnected periods, during which no traffic is being sent (apart from retransmissions), and all elements of the testing machine are idle, with the sender stalled while waiting on TCP timers.

5.2 Optimisations Tested

From the design space discussed in the previous section, five potential optimisations were identified for experimental evaluation. These optimisations, known as "smartlvls," are outlined below:

Smartlvl 0 This is the control case, and represents unaided TCP.

Smartlvl 1 Re-receiving is used as defined in Section 4.2, and the buffered packet is re-received once upon reconnection.

Smartlvl 2 As for smartlvl 1, but the packet is re-received five times on reconnection.

Smartlvl 3 Re-sending is used as defined in Section 4.3, and the buffered packet is re-sent once upon reconnection.

Smartlvl 4 As for smartlvl 3, but re-sending five times on reconnection.

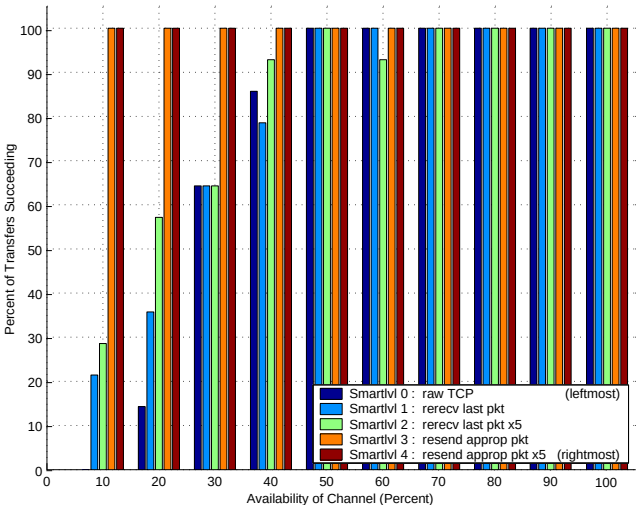

Figure 5: TCP File Transfer Tests over Simulated Disconnecting Channel

5.3 Results

In order to determine the relative performance of the various optimisations outlined above, timed bulk transfers were sent using the disconnecting channel described above. These were conducted with the following parameters.

- The transfer size was 50Mbyte.
- Mean "uptime" was set to 0.5s, while mean "downtime" was varied from to 0.0s to 4.5s to simulate channel availabilities from 10% to 100%.
- Fourteen trials were conducted at each of the ten availability levels and five smartlvls.
- The success or failure of each trial, and the time it took if successful, was noted.

As Figure 5 shows, the trials all succeed when availability is high. For availabilities below 50%, smartlvls 0, 1 and 2 begin to fail. Smartlvl 0 (raw TCP) fails most quickly, and at 10% availability experiences no successful transfers. Smartlvls 1 and 2, which use re-reception of packets, show some improvement, but smartlvls 3 and 4 are the definite "winners," with 100% of transfers completed successfully, even when the channel is only available 10% of the time.

Figure 6 shows the mean transfer time for the successful transfers, with an unmarked line indicating the minimum transfer time, which would occur if the channel were used optimally. As expected, unaided TCP degrades most quickly, and smartlvls 1 and 2 exhibit some improvement. Smartlvls 3 and 4, however, stay very close to the optimal line, providing good performance even on a channel with 10% availability.

This is further illustrated in Figure 7, which shows the low standard deviation of the trials, as compared to those of unaided TCP. This plot also allows the observation that, even at 80% or 90% availability, unaided TCP has already diverged from optimal performance,

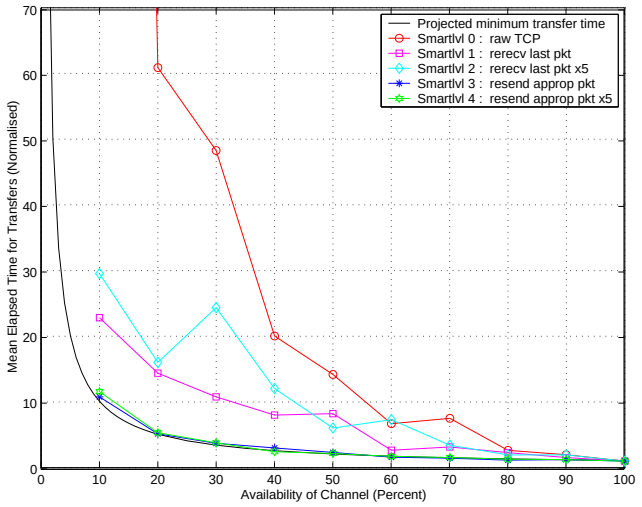

Figure 6: Mean Duration of Successful TCP File Transfer Tests over Simulated Disconnecting Channel

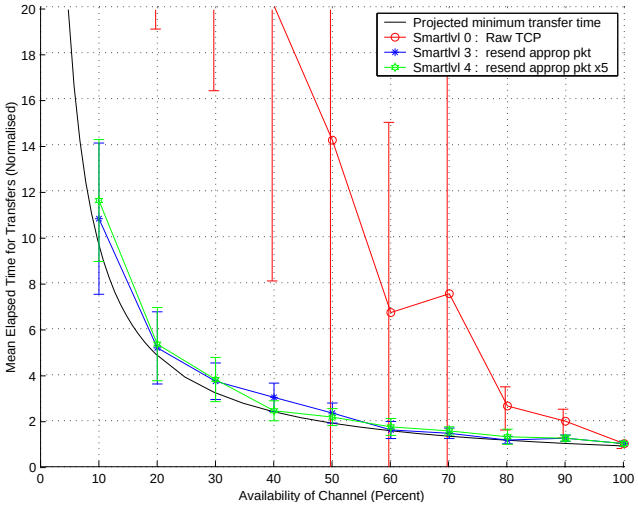

Figure 7: Detail of Figure 6 for Selected Smartlvls, with Standard Deviations

with degradations of about 100% and 50% respectively from the optimal case.

5.4 Analysis using TCP Traces

To further explore the behaviour above, traces of file transfers were taken with various smartlvls. These are shown in Figures 8 to 11. (Figure 1 showed a trace for unaided TCP, and includes a key useful for interpreting the traces.)

The traces for smartlvls 1 and 2[1] show that TCP does not respond immediately when receiving repeated acks. Although repeated acks are used as a signal to TCP to start fast-retransmit of a packet, this is not successfully invoked in either case. This can be explained by noting that the fast-retransmit mechanism is designed to be used *before* retransmission takes place due to timeout. If such a timeout has already occurred, then the congestion window has been reset to one packet. Hence, re-receptions of an old ack do not cause any response, as the window is not advanced by this event.

The traces for smartlvls 3 and 4,[2] on the other hand, show that TCP is immediately restarted after reconnection. This is because the packet re-sent on reconnection is chosen so that it sends unacknowledged data. When this packet is acknowledged, the congestion window is opened and slow-start proceeds as normal. Resending five packets at a time with smartlvl 4 does not seem to have any added effect; although they may cause multiple acks of that packet, the fast-retransmit mechanism is not useful in this case, due to the congestion window being reset as described previously.

The conclusion of these experiments is that a "smart link layer" employing re-sending of a well-chosen packet on reconnection can improve the performance of TCP on disconnecting channels. Whereas unaided TCP experiences bad performance even when the channel

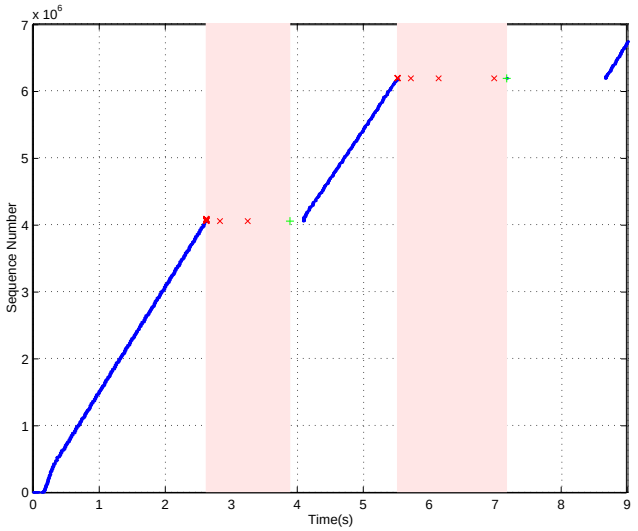

Figure 8: TCP File Transfer with Disconnection and Smart Link Layer (Smartlvl 1)

[1]The five individual plus marks for the five re-receptions used in smartlvl 2 are not distinguishable at this scale, and look like a single plus mark.

[2]The five individual circles for the five re-sent packets used in smartlvl 4 are not distinguishable at this scale, and look like a single circle.

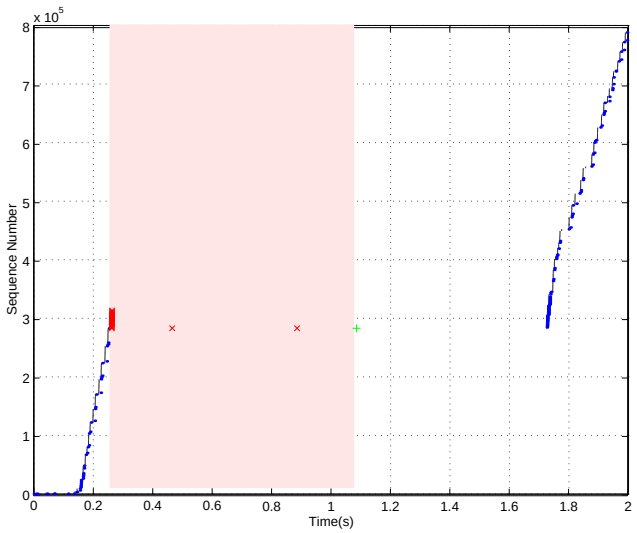

Figure 9: TCP File Transfer with Disconnection and Smart Link Layer (Smartlvl 2)

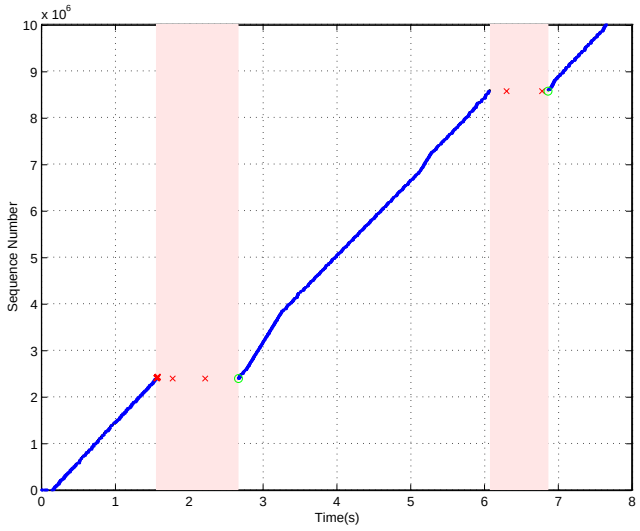

Figure 10: TCP File Transfer with Disconnection and Smart Link Layer (Smartlvl 3)

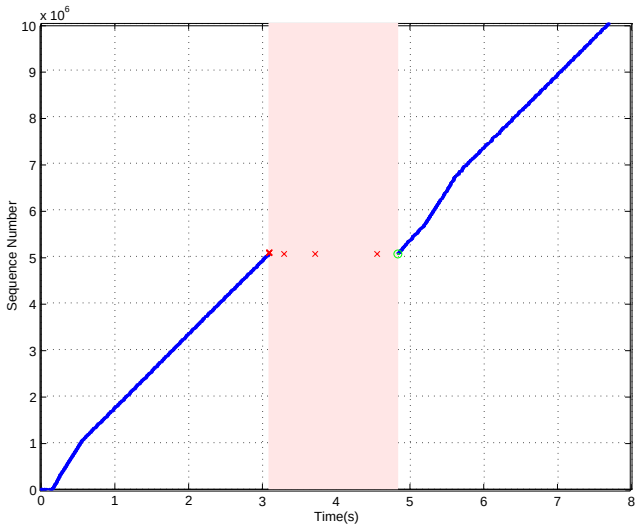

Figure 11: TCP File Transfer with Disconnection and Smart Link Layer (Smartlvl 4)

is 90% available, the use of the smart link layer gives nearly 100% channel utilisation at availabilities down to 10%.

6. EVALUATION OF THE SMART LINK LAYER ON THE PROTOTYPE NETWORKED SURFACE

This section describes the effects of the smart link layer described above, when deployed in the context of the prototype Networked Surface. The smart link layer was deployed at both the device and the node acting as the Networked Surface IP gateway, i.e. on both sides of the disconnecting link. Two types of test were then conducted. The first was similar to the experiments described in the previous section, and evaluated the bulk transfer performance of TCP. The second characterises the interactive response of TCP over a disconnecting Surface network, by using the Virtual Network Computing (VNC) remote desktop tool.

In order to conduct these tests, the prototype Networked Surface hardware was augmented with a testing mode to allow a device to disconnect at random intervals, with an adjustable mean connected period. The "uptime" and "downtime" of the network was also recorded, so that its availability could be calculated.

6.1 Performance for Bulk Transfers

In order to examine TCP performance for bulk transfers over a disconnecting Networked Surface, experiments were conducted according to the following parameters:

- The Networked Surface prototype was configured to provide a 1Mbit/s network.
- Disconnections were caused at various rates, producing various channel availabilities.
- Smartlvls 0 and 3 were used; smartlvl 0 is the unaided TCP case, and smartlvl 3 is the best-performing smart link layer optimisation, as shown previously.

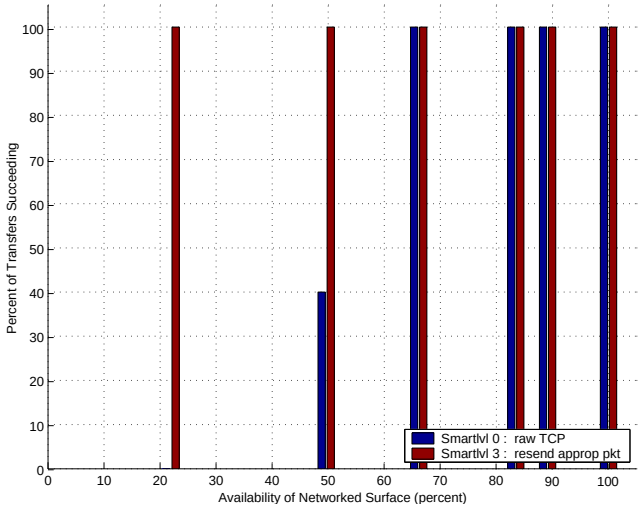

Figure 12: TCP File Transfer Tests over Disconnecting Networked Surface

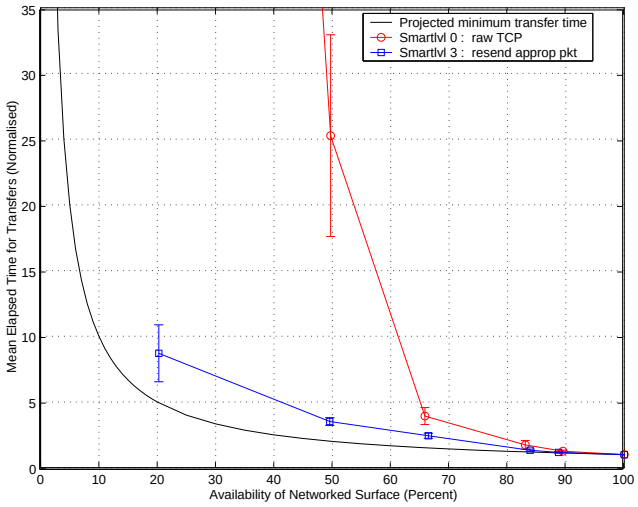

Figure 13: Mean Duration of Successful TCP File Transfer Tests over Disconnecting Networked Surface

- Ten transfers of 5Mbyte were conducted at each availability and smartlvl, and the elapsed times were recorded.

As Figure 12 shows, the smart link layer completed 100% of transfers, down to an availability of 23%, while unaided TCP did not reliably transfer the data at 50% availability or less. Figure 13 illustrates that the smart link layer stays relatively close to the "ideal" transfer time, even down to 20% availability. Unaided TCP has twice the overhead of the smart link layer at 65% availability, and very bad performance at lower availabilities.

6.2 Interactive Performance

While bulk transfer performance is important for some applications, the user of a networked device may also wish to communicate interactively. Examples of interactive applications are remote terminal programs, web interfaces, and real-time multimedia applications such as streaming audio or video. Such applications may not stress the bandwidth of the network available, so the tests in Section 6.1 are not necessarily applicable in this case. What is applicable, however, is *synchronisation*, i.e. the need for a local interface and the remote application to be representing the same state as much as possible. An example of bad synchronisation is when a user clicks on a webpage link, but only after a number of seconds does the page change to reflect this action. Another example might be a remote desktop mouse icon not following the local mouse icon closely while it is being moved.

In order to test the smart link layer's benefits for interactive applications, a quantitative metric must be found for synchronisation performance. As described below, the "frame rate" of a remote desktop application is one such metric.

6.2.1 Testing Method

The VNC [19] remote desktop application allows a user of one computer to interact with a remote computer, by "forwarding" the remote display across a network, and similarly relaying keyboard and mouse input. The VNC protocol operates as follows. When the client connects to the server, it issues an "update request" to that server. The server responds by waiting until its display differs from its record of the client's display, and then sending a "framebuffer update" containing changes to the client's display. On receiving this update, the client applies it and immediately sends another "update request."

Since the server only sends updates in response to requests from the client, the protocol is self-clocking. The requests and updates are sent over TCP, which retransmits the data if it arrives corrupted or is lost in transit, providing the guarantee that all messages eventually get delivered correctly (if channel conditions permit).

Due to the protocol outlined above, only one update is sent at any time. This implies that, for small updates, the frame rate achieved is determined by the latency of the TCP connection used and not by its bandwidth. The frame rate is therefore a good measure of interactive performance.

6.2.2 Experiments and Results

In order to gather frame rate data, tests were conducted using to the following parameters.

- A VNC desktop session was set up on a machine on the network.
- A program was run on the remote desktop, which caused a small dot to appear and disappear at regular intervals. The effect of this program was to cause the display to require a small update every 200ms.
- The Networked Surface was configured to have various availabilities, as described previously.
- Smartlvls 0 and 3 were used.
- For each test, the VNC viewer program was run on a computer using the Networked Surface for 100s, and record was made of the number of updates received over this time period.

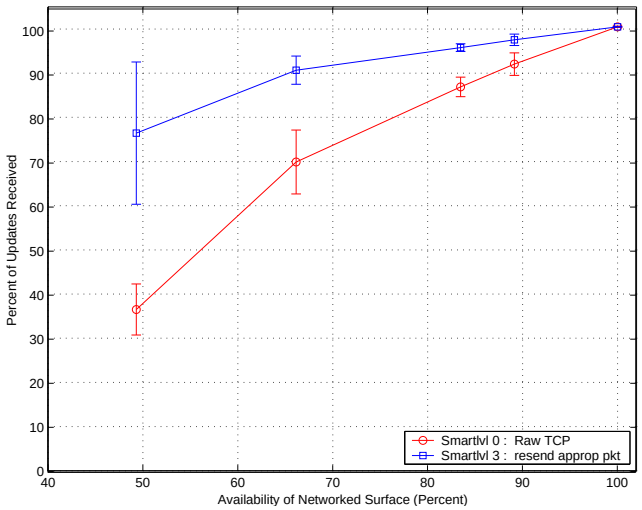

Figure 14: VNC Interactivity Tests over Disconnecting Networked Surface

- Ten tests were conducted for each smartlvl and availability.

Figure 14 shows the results of these tests. The smart link layer is shown to perform well, providing 80% of the frame updates even when the channel availability is halved, as opposed to 40% for the unaided TCP case. These results show that interactive performance of TCP over the Networked Surface channel is significantly improved when a smart link layer is used.

7. COMPARISON OF THE SMART LINK LAYER WITH OTHER SOLUTIONS

The link layer–based nature of the smart link layer allows it to enjoy several advantages when compared with solutions involving the modification of IP, ICMP and/or TCP that were described in Section 1.1. Link layer solutions can be independent of the TCP implementation used, and can therefore be deployed without concern for the devices' particular TCP implementations. In contrast, a TCP-based solution would have to be added to many different TCP implementations that are in use, some of which may be hard or impossible to modify (e.g. hardware-based TCP implementations). Also, link layer solutions may only affect nodes local to the disconnecting link. Deployment of such solutions is therefore easily carried out concurrently with deployment of the disconnecting network type.

In addition, the smart link layer's particular attributes allow it to enjoy some further advantages. As it is very lightweight, it can be deployed in modest hardware as part of a NIC for mobile computing devices, thus allowing it to avoid use of the scarce CPU and memory resources of those devices. Also, since it uses only a small amount of network bandwidth, it is relatively easy to secure it against malicious use, e.g. for denial of service attacks. Furthermore, because the smart link layer operates unilaterally, it does not require the use of authentication methods that might be necessary in solutions which communicate state information, e.g. for signalling a disconnected state to the TCP endpoints. Lastly, since the smart link layer only resends traffic that has already been sent, it is easy to see that no data security issues are created when using this solution.

However, there are also disadvantages to using link layer solutions, some of which are relevant to the smart link layer. To begin with, such solutions may experience bad interactions between TCP's retransmissions and retransmissions at the link layer [11]. The smart link layer, however, does not try to usurp TCP's role of ensuring data delivery; the retransmitted packets are solely used to kick-start TCP's own recovery mechanisms, and the fact that the data contained in this packet is delivered is a (welcome) side-effect. Bad interactions between retransmissions are therefore avoided.

Another disadvantage of link layer solutions is that they do not prevent TCP transmitting during the disconnected period. This may be seen in the first diagram, Figure 2, which shows that a full window of packets is transmitted fruitlessly during disconnection, and a periodic retransmission of the first packet in this window also occurs. In contrast, a TCP implementation which is "aware" of disconnection would halt transmission whilst in a disconnected state, thus saving network bandwidth.

The next disadvantage is that, not being integrated with TCP, it is not certain that a link layer solution will perform well with every version of TCP currently deployed, or with future versions of TCP that may become available. The smart link layer solution may incur this problem, however, as it only retransmits a single packet on reconnection, the network overhead imposed is low. Therefore, if the smart link layer solution fails to kick-start a given TCP connection, the network will not be significantly burdened, and TCP will simply assume its normal, if suboptimal, behaviour.

The final disadvantage to be highlighted is that the use of end-to-end encryption [1] may hinder or even disable link layer solutions which rely on being able to "sniff" packets. To handle encrypted TCP connections, the smart link layer could simply buffer the most recent packet per source and destination IP addresses, and retransmit this on reconnection. However, this technique would not cope with multiple TCP connections between two IP addresses (which it would be unable to distinguish), and may not send result in the best kick-start packet being buffered.

It must be noted that the smart link layer does *not* attempt to bypass the slow-start procedure of TCP [14], nor does it try to induce a raising of the congestion window. This policy is in contrast to many of the link layer solutions presented for TCP problems (and described in Section 1.1), which attempt to keep the congestion window wide despite bad channel characteristics. This difference is largely due to the timescales for which the solutions are designed; single packet losses happen on a microsecond scale, while disconnections may last a number of seconds. Also, during disconnection, devices may be moved to a different network; this may cause the congestion characteristics to change, so a slow-start is appropriate to discover the correct new value of the congestion window.

Finally, the relevance of the smart link layer to the current IETF proposals is now discussed. The smart link layer technique is in accord with the PILC working group's recommendation [16] for disconnecting networks to buffer and resend one or more of the packets arriving during the disconnected period. The validity of that technique is confirmed experimentally here, and different buffering criteria and retransmission techniques are examined.

Both PILC and the TRIGTRAN [10] proposal indicate that a pre-

ferred solution for disconnection handling is the end-to-end notification of disconnections (and other network events) so that TCP (or another transport layer protocol) can react appropriately. These proposals would avoid the disadvantages highlighted above, however, they require much more intrusive modifications of both the nodes attached to the disconnecting network and the edge nodes, as well as cooperation between these nodes, in order to solve this problem. This paper has shown that it is possible to work around the disconnecting network problem, at least in some circumstances. The techniques used in this research may prove a useful "stop-gap" solution for use while more thorough solutions involving modification of many entities in the network can be standardised and deployed.

8. CONCLUSIONS AND FUTURE WORK

This paper has presented the "smart link layer" solution for the problem of TCP's bad performance in the face of disconnecting links. Various algorithms repeating packets on reconnection were explored using an experimental testbed. Re-sending packets on reconnection proved to be more effective than re-receiving packets, and repetition of packets was shown to offer no additional improvement.

The best-performing solution was shown to achieve nearly 100% utilisation of a disconnecting channel, at availabilities down to 10%. Unaided TCP, on the other hand, shows a 50% degradation at 90% availability, and fails to complete some of the tests below 50% availability. The effect of this solution was also tested on the prototype Networked Surface, with good results shown for both bulk transfers and interactive traffic. In summary, this paper has shown the effectiveness of a simple and lightweight link layer–based solution for enhancing TCP performance in the face of disconnecting networks.

Future work involving the smart link layer may include testing of the link layer solution in a broader variety of circumstances. This may include tests involving connections over high-bandwidth high-latency networks, and tests where disconnection occurs in the middle of the network, or at multiple places in the network. Next, the current solution relies on the "kick-start" packet not being lost; this is valid given the assumption that the network is reliable when connected. In order to avoid this assumption, the smart link layer could be made to monitor the round-trip time of the connection, and then automatically retransmit the kick-start packet if it does not cause a reply within a suitable time. Thirdly, this solution might be ported to other situations where disconnected periods occur. For example, Mobile IP handoffs may take a significant time, during which the network appears to be disconnected at the TCP level. Fourthly, the use of this solution in the presence of encryption might be explored. Lastly, the viability of this solution with other transport-level protocols, e.g. SCTP [22], may be examined.

9. ACKNOWLEDGEMENTS

The authors are grateful to Frank Hoffmann, Andy Hopper and Mike Addlesee of the Networked Surfaces project at the LCE for indirect contribution to this work. The authors would also like to thank James Weatherall for helpful discussions concerning the operation of VNC, Mike Hazas and Tom Kelly for proofreading this paper, and John Wroclawski and the anonymous reviewers for their helpful comments.

This research was funded by the Schiff Foundation of Cambridge and by AT&T Laboratories Cambridge.

10. REFERENCES

[1] R. Atkinson. IP encapsulating security payload (ESP). RFC 1827, August 1995.

[2] E. Ayanoglu, S. Paul, T. F. LaPorta, K. K. Sabnani, and R. D. Gitlin. AIRMAIL: A link-layer protocol for wireless networks. *ACM Wireless Networks*, 1(1):47–60, 1995.

[3] A. V. Bakre and B. R. Badrinath. I-TCP: Indirect TCP for mobile hosts. In *Proceedings of the 1995 International Conference on Distributed Computing Systems*, pages 136–143, 1995.

[4] B. S. Bakshi, P. Krishna, N. H. Vaidya, and D. K. Pradhan. Improving performance of TCP over wireless networks. In *Proceedings of the 1997 International Conference on Distributed Computing Systems*, 1997.

[5] H. Balakrishnan, V. N. Padmanabhan, S. Seshan, and R. H. Katz. A comparison of mechanisms for improving TCP performance over wireless links. *IEEE/ACM Transactions on Networking*, 5(6):756–769, 1997.

[6] H. Balakrishnan, S. Seshan, and R. H. Katz. Improving reliable transport and handoff performance in cellular wireless networks. *ACM Wireless Networks*, 1(4):469–481, 1995.

[7] K. Brown and S. Singh. M-TCP: TCP for mobile cellular networks. *ACM Computer Communication Review*, 27(5):19–43, 1997.

[8] R. Caceres and L. Iftode. Improving the performance of reliable transport protocols in mobile computing environments. *IEEE Journal of Selected Areas in Communications*, 13(5):850–857, 1995.

[9] K. Chandran, S. Raghunathan, S. Venkatesan, and R. Prakash. A feedback-based scheme for improving TCP performance in ad-hoc wireless networks. In *Proceedings of the 1997 International Conference on Distributed Computing Systems*, pages 472–479, 1997.

[10] S. Dawkins, C. E. Williams, and A. E. Yegin. Framework and requirements for trigtran. Internet-Draft `draft-dawkins-trigtran-framework-00.txt`, February 2003.

[11] A. DeSimone, M. Chuah, and O. Yue. Throughput performance of transport-layer protocols over wireless LANs. In *Proceedings of GLOBECOM '93*. IEEE, 1993.

[12] S. Goel and D. Sanghi. Improving TCP performance over wireless links. In *Proceedings of TENCON '98*, pages 332–335. IEEE, December 1998.

[13] G. Holland and N. H. Vaidya. Analysis of TCP performance over mobile ad hoc networks. In *Proceedings of the Fifth International Conference on Mobile Computing and Networking*, pages 219–230. ACM/IEEE, August 1999.

[14] V. Jacobsen. Congestion avoidance and control. In *Proceedings of SIGCOMM '98*. ACM, 1988.

[15] V. Jacobsen. Modified TCP congestion avoidance algorithm. Email on `end2end-interest` mailing list (`ftp://ftp.ee.lbl.gov/email/vanj.90apr30.txt`), April 1990.

[16] P. Karn, Editor. Advice for internet subnetwork developers. Performance Implications of Link Characteristics (PILC) Working Group: Internet-Draft `draft-ietf-pilc-link-design-13.txt`, February 2003.

[17] J. Nagle. Congestion control in IP/TCP internetworks. RFC 896, January 1984.

[18] C. Parsa and J. J. Garcia-Luna-Aceves. Improving TCP performance over wireless networks at the link layer. *ACM Mobile Networks and Applications*, 5(1):57–71, March 2000.

[19] T. Richardson, Q. Stafford-Fraser, K. R. Wood, and A. Hopper. Virtual Network Computing. *IEEE Internet Computing*, 2(1):33–38, February 1998.

[20] J. Scott. *Networked Surfaces: A Novel LAN Technology*. PhD thesis, University of Cambridge, 2002.

[21] J. Scott, F. Hoffmann, G. Mapp, M. D. Addlesee, and A. Hopper. Networked Surfaces: A new concept in mobile networking. *ACM Mobile Networks and Applications*, 7(5), October 2002.

[22] R. Stewart, Q. Xie, K. Morneault, C. Sharp, H. Schwarzbauer, T. Taylor, I. Rytina, M. Kalla, L. Zhang, and V. Paxson. Stream Control Transmission Protocol. RFC 2960, October 2000.

[23] N. Vaidya, M. Mehta, C. Perkins, and G. Montenegro. Delayed duplicate acknowledgements: A TCP-unaware approach to improve performance of TCP over wireless. Technical report, Computer Science Department, Texas A&M University, 1999.

4+4: An Architecture for Evolving the Internet Address Space Back Toward Transparency*

Zoltán Turányi, András Valkó, Andrew T. Campbell
COMET Group, Columbia University
2960 Broadway New York, NY 10027
{zoltan,andras,campbell}@comet.columbia.edu

ABSTRACT

We propose 4+4, a simple address extension architecture for Internet that provides an evolutionary approach to extending the existing IPv4 address space in comparison to more complex and disruptive approaches best exemplified by IPv6 deployment. The 4+4 architecture leverages the existence of Network Address Translators (NATs) and private address realms, and importantly, enables the return to end-to-end address transparency as the incremental deployment of 4+4 progresses. During the transition to 4+4, only NATs and end-hosts need to be updated and not the network routers. The 4+4 architecture retains the existing semantics of Internet names and addresses, and only proposes simple changes to the network layer that focus entirely on address extension. Encapsulation is used as the main tool to maintain backward compatibility. We present the design, implementation, and evaluation of the 4+4 architecture and discuss our implementation experiences and results from local and wide-area Internet experimentation. The 4+4 source code is freely available from the Web (comet.columbia.edu/ipv44) for experimentation.

1. INTRODUCTION

One of the major challenges facing the current Internet is the exhaustion of the IPv4 address space. In 1994, the IETF Address Lifetime Expectations (ALE) Working Group projected the exhaustion of the IPv4 address space to be around 2008 [6]. Currently, there are two competing solutions to IPv4 address extension: Network Address Translators (NAT) [8] and IPv6 [16]. NATs are routers that can modify the IP address and port numbers of passing packets. This enables the reuse of a small number of public addresses by many hosts behind a NAT, if only a few of those hosts communicate with the outside world simultaneously. NATs have helped reduce the rate of address depletion, enabling continued operation even in regions with shortages of IP addresses. As discussed widely in the literature [23, 25], NATs also contributed to the loss of IP transparency [21]. As a result NATs prevent IP level end-to-end security, reduce robustness, break a number of application level protocols, complicate the network, and inhibit novel uses of the Internet (e.g., peer-to-peer networking). Despite such disadvantages NATs are widely deployed in the network. And, although NATs adversely impact the global Internet, they represent an attractive technology for many private network operators for a number of reasons. First, if the acquisition of public IP addresses is hard, complicated, and expensive, then setting up a NAT is quick, easy, and cheap. Next, for many end users a "NAT-ed" connection seems good enough. Third, alternative solutions (e.g., IPv6 deployment) seem too complicated and disruptive to deploy at present. Finally, NATs enable the isolation of a private domain's addressing and routing from that of the public Internet.

In 1994, the IETF selected IPv6 [16] as the next generation of the Internet Protocol. At that time, however, NATs were not yet in widespread use. The ngtrans Working Group of IETF [29] developed several transition mechanisms that would allow the temporary co-existence of IPv4 and IPv6. However, despite the availability of IPv6 and transition procedures, IPv6 has seen little practical deployment. One reason is the complexity associated with the transition. If one cannot replace all the routers and hosts in a site at once, then a period of co-existence follows. During such a period network of co-existence administrators must manage both IP versions, plus a number of transition mechanisms. While transition to IPv6 would benefit the whole Internet at the expense of the extra work at individual networks, using NATs benefits the individual networks directly, but negatively impacts the global Internet. This dilemma between NATs and IPv6 is best discussed in [24].

If IPv6 is ever fully deployed it is likely that the transition to IPv6 may last for a long period of time in the global Internet. In fact, it is possible that it may never be completed with a significant portion of the Internet remaining IPv4 only. Such a situation could occur if a certain population or sector (e.g., the cellular industry because of the projected growth in cellular Internet devices, or regions with shortages of IP addresses) found sufficient incentives to transition to IPv6 while others remained content with IPv4. This would result in a possibly lengthy partitioning of the Internet with intensive use of protocol translation and tunneling mechanisms, a result that could be even worse than the present situation with NATs. In short, it is possible that IPv4 and NATs will be a permanent fixture of the communications infrastructure for sometime to come. If this is the case, then a markedly different approach is necessary to solve the address depletion and transparency problems. One important alternative approach is represented by *NAT extended architectures* [27], which rely on the existence of multiple address realms and extend the address space by requiring changes only to hosts and NAT devices.

In this paper, we propose the 4+4 architecture [17], which presents one example of a NAT-extended architecture. 4+4 provides a way

back to unique, global, network layer host addresses, while maintaining some of the address isolation features of NATs. End-to-end transparency is restored in the Internet to the extent of 4+4 deployment. If 4+4 deployment becomes complete then IP address transparency is fully restored. There is no need to change routers. The transition process provides incentives for networks with both private and public addresses to upgrade and increase transparent reachability. During transition, existing IP and NAT mechanisms are used to communicate with, and between IPv4 hosts, thus, transition does not affect existing reachability. All transition mechanisms are part of either the current practice or the "final architecture", thus, there is no need to set up temporary transition features. The final state of the network is a homogeneous and transparent Internet that uses 64-bit addresses and a packet format similar to tunneling.

The paper is organized as follows. Section 2 provides a brief overview and discussion of previous address extension proposals. Section 3 and Section 4 describe the architecture and operation of 4+4, respectively. The 4+4 transition process with a brief comparison to IPv6 is presented in Section 5. A minimal impact 4+4 implementation is described in Section 6. In Section 7, the resilience, performance, and application compatibility of 4+4 is evaluated through experimentation using local and wide area 4+4 testbeds. Finally, Section 8 presents some concluding remarks.

2. RELATED WORK

There has been a considerable amount of work on IPng documented in the literature. Much of this work, however, predates the design and deployment of NATs. The Simple Internet Protocol Plus (SIPP) [10], which later became IPv6, includes a built in address extension mechanism. In SIPP, host addresses were originally 64-bits long with an additional "cluster-addressing" mechanism. Cluster addresses are 64-bit unicast addresses referring to a set of nodes behind boundary routers. When complemented with a 64-bit host address they enable the extension of the address space in quite a similar manner to 4+4. The cluster address selects the boundary router and the host address selects the host. Other uses of cluster addressing include mobility and provider selection. Later the IPv6 address space was extended to 128 bits and cluster addressing was omitted and merged into source routing.

Mike O'Dell proposed the separation of identifiers from locators in his 8+8 proposal, later known as GSE [15]. In 8+8 half of the 128-bit IPv6 address is used as a locator (termed "routing goop") and the other half as a host identifier. End systems are not aware of their full routing goop, only the part that describes their location inside their site. Site border routers place the missing part of the routing goop into the source address of outbound packets. Although there are some similarities between 4+4 and 8+8, a number of differences exist. First, in 4+4 no part of the address is used as a location independent host identifier. Second, border routers in 4+4 do not rewrite addresses; although realm gateways associated with 4+4 rearrange addresses when forwarding packets, no new addressing information is added to the packet. Next, 4+4 hosts are aware of their full addresses. Finally, the aim of the two proposals is different. While 8+8 introduces new address semantics to achieve a number of goals, the purpose of 4+4 is address space extension with minimal changes to address semantics. Analysis of 8+8 can be found in [18].

An alternative approach proposed for IPng is Nimrod [13], which also separates host identifiers from routing locators. Routing locators are hierarchically organized based on provider-customer relationships to allow natural prefix aggregation. The PIP proposal [7] is similar in separating identifiers and locators. The locator used by PIP is a list of values that can be thought of as locally significant addresses at a given level of the topology. The list effectively specifies a source route. Hosts may learn parts of the locator from the configuration, incoming packets, and the directory. PIP eventually merged into SIPP and its locator semantics are reflected mostly in SIPP's cluster addressing. We note that the idea of using a list of fields for addressing has already been considered during the design of IP [1]. The primary reason was for address extension, but again the idea was abandoned as the final 32-bit address seemed large enough. 4+4 is similar to PIP, cluster addressing and [1] in using a two-level address as a locator. However, 4+4 is different because it is incremental to the existing Internet addressing and protocol. See [9] for a detailed discussion of addressing issues.

In contrast to the IPng proposals, a number of other ideas are built on the reuse of the existing 32-bit address space. The separation of private and public address space was first proposed in [4] and then in [11]. Address translation and NATs emerged as a way to connect private networks to the global Internet. Realm Specific IP [28] starts from private address realms and NATs. It provides an explicit way for hosts in private address realms to obtain a public address when there is a need to contact a peer in a public address realm. Such hosts would then tunnel their traffic destined to a public peer through a private realm to the NAT. The NAT, in turn, de-capsulates the traffic and forwards it to the public Internet. The drawback of realm specific IP is that while it does restore network layer transparency it does not extend the address space, and as such, can only be considered a temporary fix to the problem.

Robert Hinden proposed a "medium term" routing and addressing scheme [12] that also uses tunneling as its main tool. Border routers of Autonomous Systems (AS) encapsulate egress traffic and put the source and destination AS numbers into the outer IP header. A block of IP addresses is set aside to identify ASes in such headers. These addresses are injected into the interior routing of transit ASes so only border routers need to recognize them as being special. When the packet reaches its target AS, the ingress border router de-capsulates the traffic and forwards it into the AS. If IP addresses are not globally unique then Hinden suggests the use of an AS address/host address pair as an extended address and adds that "this could be the basis for the long term solution". This idea is very similar to 4+4, but 4+4 addresses have no AS significance. Also the swapping of the two parts is new and central in 4+4 routing. The use of tunneling as a tool to solve the NAT problem is also mentioned briefly in [21, sec. 5.2.1.3] as something that "has never been fully developed, although is fully compatible with end-to-end addressing". 4+4 aims to fill that gap.

The IETF also investigated the use of IP options for address extension, as suggested by Brian Carpenter [5]. This idea was abandoned and never implemented mostly because it is based on IP options and hence requires changes to ARP, DNS, SNMP, and routers within a site. 4+4 is similar to this idea, as well, but uses tunneling instead of IP options to carry the extended addressing information.

The ideas discussed above naturally point toward NAT-extended architectures. The recent IP Next Layer (IPNL) [27] proposal is an early example of a NAT-extended architecture. IPNL uses existing IP address realms as "links" to an overlay (or next layer) protocol. The new protocol is tunneled in IP packets and operates

below the transport layer. Enhanced NAT boxes, called *nl-routers*, route packets realm by realm toward the destination. IPNL introduces *realm numbers* to identify different realms behind the same *frontdoor* - an nl-router that connects private realms to the public Internet. The public IPv4 address of a frontdoor concatenated with a realm number identifies a private realm. The (frontdoor address, realm number, host address) triplet, called IPNL address, fully identifies a host in a private realm. However, IPNL addresses are not used as long-term host identifiers, only as locators that can change frequently. One reason for change might be a switch to a new frontdoor (e.g., when the old one fails). IPNL uses fully qualified domain names (FQDNs) as host identifiers. Nl-routers are able to route packets based on both FQDNs and IPNL addresses. During the initial packet exchanges peers use FQDNs in packet headers to address each other. At the end of such an exchange peers learn both their own and each other's IPNL addresses, and uses those addresses for subsequent communications. If a connection toward two peers exits through different frontdoors then a host may learn two or more different addresses for itself. Hosts are not aware of all their possible addresses. This is not necessary, as the FQDN can be used in packets to identify the host. If the IPNL address changes during a session, then peers can automatically detect this and switch to using the new address. To facilitate such routing a new routing protocol is installed among nl-routers behind the same frontdoor that distributes DNS and realm number information among nl-routers. Nl-routers are also aware of the FQDNs and host addresses of all hosts in the realms they are attached to. This enables them to resolve FQDNs to IPv4 addresses for incoming FQDN-addressed packets. Nl-routers are also capable of performing realm number translation to allow two realms behind the same frontdoor to use the same realm number.

While we believe that IPNL is much easier to deploy than IPv6, IPNL is a fairly complex architecture that represents a significant departure from the current routing and addressing paradigm. Hosts are no longer aware of their full network layer addresses, only their names. Host identifiers are DNS names once and for all. Moreover, the routing infrastructure becomes irrevocably intermingled with the DNS. Domain names are intentionally aggregate administratively and not topologically. A distinction between the current public address realm and private address realms is maintained in the architecture. It is not clear if, or how, this distinction can be removed later and the Internet made homogeneous again. Besides the more important architectural impact, IPNL routing raises some performance issues too. Nl-routers need to manage DNS related routing information and a per-host database. Per-packet processing is complicated for some packets. The packet header is also quite large, 44 bytes for intra-realm packets and 60 bytes for global packets, plus the size of FQDNs if used.

TRIAD [26] is also a new Internet architecture that builds on the existence of IPv4 address realms and uses names as identifiers. Its primary focus is content distribution. TRIAD introduces a *content layer* and *Content Routers*. Content routers hold DNS information and forward combined name lookups/connection setup requests toward destinations. The result of this lookup represents transport connection information and a list of relay identifiers that specify a path through several consecutive address realms. A shim header is added to IP packets that contains the list of relays, enabling *relays* at the border of address realms to forward packets. IP addresses become locally significant with names representing globally unique identifiers. In comparison, 4+4 proposes simple changes to the network layer focusing entirely on address extension while retaining existing semantics as far as possible.

3. 4+4 ARCHITECTURE

The primary goal of the 4+4 address extension is to provide nodes currently in private address realms with an end-to-end address. The 4+4 extended address is formed by concatenating two 32-bit IPv4 addresses: a public and a private one. The public address selects the address realm, while the private address selects the node inside the realm. In fact, the public part is the address of the NAT connecting the realm to the public Internet. Nodes in the public Internet use their existing address as the public part and 0.0.0.0 as the private part.

4+4 packets are minimally encapsulated IP packets [14]. They contain two 32-bit fields for each of the source and destination addresses. The two parts of the extended address are placed into the two 32-bit address fields. The fields are managed such that the outer header always contains addresses that are understood by the IPv4 routers in the realm the packet is transiting; that is, in a private realm the private half of the extended address is visible in the outer header, while in the public Internet the public half is visible. This ensures that routers can forward packets toward the destination without understanding 4+4.

In what follows, we describe addressing and the header format, while routing is discussed in Section 4.

3.1 Network Model

We define an address realm as a collection of networks using addresses from the same address block, while using one address only once; that is, an IP address unambiguously identifies an interface within an address realm. We further differentiate the one and only public address realm that uses the public IPv4 address space and several private address realms, each of which may re-use the private address space designated by [11]. It is also possible for a private address realm to use a block of IP addresses not belonging to [11], if none of those addresses are used anywhere in the public address realm. This allows a private realm to maintain any existing addressing when transiting to 4+4. For the sake of clarity in this paper, we use the term "private" for realms and addresses outside the public realm. Both the public and private realms contain the usual TCP/IP networks with routers and hosts. Today, private realms connect to the public realm via NATs.

Figure 1 shows a typical network scenario assumed by 4+4. The "grey colored" networks belong to the public address realm and each "white colored" network represents a separate private realm. For example, networks 1 and 3 represent realm ***A*** and realm ***B***, respectively, and both can re-use the entire private address space.

Upgraded NATs, called *realm gateways*, represent an integral part of the 4+4 architecture. In addition to the address translation function, realm gateways perform a few simple operations on 4+4 packets. Note that the use of these address translation functions will diminish as transition to 4+4 progresses and be fully removed once full deployment is complete. Realm gateways also act as legacy routers that forward IPv4 packets with public destination addresses. Realms may be interconnected using arbitrary topology and an arbitrary number of realm gateways. The only requirement is that each interface of a realm gateway must strictly belong to either a private or the public realm to separate the address spaces. Note that the public address realm need not be contiguous.

In Figure 1 private realm ***A*** and ***B*** are connected to the public network 2 via realm gateways with public addresses **A** and **B**, respectively. These addresses are separate from the address pool assigned for the address translation function. The figure also shows four hosts, two in the public realm (nodes **C** and **D**) and two in private address realms (nodes **X** and **Y**).

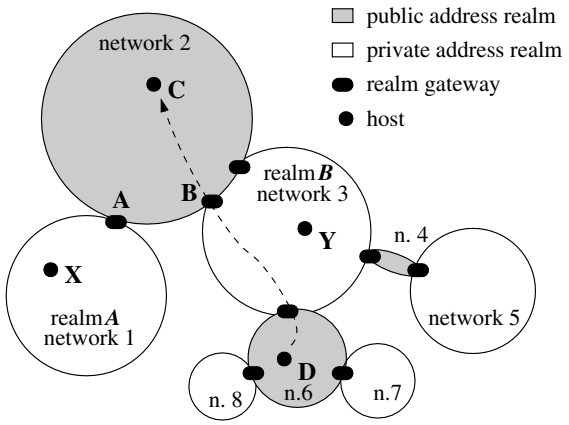

Figure 1: Example networks with arbitrarily interconnected realms.

IPv4 addresses in the public realm are called *level 1* addresses (addresses **A**, **B**, **C** and **D** in Figure 1), while addresses in private realms are called *level 2* addresses (**X** and **Y**). In the remainder of this paper bold capitals are used to denote 32-bit IPv4 addresses and the nodes having those addresses. The term *router* always denotes a legacy IPv4 router that is unaware of 4+4.

The routers inside private address realms are configured with the routing information for both private and public addresses; that is, they know how to route toward both level 1 and level 2 destinations. For example, assume that node **D** in network 6 posts a packet to node **C** in network 2, as illustrated in Figure 1. The packet uses public source and destination addresses and is delivered unaltered to the destination through the routers and realm gateways of network 3. The routers in the public address realm, on the other hand, are configured to route toward public addresses only. This means that realm gateways must filter out routing information of private addresses when communicating with routers in the public address realms. Note that realm boundaries do not have to coincide with autonomous system boundaries.

3.2 Addressing

The 4+4 address of a node inside a private realm is a concatenation of two IPv4 addresses: the public address of a realm gateway and one of the node's own private addresses. We denote this as **A.X**, where **A** represents the realm gateway and **X** represents the node's own address. The first and second parts of the address are called *level 1* and *level 2* parts, respectively. 4+4 nodes may have multiple addresses if the host has multiple interfaces or the realm is 'multihomed' (i.e., there are multiple realm gateways with different addresses). Any (level 1 part, level 2 part) combination constitutes a valid address of a node. Multiple addresses of the same node are treated the same way as in IPv4, that is, nodes accept packets on all of their addresses and may use any of their addresses as the source address for outgoing packets. Transport layer semantics remain unchanged where sockets are bound to a tuple of two 4+4 addresses and two port numbers.

The DNS is used to store and retrieve 4+4 addresses in conceptually the same manner as with IPv4. There are two possible alternative ways to store 4+4 addresses in the DNS. First, a new record type could be defined. Second, two type A records could be used to store the level 1 and level 2 parts of the 4+4 address. The two records would be accessible through a prepended domain name, similar to SRV records [20]. For example, the level 1 and 2 parts of the 4+4 address of the machine `foo.bar.edu` could be stored under `_l1.foo.bar.edu` and `_l2.foo.bar.edu`. The benefit of the latter approach is that it requires no modification to DNS servers. Our implementation, which is discussed later in this paper, uses this approach.

A host may have multiple level 1 and level 2 address parts. This is similar to a host having multiple IPv4 addresses today. The level 2 address of a host inside a private realm is also advertised as a legacy IPv4 address to allow IPv4 communications inside a realm. Similarly, the level 1 address of a host in the public Internet is also available as an IPv4 address. Reverse DNS can be provided by prepending the reverse of the level 2 address part to the usual d.c.b.a.in-addr.arpa DNS name.

One important feature of 4+4 is the isolation of routing and addressing between various realms. The administrator of one private realm may choose arbitrary routing and addressing plans inside the realm without affecting other realms. Realms can also be extended without involving any globally coordinated address allocation process. The isolation also permits the migration of a private realm (e.g., because of a change of provider) without changing the addresses and communication inside the realm. Naturally, if a private realm changes the public address of its realm gateway, the 4+4 address of all the hosts inside will change. This, however, does not affect IPv4 routing and the traffic inside the domain. Provider change affects only the level 1 part of the 4+4 address and the old level 1 part can co-exist with the new one for extended periods of time. During such a coexistence new sessions with external hosts can start with the new level 1 part allowing smooth migration. We note that such isolation also exists when using NATs. Besides easy address expansion, this is another important feature of NAT that makes it a popularity technology.

Finally, we note that more than two levels of address parts is possible, if the amount of address extension provided by the two levels is not sufficient. Details can be found in [31] that address this issue.

3.3 Header Syntax

The 4+4 header is shown in Table 1. On the one hand it can be viewed as an IPv4 encapsulated IPv4 packet with a syntax similar to minimal IP-in-IP encapsulation [14]. This is how legacy IPv4 routers view the packet; they process only the IPv4 header part leading to backward compatibility. On the other hand, the full 4+4 header can be viewed as a new network protocol header with 64-bit source and destination addresses. This is how 4+4 capable endhosts view it.

The first three rows of fields in the 4+4 header are interpreted in the same manner as the IPv4 header, with the exception that the `Protocol 1` field is set to a value that specifies 4+4 encapsulation. Currently this value is 233. The `Source Address 1` and `2` fields collectively contain the full 4+4 address of the source node, while the `Destination Address 1` and `2` fields con-

Ver.	Hlen	DS byte	Total Length
Identification		Flag	Fragment offset
TTL		Protocol 1	Header Checksum 1
Source Address 1			
Destination Address 1			
Source Address 2			
Destination Address 2			
Protocol 2	SPos	DPos	Header Checksum 2

Table 1: The 4+4 header

tain the full destination address.

The `SPos` and `DPos` fields indicate how the source and destination 4+4 addresses are partitioned into the two 32-bit fields. A value of 0 means that the address is *unswapped*, that is, the level 1 address part is in the outer header and the level 2 is in the inner header. A value of 1 means that the two address parts are exchanged and the address is *swapped*.

The `Protocol 2` field indicates the transport protocol, while the `Header Checksum 2` covers the end-to-end information in the 4+4 header. This includes the addresses, the `Protocol 2` field, and the payload length, which is the total length minus the IP header length.

Similar to IPv6, only end hosts may fragment IP packets. This is accomplished by setting the `Don't Fragment` bit in the IPv4 header and using path MTU discovery [3]. The fragmentation related fields of the IPv4 header are used exactly as in IPv4. All fragments contain the inner header as well. Reassembly is performed at the final destination only.

A system using extension headers similar to IPv6 can be defined for 4+4. This would allow the reuse of several mechanisms defined for IPv6. Fragmentation can also be made part of the extension header mechanism leaving the second row of the header mostly unused. Due to the change in addressing, the TCP and UDP pseudo-headers also change to include the full 64-bit address in calculating checksums when sending 4+4 packets.

We note that an alternative way of storing the 4+4 address information in packets would be the use of a new IP option. Legacy routers would only need to transparently forward this option unchanged. However, it may seriously impact performance caused by slow-path processing in many routers for all 4+4 packets. In addition, packets with unknown IP options are often dropped in the Internet. As a results of these issues, we did not pursue this alternative further.

4. ROUTING

One of the benefits of 4+4 is that IPv4 only nodes can essentially communicate as today. They use IPv4 packets, the existing IPv4 routers and if they are in different address realms then they use NATs. Therefore, no new transition mechanisms are needed to provide service to legacy IPv4 hosts.

Four different scenarios are possible with regards to the relative location of two IPv4 only nodes. If the two nodes are located in the same private realm, the private IPv4 addresses and the IPv4 protocol are used. If both nodes reside in the public address realm, then they can use their public addresses and the IPv4 protocol to communicate, even when the nodes are separated by one or more private realms. This is possible because the routers inside the private realms have public address routing information and are capable of forwarding packets with public addresses. If one of the IPv4 nodes is in a private realm and the other node is in the public Internet, then address translation is performed as it is done today using a NAT. In this case, the known problems of NATs apply. These limitations can be resolved by upgrading the nodes to 4+4. Finally, if both nodes are in different private address realms then it is impossible for them to communicate unless they upgrade to 4+4. In what follows, we describe, how two 4+4 nodes in different private address realms can communicate with each other.

4.1 Routing between two Private Realms

Assume that a node **X** wishes to send a packet to node **Y**, as illustrated in Figure 2 and shown with solid arrows. Assume further, that both nodes are 4+4 aware, that is, their operating systems can send and receive 4+4 packets. First, node **X** checks if any of the level 1 address parts returned by DNS for node **Y** match any of its own level 1 address parts. If that is the case then the two nodes are in the same address realm and the node uses IPv4 packets to communicate. If this is not the case then **X** selects one level 1 and level 2 part from the set of address parts of node **Y** to form the destination address (e.g., **B.Y**). A source address is also selected (e.g., **A.X**).

Next, the source node creates a 4+4 header and fills in the source and destination address fields as follows. The level 1 part of the source address is placed in `Source Address 2` and the level 2 part in `Source Address 1`. The level 1 and level 2 parts of the destination address are placed in `Destination Address 1` and `2`, respectively. In other words the source address in the packet is swapped, while the destination address is unswapped. This packet header is denoted symbolically as,

$$\begin{matrix}\mathbf{X} \\ \mathbf{A}\end{matrix} \rightarrow \begin{matrix}\mathbf{B} \\ \mathbf{Y}\end{matrix}$$

where the upper row represents the addresses in the outer IPv4 header (source address is **X**, destination address is **B**) and the lower row represents the addresses in the inner header (source address is **A**, destination address is **Y**). The full 4+4 source address **A.X** is in the left column and is swapped. The full 4+4 destination address **B.Y** is in the right column and is unswapped. The IPv4 routers in realm ***A*** only see $\mathbf{X} \rightarrow \mathbf{B}$.

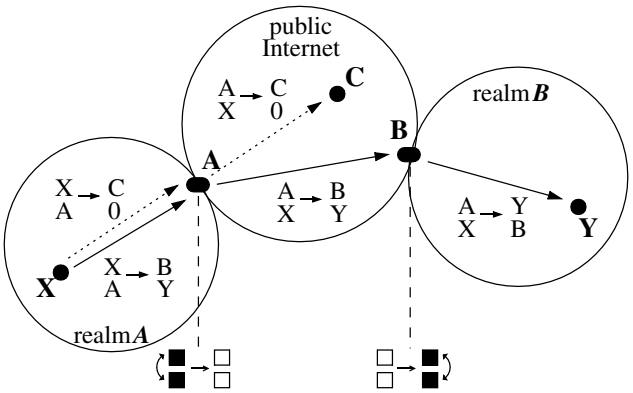

Figure 2: Routing between two 4+4 nodes in different realms.

As a result, the routers will forward the packet toward node **B**, as shown in Figure 2. When it reaches the border of realm ***A***, the realm gateway exchanges the content of the fields `Source Address 1` and `2` leaving the source address unswapped. This is depicted visually at the bottom of Figure 2. Following this the packet is forwarded into the level 1 (public) Internet. At this point the addresses in the packet are as shown in Figure 2. The IPv4 routers see only $\mathbf{A} \rightarrow \mathbf{B}$ and continue forwarding the packet toward node **B**. We note that in the case of a 'multihomed' realm, it may be a realm gateway other than **A** that executes the change. When node **B** receives the packet and sees that it is a 4+4 packet, then it swaps `Destination Address 1` and `2`. The outer header contains $\mathbf{A} \rightarrow \mathbf{Y}$ allowing the routers inside realm ***B*** to forward the packet to node **Y**. When node **Y** receives the packet, it recognizes itself in the full destination address. If it is required to send a response, it gets the complete 4+4 address of the sender from the packet. The response packet is routed across realm boundaries in the same way as the forwarded packet. Realm gateways **B** and **A** will swap source and destination addresses, respectively. The addressing fields of the packet will be,

$$\begin{matrix}\mathbf{Y}\\ \mathbf{B}\end{matrix} \rightarrow \begin{matrix}\mathbf{A}\\ \mathbf{X}\end{matrix}, \quad \begin{matrix}\mathbf{B}\\ \mathbf{Y}\end{matrix} \rightarrow \begin{matrix}\mathbf{A}\\ \mathbf{X}\end{matrix}, \quad \begin{matrix}\mathbf{B}\\ \mathbf{Y}\end{matrix} \rightarrow \begin{matrix}\mathbf{X}\\ \mathbf{A}\end{matrix}$$

as the packet travels through realm ***B***, the level 1 realm and realm ***A***, respectively.

This swapping procedure ensures that private IPv4 addresses are never used in the outer header outside the private realm they belong to. Therefore, IPv4 routers in both private and public realms only see the addresses on which they have routing information. When realm gateways swap source or destination addresses, they also set the `SPos` or `DPos` fields accordingly. Realm gateways also decrement the `TTL` field and recalculate the `Header Checksum 1`. In this manner TTL scoping remains unchanged with each router and realm gateway counting as one hop.

4.2 Routing between Public and Private Realms

In what follows, we describe routing between a node in the public Internet and a node in a private address realm (e.g., node **C** and node **X** in Figure 2).

If both nodes are 4+4 capable then they can use the full 4+4 header to communicate. The 4+4 address of node **C** is **C.0**. Here **0** refers to the all-zero IP address 0.0.0.0. The realm gateway performs exactly the same address swapping operation as described in the previous section. The addresses in a packet sent from **X** to **C** are shown in Figure 2 next to the dotted arrows. On the return path, the fields are the same with the source and destination exchanged. If any of the nodes is not 4+4 capable, for example when no 4+4 address is available from the DNS, then IPv4 and traditional address translation is used. Note that here we use an existing mechanism (i.e., address translation) as a transition tool to enable communication between upgraded and non-upgraded hosts in different realms.

4.3 ICMP Message Routing

ICMP messages provide important error feedback, control, and debugging functions that are an integral part of the Internet Protocol suite. ICMP messages are used in 4+4 conceptually the same way as in IPv4. The current definition of the ICMP protocol is used unchanged, although it is possible to restructure the ICMP protocol as done in the case of IPv6. ICMP messages generated by 4+4 end-hosts, such as Echo or Port Unreachable are addressed and routed just like any other 4+4 packet. End-hosts include the full 4+4 header plus 8 bytes of the original packet. This allows for the inclusion of protocol and port numbers.

Some ICMP messages generated by routers in response to packets not addressed to them require special attention from realm gateways and 4+4 hosts. These ICMP messages include Redirect, Host and Network Unreachable, Fragmentation Required, Time Exceeded, Parameter Problem and Source Quench messages. Other messages, such as Router Discovery messages or Echo and Echo Reply are either always sent inside subnets and thus are not affected by 4+4 or are end-to-end messages. Since routers along a path may only be IPv4 routers, the ICMP messages may not be sent to the original source, but to the outer IPv4 source address of the packet. The following paragraphs discuss this issue in more detail.

Assume that node **A.X** sent a packet (packet p) to node **B.Y**, but the packet cannot be delivered and router **R** generates an ICMP message in response. If **R** is in realm ***A*** then the outer source address field of p contains the level 2 address of the source node **X**. In this case, the ICMP message will reach the source node without any special treatment. The source node is able to recognize that the ICMP message is sent in response to an 4+4 packet by looking at the ICMP payload.

If the router **R** is in the public address realm, then the ICMP message will be sent to the realm gateway **A**. This realm gateway determines that the packet included in the ICMP message is a 4+4 packet and converts the IP header of the ICMP message to 4+4. The destination address will be **A.X** (swapped) copied from packet p included in the ICMP message. The source address will be **R.0**. This allows the original sender to identify the router that generated the ICMP message.

If the router is in realm ***B***,or in any private address realm different from ***A***, then the ICMP message will be routed toward the realm gateway **A**. However, because the ICMP packet is an IPv4 packet containing a private source address, it needs address translation and will be captured by the realm gateway at the realm border, (i.e., **B** in our case). Recognizing the ICMP message as a response to a 4+4 packet, the realm gateway converts the ICMP message header into 4+4, with the destination address **A.X** unswapped and source address **B.R** unswapped and forwards it into the public address realm. This packet is then routed to node **A.X** as a regular 4+4 packet.

4.4 Routing Configuration

Realm operators may use their own addressing plan inside a private realm. Nodes need not renumber their level 2 address parts when the level 1 address parts of the realm changes, e.g., the realm switches providers or a realm gateway is added or removed. It is also possible to partition a private realm by separating the two networks and changing the level 1 parts of the two partition differently.

Multihoming private realms may be configured in several ways. One extreme is to assign a different IPv4 address to each realm gateway. Any of these addresses can be used as the level 1 part of the 4+4 address for hosts inside the realm. In this case, the hosts need to be configured with all or some of the possible level 1 address parts. By selecting the level 1 address part, nodes can effectively select the ingress realm gateway for the traffic addressed to them. An additional benefit of such a realm gateway configuration is that the address of realm gateways can be easily aggregated in

the core of the Internet. This, however, comes at the expense of resilience. If a realm gateway or its provider fails, no other realm gateway can take over.

To overcome this problem, realm gateways can advertise and use the address of another gateway in addition to their own. If one realm gateway fails, traffic addressed to it will be rerouted to another realm gateway advertising its address. The extreme case is when all realm gateways are configured with the same address, resulting in a single possible level 1 address part for the hosts inside. Since realm gateways hold no per-flow state, if one of the realm gateways fails, another one can take over in forwarding the flows for the failed realm gateway. This feature of realm gateways opens the way for a number of further multihoming setups that are outside the scope of our current work.

We note that the situation discussed above is very similar to the issue of multihoming in IPv4 or IPv6. With 4+4, however, the internal addressing of the realm can be hidden from the public Internet. This reduces both the amount of routing information (i.e., the number of prefixes) and the number of changes needed in the core of the Internet. In addition, there is no need to request a new, possibly separate, address block whenever a realm grows beyond its current allocation.

5. TRANSITION TO 4+4

Technically the transition to 4+4 represents a straightforward, stepwise upgrade of NATs and hosts. To upgrade a private access realm at least one of its NATs must be upgraded first to act as a realm gateway. The new functions required include (1) the ability to swap addresses in 4+4 headers; (2) the conversion of ICMPv4 message headers; and (3) the participation in routing and filtering of private addresses. The last function is already part of many NATs today.

Once the private realm has at least one realm gateway, hosts inside the realm can start upgrading. To upgrade a host, its operating system must be augmented with the ability to send and receive 4+4 packets. Auxiliary protocols, such as DHCP, ARP, RARP, router discovery, etc., need not be modified. Similar to IPv6, some applications also need to be upgraded at least to handle larger addresses. Bump-in-the stack address translation [19] developed for IPv6 might be used allowing applications that do not carry IP addresses in payloads to run unchanged.

The DNS itself need not be modified if 4+4 addresses are stored as two type A records, as discussed in Section 3.2. However, if a particular upgraded host needs a domain name its address needs to be included in the DNS zone files and made available to the outside world.

5.1 Transition Incentives

Transition is likely to be started by networks that have an insufficient amount of IPv4 addresses. These may be existing networks using NATs or new networks that find the acquisition of many IP addresses too costly. This is part of the incentives, as transition is directly motivated by the problem the 4+4 architecture aims to solve, i.e., address depletion.

Upgraded hosts immediately gain access to all other 4+4 nodes globally regardless of location. This immediately enables several new applications, such as file sharing or peer gaming that are complicated today because many hosts and therefore users are behind NATs. Upgraded hosts use IPv4 inside a realm and IPv4 plus NATs outside a realm to communicate with non-upgraded hosts, just as they did before the upgrade. This means that hosts can be upgraded one-at-a-time without impacting other hosts. Also, transition builds on the popularity of NATs by using a similar network setup.

As the number of upgraded hosts increases in private realms, hosts in the public address realm also have a growing incentive to upgrade to 4+4. They can do so at any time individually. As a result, upgraded nodes gain access to hosts in other already established private 4+4 domains, (e.g., to run peer-to-peer applications). At this point IP transparency between such hosts is accomplished. End-to-end transparency is restored in the Internet to the extent of 4+4 deployment. When, if ever, deployment becomes complete then IP address transparency will also be fully restored. Note that this would be accomplished without replacing or even reconfiguring routers. Backbone operators, for example, may remain completely unaffected. Of course, if a router has not been upgraded, its control plane cannot be reached from outside the address realm, (e.g., for management purposes). But because routers are usually managed from within the same domain this problem may not be serious. In addition, a control software upgrade of the router would solve this problem.

As the number of hosts reachable via 4+4 increases, organizations that had no NATs before may see the benefits of setting up their own address realms. By doing so, they have the opportunity to install new equipment without obtaining more public IP addresses. The existing nodes need not be renumbered; public addresses may remain to be used inside the realm even for communicating with the outside world. In addition, the routing information of the realm may be hidden from the outside world. The address translation function of realm gateways is required only for communicating with IPv4 only hosts. As transition progresses it will be invoked less frequently and can be completely removed once the majority of the nodes have transitioned. This way 4+4 provides a way out of using NATs.

One benefit of the above transition process is that the upgrade of individual realms is de-coupled and may happen at different paces. The most complex transition tool is the NAT itself. This provides communication between hosts if no native IPv4 or 4+4 path is available. There is no transition mechanism to allow communication between certain hosts (e.g., IPv4 only hosts in different private address realms). However, such a possible lack of reachability would not discourage starting the transition, as it is already in place today. In contrast, it provides an incentive as transition provides the missing reachability.

5.2 4+4 and IPv6

The major benefits of 4+4 over IPv6 are its backwards compatibility, the ease of transitioning, and the isolation of realms. Due to the backward compatibility of the packet header and addressing, the transition can be gradually started, gradually evolving to the new 4+4 architecture. There is no need for temporary transition mechanisms (such as tunneling, tunnel brokers, 6to4, 6over4 or DSTM, as discussed in [29]), all new mechanisms are final. There is no need for a new addressing plan, dual routing, new network management tools, new routing protocols, or new routers. The transition requires little new software and minimal changes to a running network. Full backward compatibility is maintained even when all hosts have transitioned. 4+4 and IPv4 only hosts/networks can coexist without new overhead. 4+4 immediately provides a large address space for realms without introducing new routers and a new

protocol. This may be beneficial for larger organizations where most of the traffic is local.

On the other hand, many of 4+4 features (e.g., the header format) include a number of design compromises necessary for backward compatibility. Therefore, in comparison to the IPv6, 4+4's design is not based on a clean slate. The extended address space is substantially smaller than that of IPv6. Applications placing IP addresses in payloads also need to be modified if they are to be used between realms. This, however, is unavoidable if the address space is truly extended, since the number of possible destinations may not fit into 32-bits. If operators and users choose to undergo the IPv6 transition, 4+4 is not needed. However, if IPv4 and NATs prevail, 4+4 provides a plausible solution to the known problems discussed in this paper.

6. IMPLEMENTATION

We implemented 4+4 under the Linux operating system using kernel version 2.4.18. The 4+4 source code is publicly available from the web [32] for experimentation. One of the key goals driving the 4+4 implementation is a minimal impact implementation on the kernel and applications. Hence, no modification has been made to the kernel itself. All the 4+4 functionality is provided in the form of a kernel module and an accompanying user space daemon that can be loaded and unloaded to/from a running kernel. As a result of our minimum impact implementation we sacrificed some performance. The implementation includes both the end-host and the realm gateway functionality. It does not contain NAT functions itself but interworks with the standard Linux NAT. The kernel module and userspace daemon comprise roughly 2200 and 1200 lines of C code, respectively. In what follows, we describe our implementation.

6.1 Peer Identifiers

To implement socket network programming with a new address space, the obvious solution is to define a new address family as is the case with most IPv6 implementations. This, however, requires the revision and porting of existing networking code to the new address family. In addition, applications need to be modified. To achieve backward compatibility with existing applications and to minimize the implementation work, we adopted a different strategy. The end-host functions are implemented using a transparent protocol translation mechanism similar to [19]. 4+4 addresses are mapped to 32-bit *peer identifiers* that are of local significance only and are taken from a yet unused block of the IPv4 address space. By default the block 1.0.0.0/8 is used. Our implementation transparently translates between IPv4 packets with peer identifiers and 4+4 packets. Applications are only presented with peer identifiers. The mapping between peer ids and 4+4 addresses is established by incoming 4+4 packets and DNS queries. These mappings are automatically timeout if they remain unused by incoming or outgoing packets for a period of time. In addition to the translation function, an API is defined and implemented that provides functions to establish, query, and remove peer mappings. In this manner, the full functionality is available to 4+4 aware applications without compromising backward compatibility.

The 4+4 kernel module is configured with the list of level 1 and level 2 addresses of the node. Configuration is automated: if an interface has a private address it is considered level 2 by default, and level 1 otherwise. If a node has no level 2 address then `0.0.0.0` is used assuming that it is in the public Internet. If a node has no level 1 address then the DNS is queried for the hostname to obtain the level 1 address. The kernel module operates using the Netfilter architecture of the Linux 2.4 kernel [30]. More information on the 4+4 implementation can be found on the project webpage [32].

6.2 DNS Translation

To maintain compatibility with the deployed DNS infrastructure, 4+4 addresses are stored as discussed in Section 3.2. The 4+4 kernel module intercepts incoming DNS reply messages from type A queries if the reply contains no valid answer. Such packets are passed to the 4+4 userspace daemon, which prepends the existing domain name with "`l1.`" and "`l2.`" and then performs two type A queries on the revised names. If both the queries are successful, then a new peer id is allocated to the 4+4 address. This peer id is then placed in the original DNS reply packet, which is then passed back to the kernel and from there on to the querying application. As a result, if a host has a 4+4 address then a querying application will receive a host address that is a peer identifier corresponding to the 4+4 address of the host.

Reverse DNS queries are also captured and translated in this manner. For example, a query to `12.0.0.1.in-addr.arpa` is translated into a query to `2.0.168.192.131.67.59.128.in-addr.arpa`. Similarly, replies are translated back. As a result, applications have total DNS transparency, (e.g., ping and tcpdump are able to show the DNS names for 4+4 hosts of which they only know the peer identifiers).

6.3 ICMP Translation

Certain ICMP messages may carry IP packets in their payload. Upon receiving such ICMP packets, the kernel module checks if the packet in the payload is a 4+4 packet. If this is the case, then the 4+4 header is removed and the source and destination addresses are translated to peer identifiers. This allows the kernel to match ICMP error messages to the sockets associated with peer identifiers. If the host is sending an ICMP message in reply to another packet, the kernel module checks whether the included packet's source or destination address are peer identifiers. If so, then a 4+4 header is added to the included packet with the appropriate 4+4 addresses. This also ensures that peer identifiers are never exposed on the wire at least for ICMP.

One problem with ICMP messages generated by routers is that the ICMP protocol [2] mandates the inclusion of the original IPv4 header including options plus 8 bytes of the payload only. In case of a legacy IPv4 router generating an ICMP reply in response to a 4+4 packet, this excludes the transport protocol and port information. As a consequence, the source host cannot identify the transport connection for which it received an ICMP message. Some router implementations (e.g., Linux) return more than 8 bytes, however. To upgrade most routers to do this would of course solve this problem. This problem, however, is not as serious as it first seems because most ICMP messages generated by routers correspond to all transport sessions at the destination host. With this in mind, our current implementation delivers the signal to all relevant transport identities, that is, to all sockets with the same source and destination addresses. In other cases, such as when the TTL is exceeded or when a parameter problem message is generated this action is not meaningful, however, sockets usually ignore these messages. The fragmentation needed ICMP message needs special handling because they are part of the Path MTU discovery process [3]. In response to this the kernel module decreases the reported MTU size in fragmentation needed ICMP packets sent in response to 4+4

packets. This forces the transport to generate smaller packets that fit the path requirements even for 4+4 packets. The kernel in a more integrated 4+4 implementation could directly take the size of the 4+4 header into account when calculating TCP segment sizes.

6.4 Realm Gateway Operation

If the kernel module determines that a host has interfaces with both private and public addresses then it will start operating as a realm gateway, assuming that this mode is not disabled. The Linux kernel routing features and NAT implementation are used to provide the actual packet routing and NAT functionality, respectively. 4+4 packets are allowed through the NAT unmodified. If the private realm is not a "stub" realm, then additional rules are needed to protect packets with public source and destination addresses from being translated by the NAT. In the realm gateway mode, the kernel module detects 4+4 packets crossing the realm boundary and performs address switching. ICMP packets are handled as discussed in Section 4.3.

6.5 Configuration Tasks

The configuration of an end-host is essentially the same as an IPv4 host. There are only two additional tasks that need to be considered. First, the host must know if it is in a private or public realm. Second, if it is in a private realm, then it needs to be configured with the level 1 part(s) of its address. The first piece of information can reliably be estimated from the IP address of the host. The second part can be read from the DNS because the host is able to communicate using IPv4 even without its full 4+4 address. Configuring the DNS is straightforward to do with only the two type A records needing to be added to represent the 4+4 address.

Configuring realm gateways requires somewhat more effort. First, a fully functional NAT must be maintained during the transition period. Second, it needs to specified which interfaces of the realm gateway belong to public or private realms, and what are the level 1 address parts of that realm. Third, routing and route filtering must be set up such that routing information on public IPv4 addresses are propagated both in and out of the gateway, while routes toward private addresses are filtered out. This can be achieved using the current routing tools, either by using a different routing protocol in the two realms (e.g., BGP/OSPF or OSPF/RIP) or using OSPF with different areas.

7. EXPERIMENTAL RESULTS

In this section, we discuss our results from the evaluation of our 4+4 implementation using local and wide area Internet experiments. We first discuss some experiences with the resilience of our approach. In particular, we demonstrate 4+4's resilience to realm gateway failure. Following this, we present some performance results of our implementation. Finally, we discuss a set of experiments and results that test application compatibility with using 4+4.

7.1 Resilience

To test the robustness of 4+4 to realm gateway failure, we constructed the topology shown in Figure 3. The topology contains a public and private realm, two routers (`R1` and `R2`), two realm gateways (`RGW1` and `RGW2`) and two end-systems (`taygeta` and `galaxy`). The routers are legacy, unmodified IPv4 routers. We used the gated-3.6 routing software in the routers and gateways. Two OSPF routing areas are configured, the backbone consisted of `R1`, `RGW1` and `RGW2`, whereas an additional OSPF stub area consisted of `RGW1`, `RGW2` and `R2`. Networks in the stub area are private addresses and are filtered by the realm gateways. Realm gateways also advertised an extra address `128.59.67.213` that is exclusively used as the level 1 part of all 4+4 addresses in the private realm.

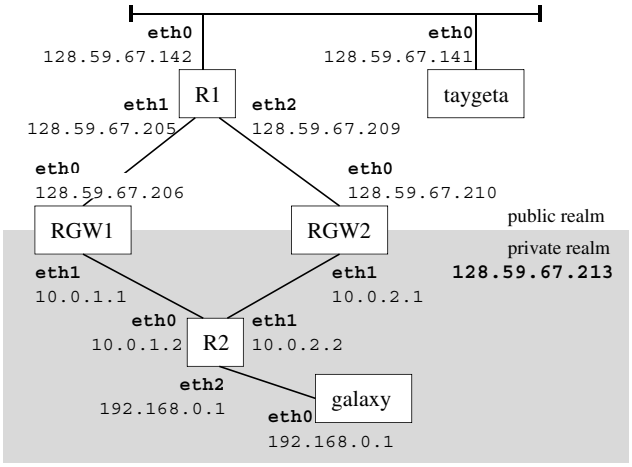

Figure 3: Test network 1 topology (multihoming private realm)

We started a long TCP data transfer from `taygeta` to `galaxy` using the `ttcp` utility. The initial routing preferred `RGW1` in both directions, a traceroute listing of the path from `tagyeta` to `galaxy` is shown in Table 2. The listing is generated using a modified version of the traceroute utility, as discussed in more in detail in Section 7.3.

At around two seconds after the start of the TCP connection we disconnected both cables of `RGW1`. The routing software did not receive a link-layer disconnection signal, so the failure can only be detected by missing Hello packets. Using the defaults of `gated`, the connecting routers declared `RGW1` down after 40 seconds. At this point the routers establish new routes and the TCP connection was resumed soon after. Since realm gateways hold no flow specific state, nothing prevented the session from continuing once routing stabilized. Apart from the temporary loss of connectivity, end systems were not affected.

7.2 Performance

Realm gateways have two 4+4 specific packet processing tasks. First, they swap the source and destination addresses in 4+4 packets forwarded between different realms. Second, they add a 4+4 header to certain ICMP packets.

To illustrate the costs associated with packet processing tasks, we performed a series of measurements in the Linux kernel. Figure 4 shows the results. The measurements are performed using an unloaded machine with one 1GHz Pentium III processor and 256 megabytes of memory.

Each group of bars in Figure 4 corresponds to a packet type. The height of the bar shows the processing time of the packet in the network layer. This is the time between passing the PRE_ROUTING and POST_ROUTING Netfilter hooks. For IP packets the time includes routing table lookup (usually a cached value), TTL decrement, IP option processing and fragmentation; none of the last two functions were exercised in our experiments. The first bar shows

```
traceroute4+4 from taygeta.ipv44.comet.columbia.edu (128.59.67.141.0.0.0.0)
1:  r1-eth0.comet.columbia.edu (128.59.67.142):  0.308ms 0.262ms 0.159ms
2:  rgw1-eth0.comet.columbia.edu (128.59.67.206):  0.274ms 0.264ms 0.196ms
3:  r2-eth0.ipv44.comet.columbia.edu (128.59.67.213.10.0.1.2):  0.365ms 0.611ms 0.343ms
4:  galaxy.ipv44.comet.columbia.edu (128.59.67.213.192.168.0.2):  0.445ms 0.630ms 0.370ms
```

```
traceroute4+4 from taygeta.ipv44.comet.columbia.edu (128.59.67.141.0.0.0.0)
1:  r1-eth0.comet.columbia.edu (128.59.67.142):  0.284ms 2.582ms 0.214ms
2:  rgw2-eth0.comet.columbia.edu (128.59.67.210):  0.414ms 0.292ms 0.241ms
3:  r2-eth1.ipv44.comet.columbia.edu (128.59.67.213.10.0.2.2):  0.437ms 0.669ms 0.321ms
4:  galaxy.ipv44.comet.columbia.edu (128.59.67.213.192.168.0.2):  0.444ms 0.676ms 0.385ms
```

Table 2: Traceroutes before and after the topology change

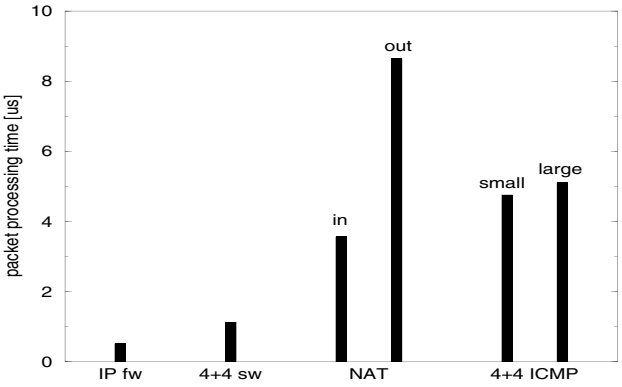

Figure 4: Network layer processing time during forwarding

the time for regular IP packet forwarding averaged over 1000 measurements. The second bar shows the forwarding time of a 4+4 packet, including the address swap operation. The third group shows the time of an address translation operation for packets entering and leaving the private realm, respectively. The time difference may be due to connection state management where packets leaving the private realm may establish entries. ICMP echo requests/replies are used to take the measurements. The last four bars show the processing time of router-generated ICMP messages that carry a 4+4 packet inside. In this case the realm gateway needs to insert a 4+4 header into the packet. In the Linux kernel, this usually involves a memory copy of the packet due to linear packet buffers. This explains the time difference between small (84-byte) and large (1428-byte) packets. We note that ICMP packets generated by end-systems do not fall into this category.

Although the above figures may certainly be different for different router platforms, we argue that 4+4 packet forwarding (i.e., the swap operation) is a simple operation that requires a small and constant number of steps. We believe it is amenable to hardware implementation in the fast path of routers. ICMP masquerading, on the other hand, is an operation that requires more changes to the packets and may be too expensive to implement in hardware. However, this poses no problem as ICMP processing can and should be rate limited. Realm gateways are free to drop excess ICMP traffic.

7.3 Applications

We experimented with a number of applications to test interoperability with 4+4. In general, applications and protocols that do not carry IP addresses in the packet payload work well with 4+4. The testbed used to experiment with Internet applications is a simplified version of the one shown in Figure 3 with parts of the network in New York in Budapest (see Figure 5) and separated by 17 hops.

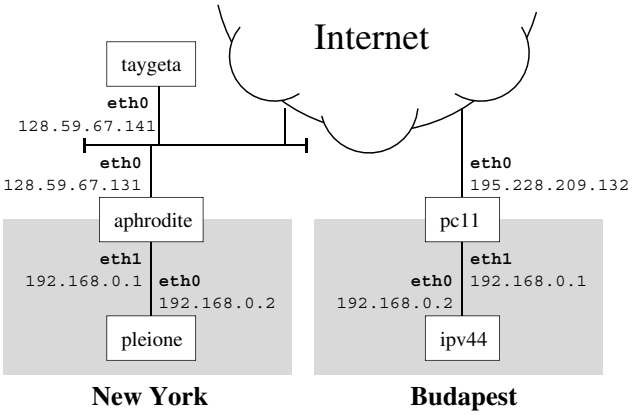

Figure 5: Test network 2 topology (wide area)

Hypertext Transfer Protocol (HTTP). The HTTP was tested by setting up a 4+4 compatible webserver both in a private and the public realm. In both cases we used the `apache` webserver. 4+4 aware clients were able to reach both webservers from anywhere using popular browsers including Netscape, Mozilla, Opera, Lynx. Webservers are reachable by specifying DNS names that point to 4+4 addresses.

Email Protocols. The SMTP, IMAP and POP3 protocols were tested by setting up e-mail forwarders in both realms using the `exim` utility. The popular `pine` e-mail program and the Netscape's mailer were used as user agents. The e-mail servers were specified using domain names. Again, 4+4 clients were able to download and send e-mail even if the 4+4 aware server was at a different realm.

Secure Tools. The `ssh` and `scp` tools were used on a regular basis to communicate between the machines. The host database of these tools can get mixed up if peer identifiers are used directly to identify targets hence the use of domain names is preferred.

File Transfer Protocol (FTP). FTP did not work as expected because it places the peer identifier in the protocol payload.

We also tested a number of network tools. The **`ping` utility** works unmodified. The only issue is that it displays peer identifiers in-

stead of full 4+4 addresses when pinging a 4+4 machine (e.g., "`PING pleione (1.0.0.2) from ...`").

The **`traceroute` utility** does not work, as it uses raw sockets and manipulatess UDP ports directly. The port information is usually not returned in the ICMP messages with 4+4. See Section 6.3 for a detailed explanation of the reasons for this problem. Therefore, we created a simple version of `traceroute` that only uses UDP sockets and does not code sequence number information into the port numbers. The only drawback is that two traceroutes performed on the same host at the same time toward the same destination by two different processes may get mixed up. The benefit is that since no raw sockets are used no root privileges are needed. In addition, as the 4+4 module passes incoming ICMP messages to all relevant sockets, the utility works well with 4+4 as well. We added a small piece of code to display the 4+4 numeric address, if the IP address seen is a peer identifier. Reverse DNS, however, was used unmodified. The new version of `traceroute` can be downloaded from [32] and was used to generate the listings shown in Table 2.

The **`tcpdump` utility** works well, but cannot decode 4+4 packets. To this end, we have written a small plugin to the `ethereal` utility to dissect 4+4 packets. The plugin is part of the 4+4 source code package that can be downloaded from [32].

8. CONCLUSION

In this paper, we have presented and evaluated 4+4, a new address extension architecture for Internet. 4+4 leverages existing private address realms and NATs, and represents an evolutionary approach toward Internet address extension. 4+4 offers a lightweight, well defined, incentive-driven transition process that can be incrementally deployed in the network today. Upgraded hosts immediately gain access to upgraded hosts in all other realms, while existing communication is not disrupted in any way. 4+4 is simple and retains the existing semantics of Internet names and addresses. Encapsulation is used as the main tool to maintain backward compatibility with existing routers that need not be modified. 4+4 does not employ address translation and provides end-to-end address transparency. Existing NATs are only used as a transition tool, with their use diminishing as 4+4 deployment progresses. In fact, removing NATs is one of the motivations for such a transition. However, the address isolation feature of NATs is retained by the 4+4 architecture.

We have evaluated the properties and performance of 4+4 based on local and wide area testbed experimentation, and discussed our experiences using a number of applications with 4+4. A number of configurations and possible pitfalls were explored and discussed. We found that 4+4 is easy to implement, scalable, introduces no single points of failure, and its performance look very promising. We believe that 4+4 provides one alternative should IPv6 be deemed too expensive or complicated for transitioning.

9. REFERENCES

[1] J. Postel, "Extensible Field Addressing," *Internet RFC 730,* May 1977.

[2] J. Postel, "Internet Control Message Protocol," *Internet RFC 792,* September 1981.

[3] J. Mogul, S. Deering, "Path MTU discovery," *Internet RFC 1191,* November 1990.

[4] Z. Wang, J. Crowcroft, "A Two-Tier Address Structure for the Internet: A Solution to the Problem of Address Space Exhaustion," *Internet RFC 1335,* May 1992.

[5] Minutes of the Address Extension by IP Option Usage BOF, *proceedings of 29th IETF meeting,* Seattle, April 1994.

[6] Minutes of the Address Lifetime Expectations working group, *proceedings of 29th IETF meeting,* Seattle, April 1994.

[7] P. Francis, "Pip Near-term Architecture," *Internet RFC 1621,* May 1994.

[8] K. Egevang, P. Francis, "The IP Network Address Translator (NAT)", *Internet RFC 1631,* May 1994.

[9] P. Francis "Addressing in Internetwork Protocols," PhD Thesis, *University College London,* available at `www.ingrid.org/francis/thesis.ps.gz`, September 1994.

[10] R. Hinden, "Simple Internet Protocol Plus White Paper," *Internet RFC 1710,* October 1994.

[11] Y. Rekhter, B. Moskowitz, D. Karrenberg, G. de Groot, E. Lear, "Address Allocation for Private Internets," *Internet RFC 1918,* February 1996.

[12] R. Hinden, "New Scheme for Internet Routing and Addressing (ENCAPS) for IPNG," *Internet RFC 1955,* June 1996.

[13] I. Castineyra, N. Chiappa, M. Steenstrup, "The Nimrod Routing Architecture," *Internet RFC 1992,* August 1996.

[14] C. Perkins, "Minimal Encapsulation within IP," *Internet RFC 2004,* October 1996.

[15] M. O'Dell, "8+8 – An Alternate Addressing Architecture for IPv6," *Internet Draft,* named as draft-odell-8+8-00, Work in progress, November 1996.

[16] S. Deering, R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification," *Internet RFC 2460,* December 1998.

[17] Z. Turányi, A. Valkó, "4+4: Expanding the Internet Address Space without IPv6," *Ericsson Internal Report,* August 1999.

[18] M. Crawford, A. Mankin, T. Narten, J. Stewart, L. Zhang, "Separating Identifiers and Locators in Addresses: An Analysis of the GSE Proposal for IPv6," *Internet Draft,* named as draft-ietf-ipngwg-esd-analysis-05, Work in progress, October 1999.

[19] K. Tsuchiya, H. Higuchi, Y. Atarashi, "Dual Stack Hosts using the "Bump-In-the-Stack" Technique (BIS)," *Internet RFC 2767,* February 2000.

[20] A. Gulbrandsen, P. Vixie, L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)," *Internet RFC 2782,* February 2000.

[21] B. Carpenter, "Internet Transparency," *Internet RFC 2775,* February 2000.

[22] M. Crawford, "Router Renumbering for IPv6," *Internet RFC 2894,* August 2000.

[23] T. Hain, “Architectural Implications of NAT,” *Internet RFC 2993,* November 2000.

[24] G. Huston, “To NAT or IPv6 – That is the question,” *Satellite BroadBand magazine,* available at the author’s page `http://www.telstra.net/gih`, December 2000.

[25] M. Holdrege, P. Srisuresh, “Protocol Complications with the IP Network Address Translator,” *Internet RFC 3027,* January 2001.

[26] M. Gritter, D. R. Cheriton, “An Architecture for Content Routing Support in the Internet,” *Usenix Symposium on Internet Technologies and Systems,* `http://gregorio.stanford.edu/triad`, March 2001.

[27] P. Francis, R. Gummadi, “IPNL: A NAT-Extended Internet Architecture,” *SIGCOMM’01,* August 2001.

[28] M. Borella, J. Lo, D. Grabelsky, G. Montenegro, “Realm Specific IP: Framework,” *Internet RFC 3102,* October 2001.

[29] The IETF Next Generation Transition (ngtrans) working group, `http://www.ietf.org`

[30] Linux 2.4.x Netfilter homepage, `http://www.netfilter.org`

[31] Z. Turányi, A. Valkó, “4+4,” *10th International Conference on Networking Protocols (ICNP 2002),* November 2002.

[32] The IP4+4 project webpage at `http://comet.columbia.edu/ipv44`

NETKIT: A Software Component-Based Approach to Programmable Networking

Geoff Coulson, Gordon Blair,
David Hutchison, Ackbar Joolia,
Kevin Lee, Jo Ueyama,
Antonio Gomes, Yimin Ye
Computing Dept.,
Lancaster University
Lancaster LA1 4YR, UK
+44 1524 593054
geoff@comp.lancs.ac.uk

ABSTRACT

While there has already been significant research in support of openness and programmability in networks, this paper argues that there remains a need for generic support for the integrated development, deployment and management of programmable networking software. We further argue that this support should explicitly address the management of run-time reconfiguration of systems, and should be independent of any particular programming paradigm (e.g. active networking or open signaling), programming language, or hardware/ operating system platform. In line with these aims, we outline an approach to the structuring of programmable networking software in terms of a ubiquitously applied software component model that can accommodate all levels of a programmable networking system from low-level system support, to in-band packet handling, to active networking execution environments to signaling and coordination.

General Terms

Management, Design, Reliability, Experimentation, Standardization.

Keywords

Programmable networking, components, reflection, middleware.

1. INTRODUCTION

There are steadily increasing demands for openness and programmability in today's networks. In particular, both network operators and users want to be able to dynamically introduce new mechanisms into the network with ease and convenience. Examples of such mechanisms are quality of service (QoS) elements like intserv/ diffserv/ MPLS/ RSVP/ RED/ ECN; in-band media-stream filters; network address translators; firewalls and other security mechanisms; and application-level routers (e.g. for multicast or peer-to-peer networking).

The requirement for openness and programmability is further underlined by the desire to dynamically deploy emerging services like ubiquitous computing, ad-hoc networking, dynamic private virtual networks, and e-Science Grids. Furthermore, there is an associated requirement for *manageability* of such mechanisms and services so that they can be flexibly configured (including deployment, instantiation and initialisation) and *re*configured (including run-time adaptation, extension, evolution and removal) with ease and convenience.

The view expressed in this paper is that, while there has already been significant research in support of such requirements, there remains a need for *generic programming model support* to facilitate programmable networking. Ideally, this support should be programming language-, platform-, and even *paradigm*-independent (see below) and should explicitly facilitate the management of both configuration and reconfiguration as defined above.

The approach we are pursuing is to apply the notion of *software components* [39] to the programmable networking environment. According to Szyperski [39], software components *i*) have formally specified interfaces, *ii*) are packaged and distributed in binary form, and *iii*) can be dynamically deployed in address spaces. Unlike other research that advocates a component-based approach (e.g. [28] and [37]) we envisage components being uniformly applied at *all* levels of the programmable networking environment from fine-grained, low-level, in-band packet processing functions, to high-level signaling and coordination functions. In outline, we envisage on-demand component loading/ unloading and binding/ unbinding services as the basis of both configuration and reconfiguration.

The remainder of this paper is structured as follows. First, §2 provides an overview and analysis of the field of programmable networks. Next, §3 presents our generic component-based approach to programmable networking together with a discussion of the potential benefits of the approach. Then, in §4, we discuss current design and implementation work in our recently initiated NETKIT project that follows the component-based approach. As the project is at a relatively early stage, the discussion is in terms of work-in-progress rather than definitive results. Finally, §5 discusses some related work (in addition to that surveyed in §2), and §6 presents our conclusions.

2. PROGRAMMABLE NETWORKING RESEARCH

2.1 The Design Space of Programmable Networking

The design space of programmable networking can be broadly represented in terms of the (highly abstract) reference architecture depicted in Figure 1.

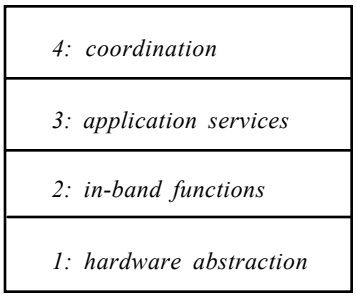

Figure 1: A reference architecture for programmable networking.

In this architecture, a *hardware abstraction* stratum (we use the term 'stratum' rather than 'layer' to avoid confusion with layered protocol architectures) contains the minimal operating system (OS) functionality (e.g. threads, memory allocation, and access to network hardware) that must be available on any participating node (e.g. router) to support higher-level network programmability. Services in this stratum typically try to mask underlying hardware heterogeneity so that, say, a standard PC-based router and a specialised programmable router (e.g. a router based on the Intel IXP1200 [23] network processor which provides multiple processors and distributed/ hierarchical memory arrays) will look as similar as possible to the higher strata. Furthermore, the nature of the stratum 1 services largely determines the QoS (e.g. predictability, throughput and latency) capabilities of programmable networking software in the higher strata.

Second, an *in-band functions* stratum comprises packet processing functions (e.g. packet filters, checksum validators, classifiers, diffserv schedulers, shapers, etc.) that touch all or most packets. As these functions are inherently low-level, in-band, and fine-grained, this is a highly performance critical area in which machine instructions must be counted with care.

Third, an *application services* stratum comprises coarser-grained 'programs'—in the active networking execution-environment sense [1]—that are less performance critical and act on pre-selected packet flows in application specific ways (e.g. per-flow media filters).

Finally, a *coordination stratum* includes or supports out-of-band signaling protocols that perform distributed coordination (e.g. configuration, reconfiguration) of the lower strata. Examples are RSVP, or protocols that coordinate resource allocation on a set of routers participating in a dynamic private virtual network, as employed by systems like Genesis [6], Draco [24] or Darwin [10].

2.2 Current Paradigms

Historically, there have been two main paradigmatic approaches to the provision of openness and programmability in networks: First, in the *active networking* paradigm (see, e.g., [1], [15], [34], [16], [17], [18], [19]) 'active packets' called carry programs that execute on 'active nodes', often in a Java-based execution-environment. Second, in the *open signaling* paradigm (see, e.g., [6], [10], [24]), routers export 'control interfaces' through which they can be remotely (re)configured by out-of-band, application specific, signaling protocols. More recently, a third approach—we'll call it out-of-band active—has become popular (see, e.g. [7], [11], [13], [22], [28]). In this approach, downloadable modules are dynamically installed onto routers through some (often unspecified) out-of-band mechanism. These systems vary in their support for kernel vs. user space modules, and whether or not in-band functions can be reconfigured.

Overall, active networking is the most dynamic of the three approaches and can operate on the finest time scales. However, it is not as easy to deploy as the other approaches, is perceived as more prone to security threats, and tends to be language specific (often Java). While being coarser grained and less dynamic, the open signaling approach is typically easier to deploy (especially for complex services like dynamic private virtual networks), easier to secure, and typically performs better than Java-based active networking systems (especially at the level of fundamental QoS elements like intserv or diffserv). The out-of-band active approach is between the two classic approaches in terms of both deployability and security vulnerability.

Combining the above analysis with that of §2.1, it is interesting to observe that much programmable networking research addresses only a *subset* of the concerns implied in Figure 1. In particular:

- active networking research tends to focus on stratum 1 (e.g. the Scout implementation of NodeOS [34], [35]) and stratum 3 (the performance requirements of stratum 2 typically cannot be met in a Java-based execution-environment, and stratum 4 coordination is typically left to the 'application')[1];
- open signaling approaches focus mostly on strata 2 and 4 (typically, router control interfaces enable stratum 2 configurability but do not support stratum 3 functions and completely hide stratum 1);

It can also be observed that most out-of-band active systems address only stratum 2 and/ or stratum 3 concerns (sometimes stratum 1 is partially addressed as well). For example, the Click modular router [28], the NetBind component binding system [7], Washington University's pluggable router framework [13], and the IEEE P1520 router component model [22] are all targeted at stratum 2. (Click employs a fine grained C++-based component model with flexible support for the configuration of packet scheduling, route lookup and queue drop modules etc.; NetBind is similar in concept but is lower-

[1] Some active networking implementations (e.g., the Scout NodeOS implementation reported in [35], and the Lancaster work on LARA++ [11]) do address stratum 2 as well as strata 1 and 3. However, there is typically a distinction drawn between an in-kernel 'fast path' environment for 'default' packet handling, and a less efficient, user-space, environment for configurable/ extensible packet handling code. While the performance deficit is not so great as in Java-based execution environments, it remains true that the custom path suffers in terms of performance while the fast path suffers in terms of flexibility. It is not so necessary to face this trade-off in the open signalling approaches discussed next.

level and targeted at network processors; the Washington work is a framework for pluggable per-flow modules in the NetBSD environment; P1520 is working towards a standardised, language-independent, component model for modular routers.) Slightly more generally, the Knit system [37] supports stratum 2 (and stratum 1 also) in the form of a component model that has been used for both OS and in-band packet handling functions. However, Knit is supported only on conventional workstation architectures, not on specialised programmable routers. The VERA extensible router architecture [27] supports stratum 1 and stratum 2 on a wider range of router architectures but offers a far less general and flexible component model.

Overall, what appears to be missing from the state-of-the-art is a generic framework that is *both* paradigm-independent *and* equally applicable to all strata of the reference architecture.

2.3 Run-time Reconfiguration

It can also be strongly argued that support for *run-time reconfiguration* is inadequately addressed by current research. For example, while the above-cited component models support the initial configuration of components, none of them explicitly support the subsequent reconfiguration of a running system (e.g. to accommodate newly discovered services in a ubiquitous computing environment; to reconfigure an ad-hoc network; or to adjust the resources allocated to a dynamic private virtual network). Furthermore, systems that *do* allow reconfigurability (e.g. most active networking systems) still fail to adequately support the *management of system integrity* over reconfiguration operations (e.g. ensuring that firewall updates are applied universally and consistently; or that a change in a source media-filter type is accompanied by a compatible change at the sink; or that allocating more resources to one dynamic private virtual network does not lead to starvation in another).

There has been some work on the use of *reflection* to address such management related issues. For example, [21] describes reflective support for checking the integrity of coordination/ control code being downloaded into an execution environment, and [40] further supports some degree of dynamic reconfiguration of downloaded control code. More recently, [46] supports reconfiguration through dynamic linking, but not in the context of a principled reflective component model. On the other hand, [47] provides a reflective component model but focuses on a flexible deployment architecture rather than on fine-grained reconfiguration.

However, this work is again *partial*; it typically addresses only execution environment and coordination strata concerns (i.e. strata 3 and 4 in the reference architecture), and is programming language specific (Java).

2.4 Summary

Overall, we argue that while there has been significant research in programmable networking, most work to date has focused on specific and limited areas of the overall design space. This lack of recognition of the 'big picture' has led to a proliferation of programmable networking solutions that are on the one hand partial and on the other hand incapable of being easily combined to produce more comprehensive solutions. More specifically, there has been insufficient attention paid to the development of 'integrated' solutions that are capable of offering:

- a language-, platform- and paradigm-independent programming model that can be uniformly applied in all four strata of the reference architecture without unacceptable compromise (e.g. in terms of performance), and
- flexible support for both the configuration (e.g. deployment, instantiation, initialisation) and run-time reconfiguration (e.g., adaptation, extension, evolution, removal) of mechanisms and services in all strata.

Our approach to the provision of such an integrated solution is detailed in the rest of this paper.

3. TOWARDS A COMPONENT-BASED APPROACH TO PROGRAMMABLE NETWORKING

3.1 Support for Components

3.1.1 A Component-Based Computational Model

To realise the software component concept in the programmable networking environment, we first need a *component-based computational model* that satisfies the particular demands of that environment. As the basis of NETKIT, we employ an abstract, minimal, generic, language-independent, component-based computational model that is derived from our previous work on component-based middleware [8].

The key concepts embodied by the computational model are: *component*, *interface*, *receptacle*, *binding*, and *capsule*. These are illustrated in Figure 2 which shows two components inside a capsule (dotted lines). The component at the top left supports two interfaces (small circles) and one receptacle (small cup). This receptacle is bound to one of the interfaces of the bottom right component.

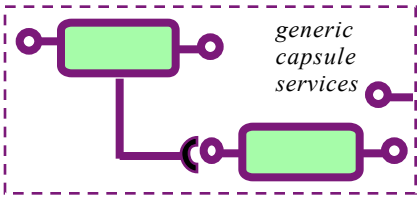

Figure 2: The component-based computational model.

Components can support any number of interfaces and receptacles. Interfaces are strongly typed and consist of a set of datatype definitions and operation signatures; they are defined in a programming-language-independent interface definition language such as OMG IDL or Microsoft IDL (we use OMG IDL). Receptacles are 'anti-interfaces': whereas an interface expresses a unit of service *provision*, a receptacle expresses a unit of service *requirement*. (The term 'receptacle' is also employed by the CORBA Component Model [36]. The concept itself appears in various other component models under various names.) Receptacles are used to make explicit a dependency of one component on another. For example, if a component relies on a service of type *S*, it would declare a receptacle of type *S'* that would be bound at run-time to an interface instance of type *S* (which would be provided by some other component). The fact that dependencies are explicitly represented means that when a component is dynamically loaded it is possible to determine what other components and

interfaces must be present for it to work correctly. This is a crucial enabler for 'third-party' configuration and dynamic reconfiguration of component topologies.

Bindings are associations between receptacles and interfaces that reside in the same capsule (and are type compatible). They are assumed to be implemented minimally and with negligible or low overhead. The viability of the component model at fine granularities, particularly in demanding areas like in-band packet processing, is heavily dependent on the degree of this overhead which must be comparable to, or less than, the overhead of a function call in a language like C. It is important to note that, as bindings are abstract, there is no prescription of a particular underlying implementation. This fact is heavily exploited in our current implementation work, as discussed in §4.2, which employs multiple alternative implementations of binding.

Finally, *capsules* provide a run-time environment for a set of component instances that are mutually participating in bindings. Capsules are typically, but not necessarily (see section §4.2 below), implemented as address spaces. The central role of capsules is to provide generic services for dynamically loading and unloading components, and for creating and destroying bindings. As well as being available from within the capsule in a third-party manner, these services can be made available from outside the capsule to support *external* third-party loading and binding[2]. This is useful to enable bootstrapping and third-party management of capsules (possibly from a remote site). In the programmable networking environment, it must additionally be possible to render the (un)loading and (un)binding of components subject to *security constraints* (i.e. to constrain who has rights to deploy, use, bind, reconfigure, etc.) and *safety constraints* (i.e. limits on what components can do to their host node). While *policy* in these areas is clearly application dependent, basic security and safety *mechanisms* should be built into the component model itself (e.g., the capsule) wherever possible and appropriate.

Although they may appear superficially similar, capsules are very different from active networking 'execution environments' (e.g. [ANTS,01]). Capsules are a minimal bootstrapping facility and are neutral with respect to programming language and API (beyond the very minimal load/ unload, bind/ unbind 'meta-API' outlined above). Capsules form the basis of a generic component model that, in turn, serves as the basis for any desired programmable networking functionality (including the construction of execution environments, which in our architecture would be implemented as component frameworks—see §3.3).

3.1.2 Portability Considerations

Portability is a crucial issue for us; we need to deploy the component model on a wide range of hardware platforms, from standard PCs to a variety of specialised programmable router platforms.

The obvious approach to portability is to define a single 'standard' OS-level API that all hardware platforms must support. Unfortunately, this simple approach has major drawbacks. First, some platforms will suffer sub-optimal performance because the abstractions employed by a necessarily 'lowest common denominator' API will tend to map better to some platforms than others (e.g. abstractions that implicitly assume shared memory may be hard to implement efficiently in a distributed memory environment). Second, a standard API precludes the exploitation of specialised platform-specific hardware (e.g. the availability of 'microengine' processors—as on the Intel IXP1200—or direct access to I/O ports). And, third, the work involved in porting a comprehensive API is likely to be significant in itself.

To avoid these difficulties, we adopt an approach to portability that is strongly influenced by radical micro-kernel architectures like L2 [30] and Think [41]. More specifically, we define two levels of portability. The first level comprises the component model itself; this is kept as simple as possible, and relies on an absolute minimum of system support so that it can be readily ported. Essentially, all that is needed is a sufficient implementation of capsules, including the capability to load/ unload executables and make/break bindings. The second level, which comprises all further system-oriented and hardware specific functionality (stratum 1) is then implemented *in terms of the component model itself*. This includes platform specifics like network card APIs, as well as generic OS-oriented APIs for threads, buffers, inter-capsule communication, etc.

A key benefit of this approach, apart from facilitating porting, is that only those stratum 1 services that are *actually required* on any particular platform need be ported and deployed. At the same time, thanks to the component model's explicit representation of dependencies, services that are not initially needed can be brought in later if requirements change/ evolve.

3.2 Reflection: Basic Support for Reconfiguration

Beyond the capability to construct component configurations (as provided by the basic component model outlined above), there is the further requirement, identified in §2.3, to support run-time *reconfiguration* of components in a generic and principled way. This breaks down into two areas: *adaptation* (to change behaviour along dimensions that are foreseen at deployment time), and *extension* (to add new behaviour unforeseen at deployment time). Furthermore, there is an associated requirement to first be able to *inspect* current configurations as the basis of subsequent adaptation and extension.

We employ the notion of reflection [31] to support such inspection, adaptation, and extension. Essentially, reflection is a pattern for opening up 'black box' systems to inspection, adaptation and extension. In abstract terms, this is achieved by invoking a so-called *meta-interface* on the system (see figure 3) to yield one or more *meta-models* of the system that can be inspected, adapted and extended. A defining feature of reflection is that these meta-models (which are said to reside at the *meta-level*) relate to the underlying system (referred to as the *base-level*) in a *causally connected* manner. This means that a change made to a meta-model implicitly causes a corresponding change in the underlying system, and vice versa. As an example, a topological graph-like meta-model (as in figure 3) could be used to explicitly represent the implicit

[2] This implies that the capsule's loader must include simple protocol support for remote access. We provide a 'bootstrapping' TCP/IP implementation on each NETKIT enabled router for this purpose. To provide more comprehensive remote access, our approach, based on our previous work [8], would be to deploy CFs in the capsule that provide appropriate middleware functionality.

topology of a composition of components—e.g. a fine-grained component-based packet forwarder à la Click [28]. Thanks to causal connection, when the graph is manipulated, e.g. by deleting or redirecting an arc, the underlying configuration is changed correspondingly (e.g. in terms of bindings).

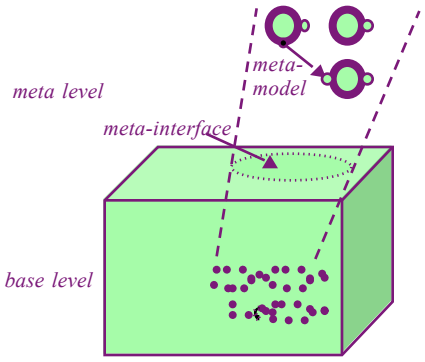

Figure 3: The concept of reflection.

Examples of reflective meta-models that we employ in our current work are as follows:

- an *architecture* meta-model which provides inspection, adaptation and extension of component compositions (as above),
- an *interception* meta-model which supports pre- and post-method call interception of invocations being made across bindings,
- an *interface* meta-model which supports the navigation of interfaces and receptacles on a component (cf. MS COM's 'IUnknown' convention), and inspection of interface/ receptacle signatures (cf. standard Java reflection in which interfaces can be discovered and inspected at run-time), and
- a *resources* meta-model that represent types and quantities of resource dedicated to various components or sets of components].

Detailed discussions of the first three of these meta models can be found in [2]. Detail on the resources meta-model is available in [3].

3.3 Component Frameworks: Constraining Reconfiguration and Providing Structure

Although *necessary*, the component model's explicit representation of dependencies and its reflective meta-models are not in themselves *sufficient* for the *management* of reconfiguration. In particular, their genericity precludes *specific* competencies in imposing and policing domain-imposed constraints on reconfiguration. For example, they cannot prevent the nonsensical replacement of an H.263 encoder with an MPEG encoder, or mandate that a packet scheduler must always receive its input from a packet classifier. Such constraints are essential if we are to ensure meaningful configuration and reconfiguration, and therefore the system must provide support for their expression and enforcement.

To add the necessary dimension of specificity and constraint, and also to provide *structure* for domain-specific component configurations, we apply the notion of *component frameworks*. These were originally defined by Szyperski [39] as "*collections of rules and interfaces that govern the interaction of a set of components 'plugged into' them*" (see figure 4). More concretely, component frameworks (hereafter, CFs) are targeted at a specific domain and embody 'rules and interfaces' that make sense in that domain. For example, we might employ a *protocol CF* that embodies knowledge, in the form of appropriate rules and interfaces, about the configuration (and reconfiguration) of the 'plugged-in' protocols that it hosts (e.g. "you may not place an IP component on top of a TCP component"). Similarly, a packet-forwarding CF might accept packet-scheduler plug-ins; or a media-stream filtering CF might accept various media codecs as plug-ins.

Essentially, CFs serve as 'life-support environments' for components in a particular domain or application area. They contain arbitrary CF-specific state, embody shared services for plug-ins, and actively police their plug-ins to ensure that they conform to their domain-specific rules and interfaces (e.g. interfaces can be inspected at run-time using reflection). CFs can support multiple instances of multiple types of plug-in, and plug-ins can either be independent of each other or can be bound together in arbitrary configurations (as long as these conform to the rules imposed by the host CF).

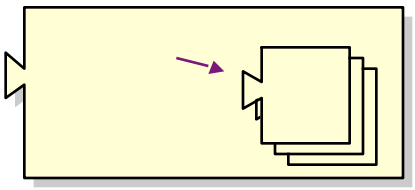

Figure 4: The concept of component frameworks.

CFs themselves are packaged as components. One implication of this is that, like any other component, CFs can be loaded/ unloaded dynamically. Another implication is that we can *nest* CFs to gain the benefits of hierarchical composition. For example, we have previously built a whole middleware infrastructure as a nested set of CFs [8].

To support reconfigurability that is consistent with domain-specific constraints, CFs can also provide CF-specific reflective meta-models that embody domain specific semantics. These are typically layered on top of one or more of the generic meta-models mentioned at the end of §3.2. For example, a protocol CF could constrain an architecture meta-model to accept only linear topologies. In addition, CFs often require their plug-ins to support pre- and post-reconfiguration operations so that the host CF can ensure that they are in a dormant state before being reconfigured and can secure their state over reconfiguration operations.

3.4 Potential Benefits

The most obvious potential benefit of the proposed approach is that its ubiquitously-applied component model promises a uniform environment for the development, configuration, and reconfiguration of programmable networking software at all levels of the system and at any appropriate granularity and using any appropriate programming language. For example, functions as diverse as in-band packet handling and signaling can be developed, deployed, configured and reconfigured in a common manner and can rely on common support (such as dynamic remote instantiation, reflective services, and generic mechanism level security and safety support). In addition, the approach is, in principle, sufficiently general to accommodate any of the currently popular programmable networking paradigms (active networking, open signaling or application-

level active networking). Essentially, all of these (e.g. an active networking EE) can be implemented in terms of CFs. Also, because components are language independent, portable, and (hopefully) can be applied at a wide range of granularities, they offer a solid basis for the incremental deployment of *existing* programmable networking software into a common component-based environment.

At a more detailed level, the fact that they are explicitly aware of their dependencies means that components can be (automatically) loaded on demand by their host CF so that only functionality that is actually needed at any given time need be resident on each node. Thus, a JVM instance (wrapped as a component) need only be loaded when the first Java component is deployed in a given address space; or a stratum 1 threading component need only be loaded if some component requires threads. This conserves resources and enables routers with limited capabilities to participate more effectively in programmable networking environments.

In general, the approach facilitates bespoke software configurations—by selecting appropriate CFs in each stratum, desired functionality can be achieved while minimising memory footprint; trade-offs will vary for different system types (e.g. embedded, wireless devices; large-scale core routers).

The approach also facilitates analysing and operating on per-node software as a single composite—e.g. we can use the architecture meta-model to check consistency, integrity, security, etc; and can uniformly reconfigure and evolve the node's software base as needed (e.g. to load new functionality on demand, or unload functionality when no longer required; or juggle node resources between different activities); we can also instrument any part of the system in a uniform manner (using interceptors).

Furthermore, the approach helps organise ad-hoc interaction between layers—as all software is structured in terms of a uniform component model, any part of the system has the basic capability to talk to any other part (barring access control, and security etc. concerns) in a principled way (cf. [48])—e.g. application or transport layer components can straightforwardly obtain 'layer-violating' information from, e.g., the link layer (this is increasingly recognised as indispensable in mobile environments); furthermore, such links can be established in an ad-hoc, dynamic, manner.

Finally, reflection and CFs together promise significant benefits in terms of the management of configuration and reconfiguration. Generic meta-models can provide multiple views of component configurations and support 'principled' runtime inspection and reconfiguration along multiple alternative 'dimensions'. And where it is important to temper this power to honour domain-specific constraints, CF-specific meta-models can be used to appropriately constrain reconfiguration operations. Additionally, CFs simplify component development and assembly through design reuse and guidance to developers, encourage lightweight components (plug-ins), and increase the understandability and maintainability of systems. Most crucially, because CFs embody semantics and impose constraints relating to their area, they can play a leading role in maintaining *integrity* in the face of reconfiguration.

4. IMPLEMENTATION

4.1 Overview

Our implementation and evaluation of the NETKIT approach to programmable networking is still at an early stage. In this section, we describe our implementation work to date. §4.2 discusses work on deploying the component model, while §4.3 discusses a prototype stratum 2/3 CF.

To evaluate its claimed support for heterogeneity, we are currently working to deploy the NETKIT approach not only in standard PC-router environments, but also in Intel IXP1200 network processors-based routers [23], and in embedded, wireless and mobile devices [42]. This heterogeneity is crucial in validating the claimed generality of our approach. In all cases, the challenge is to maintain as much commonality as possible without compromising either (re)configurability or performance.

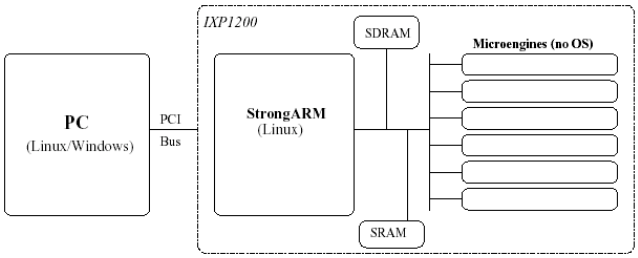

Figure 5: Schematic architecture of an IXP1200-based router.

In this paper, we focus mainly on the Intel IXP1200 implementation environment. As sketched in figure 5, the IXP1200 features an exotic hardware architecture comprising multiple processors—both a StrongARM control processor and primitive Intel-proprietary 'microengine' processors—together with various distributed/ hierarchical memory arrays.

4.2 Component Model Implementation

Our component model implementation, called *Maya*, is currently built on top of a subset of the Mozilla's XPCOM component model [45]. However, we are progressively moving away from the XPCOM dependency by applying the portability principles outlined in §3.1.2. For example, we are wrapping the stratum 1 level support provided by XPCOM into independent CFs. More importantly, we are structuring the component model run-time itself in terms of a number of CFs as follows:

- a multi-address-space capsule CF,
- a plug-in loader CF,
- a plug-in binder CF.

The multi-address-space capsule CF takes address spaces as plug-ins, resulting in a per-capsule run-time environment that comprehends multiple address spaces. For example, a capsule could encapsulate both a Linux process on the IXP1200's control processor, and one or more microengines (each microengine is associated with a single address space). Encapsulating multiple address spaces in capsules offers a powerful and general means of abstracting over tightly-coupled but heterogeneous hardware: the components within the capsule do not need to know that their execution environment differs from that of their peers, and they can

uniformly operate on their peer components, and be operated on, using a common set of meta-models.

Building on multi-address-space capsules, the plug-in loader and plug-in binder CFs support (as plug-ins) multiple alternative implementations of component loading and binding respectively. In particular, these plug-ins can provide third-party loading/ binding in (intra-capsule) address spaces other than the one from which they were invoked. This builds on the transparency offered by the multi-address-space capsule concept and makes such a capsule a truly unified component support environment. For example, a component running in a Linux address space can initiate the loading of a component onto a microengine—without necessarily knowing that the component will be placed on a microengine—and then bind itself to the newly-loaded component without being aware that the latter is in any way different from itself. Happily, this transparency entails no change to the component model: it simply leverages its existing third-party loading/ binding concept.

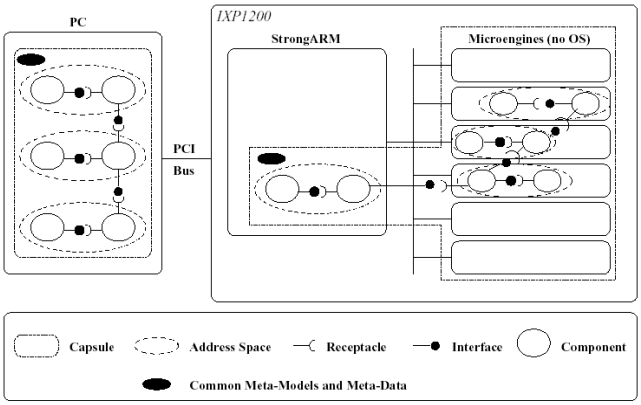

Figure 6: Multi-address-space capsules, loaders and binders

Figure 6 illustrates the multi-address-space capsule and plug-in loader/ binder concepts in the IXP1200 environment: it shows a multi-address-space capsule that encapsulates a Linux process address space and six microengine address spaces. Within this capsule are a number of components that are loaded and bound using in-capsule plug-in loaders and binders. The figure also shows a capsule in the PC environment that encapsulates three Windows address spaces, each of which contains a number of communicating components (this latter will be revisited in §4.3).

Transparency of loader/ binder selection is achieved by providing a standard set of polymorphic capsule APIs (i.e., *load()*, *unload()*, *bind()* and *unbind()*). On each call of these APIs, an appropriate plug-in is chosen on the basis of runtime configuration information. The choice of a loader, for example, might be based on attributes attached to the to-be-loaded component, such as target processor-type, target OS-type etc. Similarly, a binder might be selected on the basis of the hosting address spaces of the to-be-bound interface and receptacle. For example, to bind two components on separate microengines, a binding implementation based on shared scratch memory might be (transparently) selected. Where more control is required, and where multiple possibilities exist (e.g., where there is a choice of multiple microengines on which to load a component), transparency of plug-in selection can be foregone by means of a CF-specific meta-interface.

As well as providing a simple and consistent programming model, implementing loading and binding as plug-ins considerably simplifies the task of porting the Maya runtime to exotic architectures such as network processors. Returning to the above StrongARM/ microengines example, we simply employ a standard, generic, Linux capsule implementation. It is only the architecture-specific plug-in functionality (loaders and binders) that need to be microengine-aware. We expect the following to be a common deployment pattern: a 'primary' address space hosts the Maya runtime and 'secondary' address spaces present limited functionality to their hosted components. For example, a component hosted in a ('secondary') microengine address space will typically not have access to loaders and binders (i.e. the functionality underlying *load()*, *bind()* etc. will, for such components, be null). The approach also means, of course, that the dedicated fast-path packet-processing parts of the architecture are free of the performance and memory burden of the runtime. We emphasise again, though, that all this (i.e. notions of 'primary' and secondary' address spaces etc.) is entirely transparent to the Maya programmer.

As well as the default intra-capsule vtable-based bindings (we inherited these from Maya's XPCOM implementation base), we are currently developing a range of IXP1200-specific plug-in binding types. These are based on *i*) register transfers; *ii*) modifying branch instructions (cf. NetBind [7]); *iii*) shared memory mediated links involving either scratch memory or the additional static or dynamic RAM provided by the IXP1200; *iv*) paths over the various buses provided by the IXP1200.

We have not yet carried out a comprehensive performance evaluation of the IXP1200-specific loaders and binders. We observe, however, that the overhead of establishing and reconfiguring bindings is entirely 'out-of-band' and does not impact data flowing between components. The major factor impacting the overhead of in-band inter-component communication is the choice of binding mechanism involved. As we are using essentially the same mechanisms as other well-evaluated systems (i.e. Netbind [7] and Intel's MicroACE [23]) there is no reason to expect that performance should suffer. The one Maya-specific feature that might significantly impact performance is the *number* of inter-component bindings involved—which is a function of the granularity of components. Again, based on evaluations of previous fine-grained systems such as Click [28] we have no a-priori reason to believe that fine-grained componentisation is necessarily problematic.

4.3 Component Framework Developments

Our initial focus in the CF area has been on the design of a simple, but non-trivial, programmable networking-oriented CF that exercises many of Maya's configuration and dynamic reconfiguration features (including: multi-address-space capsules, plug-in loaders and binders, dynamic insertion of components based on the architecture meta-model; run-time type checking and interface discovery; the resources CF; and interceptors). Specifically, we have designed a stratum 2 and 3 'Router CF' which accepts, as plug-ins, Maya components that perform arbitrary user-defined packet-forwarding functions. Figure 7 illustrates one possible instantiation of the CF; however, the CF is capable of instantiating a very wide and general range of router configurations as long as these conform to a minimal set of CF-imposed rules.

In particular, the following set of rules, enforced at component load time by the CF using Maya's architecture and interface meta-models, must be adhered to by plugged-in components:

- plugged-in components must support specific packet-passing interfaces/ receptacles (called *IPacketPush* and *IPacketPull*: these respectively enable push- and pull-oriented inter-component communication [28]);
- plugged-in components may (optionally) support an *IClassifier* interface which exports an operation *register_filter()* that is used to install packet-filters; the intended semantic is that an installed packet-filter directs outgoing packets to particular outgoing *IPacketPush* or *IPacketPull* interface(s) that are named in the packet-filter specification; installing packet-filters may entail creating additional instances of these interfaces, which is possible using the standard Maya programming model;
- plugged-in components may be composite, in which case all their internal constituents must (recursively) conform to the CF's rules; additionally, composite components are expected to contain a so-called controller component that manages and configures the other internal components (see figure 7).

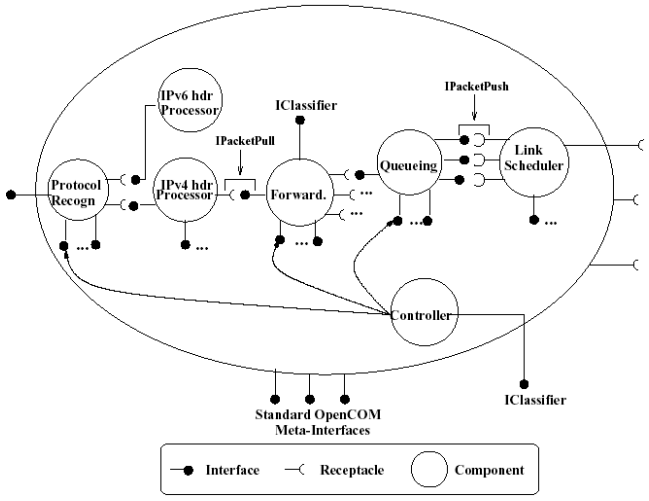

Figure 7: A composite that conforms to the Router CF

The CF also supports the definition of 'structural rules', expressed in terms of a simple XML schema, that constrain the reconfiguration of, and thus the internal topology of, composite components. Furthermore, these rules can be added or removed dynamically. Addition/ removal of rules is policed by an ACL managed by the composite's controller; the rules themselves are interpreted and enforced within an interceptor that is attached to calls of Maya's *bind()* primitive.

The Router CF also addresses safety/ security issues. To prevent untrusted plugged-in components (e.g. per-application components that act on a particular preselected packet flow) from maliciously tampering with the code/ data of other components in the same capsule, or from accidentally taking down the whole of the router capsule by crashing, we exploit Maya's support for multi-address-space capsules (see figure 6). In particular, a specialised plug-in loader is used which, if it determines that a to-be-loaded component is potentially malicious or otherwise dangerous, instantiates a new 'secondary' address space and loads the component into that (alternatively, if such an address space is already in place from a prior load, then this may be used) [43]. Such 'secondary' address spaces are barred from themselves accessing loading and binding services so that components loaded into them cannot initiate any such activities. When these untrusted components need to be bound to others in the 'primary' address space, a companion plug-in binder (having validated the legality of the binding) transparently deploys the appropriate inter-process communication mechanisms as discussed above.

Finally, the Router CF heavily exploits Maya's resources meta-model so that composites (subject to access constraints) can control the resourcing of designated tasks (e.g. packet forwarding, route lookup), especially in terms of threads, and map these flexibly to their constituent components.

The design of the Router CF is now fairly mature and we are implementing it in both PC-based and IXP1200-based routers. We hope to be able to validate its performance and flexibility in the near future. Interestingly, the IXP1200 implementation will bring to the fore the issue of component 'placement': in the PC implementation, we already, as described above, choose to place components in different address spaces according to security/ safety considerations; in the IXP environment we additionally need to situate components (whether on the control processor or on some specific microengine) according to performance, memory availability, and load-balancing considerations. We consider that the CF itself should embody the 'intelligence' to transparently manage this placement, but with the possibility to control/ override this via a 'placement' CF built into a microengine loader.

5. FURTHER RELATED WORK

§2.2 has already discussed related work in the various programmable networking paradigms. That section also discussed stratum 2 component models for programmable networking. In this section we round off these discussions by briefly surveying related work in the area of software components in general and component based middleware in particular.

MMLite [20] is a component-based operating system built using MS COM components. It offers limited support for dynamic reconfiguration through a 'mutation' mechanism which enables the replacement of a component implementation at run-time. However it has no *framework* (e.g. in terms of reflection and CFs) to support and facilitate this replacement. Think [41] is another lightweight component model that is targeted at the construction of system software. It is close to Maya in its goals but has so far only been used in operating system implementation.

In the middleware environment, other researchers have investigated lightweight and flexible component architectures—like us, they aim to build the middleware itself in terms of components as opposed to merely supporting components on top of monolithic middleware. Prime examples are the University of Illinois' DynamicTAO [29] and LegORB [38]. These are flexible ORBs that employ a dependency management architecture that relies on a set of 'configurators' that maintain dependencies among components and provide hooks at which components can be attached or detached dynamically. Maya supports a similar capability but as an integrated part of the component model. Another example is work at Syddansk University on building real-time control middleware in terms of JavaBeans [26]. Again, none of this

work has yet been applied in the programmable networking environment.

Finally, the OMG's CORBA Component Model (CCM) [36] is aimed at facilitating the deployment of distributed applications in an enterprise environment. Its central aim is to reduce the time to market for server-side code by providing a configurable server-side *container architecture* that supports generic non-functional concerns like transactions, persistence and lifecycle management. Other related solutions are Microsoft's DCOM and .NET [33], and Sun's Enterprise Java Beans. Although these technologies hold significant promise in the enterprise environment, they are not directly applicable to programmable networking environments because their container architectures carry significant overhead in terms of performance and memory footprint. In addition, some of them (i.e. EJB and .NET) operate only in a bytecode execution environment.

6. CONCLUSIONS AND FUTURE WORK

We believe that a fine-grained, reflective, language-independent component model, as discussed here, offers significant potential as the basis of an 'integrated' approach to the structuring of programmable networking software.

Apart from the potential benefits outlined in §3.4, we see our work as having potentially great applicability in the specific area of programming support for *network processors*. It is widely acknowledged that these architectures are difficult to program and that there is little or no commonality in programming environments across these machines due to their extreme architectural heterogeneity [44]. We believe that our component-based approach is a promising way of providing at least a degree of design portability across these architectures. A components- and bindings-based model seems to fit many such architectures, and the approach discussed in §4 of implementing loading and binding functionality as architecture-specific plug-ins to a generic component model runtime seems to have potential in exploiting and unifying a wide diversity of processing environments and internal communication mechanisms (the latter by means of plug-in binders). Furthermore, it is easy to see how network processor-specific hardware assists can be presented to the programmer as components. For example, a hardware checksummer can be presented as just another component to plumb in; the fact that it is implemented in hardware just means that the component implementation is effectively null (additionally, a binding to the checksummer 'component' could transparently map to whatever hardware-specific mechanism is needed to invoke the physical checksummer).

Finally, in addition to the IXP1200-related future work mentioned in §5, we are currently working with Columbia University to re-engineering their Genesis system [6]. This is a distributed service layer that supports the creation of dynamic private virtual networks, each potentially with its own semantics (addressing, routing, QoS, etc.). Apart from the opportunity to investigate the componentisation of an existing programmable networking system with a view to enhancing its deployability and (re)configurability, this is also particularly interesting to us as an exemplar of a richly-functioned stratum 4 system to complement our existing work in the other three strata.

7. REFERENCES

[1] The ANTS Toolkit, http://www.cs.utah.edu/flux/janos/ants.html.

[2] Blair G.S., Coulson G., Robin P. and Papathomas, M., "An Architecture for Next Generation Middleware", Proc. IFIP International Conference on Distributed Systems Platforms and Open Distributed Processing (Middleware'98), Davies N.A.J., Raymond K. & Seitz J. (Eds.), The Lake District, UK, pp. 191-206, 15-18 September 1998.

[3] Blair, G.S., Costa, F., Coulson, G., Duran, H., Parlavantzas, N., Delpiano, F., Dumant, B., Horn, F., and Stefani, J.B., "The Design of a Resource-Aware Reflective Middleware Architecture", Proceedings of the 2nd International Conference on Meta-Level Architectures and Reflection (Reflection'99), St-Malo, France, Springer-Verlag, LNCS, Vol 1616, pp115-134, 1999.

[4] Brown, K., "Building a Lightweight COM Interception Framework Part 1: The Universal Delegator", Microsoft Systems Journal, January 1999.

[5] Butler, R., Engert, D., Foster, I., Kesselman, C., Tuecke, S., Volmer, J., Welch V., "A National-Scale Authentication Infrastructure", IEEE Computer, Vol 33, No 12, pp 60-66, 2000.

[6] Campbell, A.T., Kounavis, M.E., Villela, D.A., Vicente, J.B., de Meer, H.G., Miki, K., Kalaichelvan, K.S., "Spawning networks", IEEE Network Magazine, Vol 13, No 4, pp. 16-29, July/Aug 1999.

[7] Campbell, A.T., Chou, S., Kounavis, M.E., Stachtos, V.D., and Vicente, J.B., "NetBind: A Binding Tool for Constructing Data Paths in Network Processor-based Routers", 5th IEEE International Conference on Open Architectures and Network Programming (OPENARCH' 02), June 2002.

[8] Coulson, G., Blair, G.S., Clark, M., Parlavantzas, N., "The Design of a Highly Configurable and Reconfigurable Middleware Platform", ACM Distributed Computing Journal, Vol 15, No 2, pp 109-126, April 2002.

[9] Coulson, G., Moonian, O., "A Quality of Service Configurable Concurrency Framework for Object Based Middleware", Concurrency and Computation: Practice and Experience (to appear), 2002.

[10] Chandra, P., Fisher, A., Kosak, C., Ng, T.S.E, Steenkiste, P., Takahashi, E., Zhang, H., "Darwin: Customizable Resource Management for Value-added Network Services", in 6th IEEE Intl. Conf. on Network Protocols (ICNP 98), Austin, Texas, USA, Oct 1998.

[11] Schmid, S.,Chart, T., Sifalakis, M, Scott, A.C., "Flexible, Dynamic and Scalabale Service Composition for Active Routers", Proc. IWAN 2002, Zurich, Dec. 2002.

[12] Clark, M., Blair, G.S., Coulson, G., Parlavantzas, N., "An Efficient Component Model for the Construction of Adaptive Middleware", Proc. IFIP/ACM Middleware 2001, Heidelberg, Nov 2001.

[13] Decasper, D., Dittia, Z., Parulkar, G., Plattner, B., "Router Plugins: A Software Architecture for Next Generation Routers", Proc. ACM SIGCOMM 98, 1998.

[14] Engler, D.R., Kaashoek, M.F., O'Toole, J., "Exokernel: An Operating System Architecture for Application-Level Resource Management". Proc. 15th ACM Symposium on Operating Systems Principles, Copper Mountain, CO, USA, pp 251-266, Dec 1995.

[15] Fry, M., Ghosh, A., "Application Level Active Networking", Proc. 4th Intl. Workshop on High Performance Protocol Architectures (HIPPARCH '98), June 98.

[16] Merugu, S., et al, "Bowman and CANEs: Implementation of an Active Network", Proc. 37th Conference on Communication, Control and Computing, Monticello, Illinois, September 1999.

[17] Yemini, Y., da Silva, S., "Towards Programmable Networks", Proc. IFIP/IEEE International Workshop on Distributed Systems: Operations and Management", Italy, October 1996.

[18] Hicks, M.W., Moore, J.T, Alexander, D.S., Gunter, C.A., Nettles, S., "PLANet: an Active Internetwork", Proc. IEEE INFOCOM (3), pp 1124-1133, 1999.

[19] Schwartz, B., et al, "Smart Packets for Active Networks", Proc. OPENARCH 1999, March 1999.

[20] Helander, J., Forin, A., "MMLite: A Highly Componentized System Architecture". Proc. 8th ACM SIGOPS European Workshop, pp 96-103, Sintra, Portugal, September 1998.

[21] Hjalmtysson, G. "The Pronto Platform - A Flexible Toolkit for Programming Networks Using a Commodity Operating System", Proc. IEEE Conf. on Open Architectures and Network Programming, OPENARCH 2000, Tel-Aviv, Israel, March 2000.

[22] IEEE P1520 Proposed IEEE Standard for APIs for Networks, http://www.ieee-pin.org/.

[23] Intel IXP1200; http://www.intel.com/IXA.

[24] Isaacs, R., Leslie, I., "Support for Resource-Assured and Dynamic Virtual Private Networks", JSAC Special Issue on Active and Programmable Networks, 2001.

[25] Jones, N.D., "An Introduction to Partial Evaluation", ACM Computing Surveys, 28(3), pp. 480-504, Sept 1996.

[26] Joergensen, B.N., Truyen, E., Matthijs, F., and Joosen, W., "Customization of Object Request Brokers by Application Specific Policies". IFIP International Conference on Distributed Systems Platforms and Open Distributed Processing (Middleware'2000). New York. April 3-7, 2000.

[27] Karlin, S., Peterson, L., "VERA: An Extensible Router Architecture", Proc. IEEE Conf. on Open Architectures and Network Programming, OPENARCH 2001, Anchorage, Alaska, pp 3-14, April 2001.

[28] Kohler, E., Morris, R., Chen, B., Jannotti, J., Kaashoek, M.F., "The Click Modular Router", Proc. ACM SOSP 1999, pp 217-231, Dec 1999.

[29] Kon, F., Román, M., Liu, P., Mao, J., Yamane, T., Magalhães, L.C., and Campbell, R.H., "Monitoring, Security, and Dynamic Configuration with the dynamicTAO Reflective ORB". IFIP International Conference on Distributed Systems Platforms and Open Distributed Processing (Middleware'2000). New York. April 3-7, 2000.

[30] Liedtke, J., "On µ-Kernel Construction", Proc. 15th ACM Symposium on Operating System Principles (Copper Mountain Resort, CO., Dec. 3-6). ACM Press, New York, NY, pp. 237-250,1995.

[31] Maes, P., "Concepts and Experiments in Computational Reflection", Proc. OOPSLA'87, Vol. 22 of ACM SIGPLAN Notices, pp147_155, ACM Press, 1987.

[32] Mozilla Organization, XPCOM project, 2001, http://www.mozilla.org/projects/xpcom.

[33] Microsoft, .Net Home Page, http://www.microsoft.com/net.

[34] NodeOS Interface Specification, AN Node OS Working Group, http://www.cs.princeton.edu/nsg/papers/nodeos.ps, Jan 2001.

[35] Peterson, L., Gottlieb, Y., Hilber, M., Tullmann, P., Lepreau, J., Schwab, S., Dandekar, H., Purtell, A., Hartman, J., "An OS Interface for Active Routers", IEEE Journal on Selected Areas in Communications, special issue on Active Networks, March 2001.

[36] Object Management Group, "CORBA Components" Final Submission, OMG Document orbos/99-02-05.

[37] Reid, A., Flatt, M., Stoller, L., Lepreau, J., Eide, E., "Knit: Component Composition for Systems Software", Proc. OSDI 2000, pp 347-360, Oct 2000.

[38] Roman, M., Mickunas, D., Kon, F., and Campbell, R.H., "LegORB", Proc. IFIP/ACM Middleware'2000 Workshop on Reflective Middleware, IBM Palisades Executive Conference Center, NY, April 2000.

[39] Szyperski, C., "Component Software: Beyond Object-Oriented Programming", Addison-Wesley, 1998.

[40] Villazón, A., "A Reflective Active Network Node", Proc. 2nd Intl. Working Conf. on Active Networks (IWAN 2000), Tokyo, Japan, Oct 2000.

[41] Fassino, J.-P., Stefani, J.-B., Lawall, J., Muller, G., "THINK: A Software Framework for Component-based Operating System Kernels", Proc. Usenix Annual Technical Conference, Monterey (USA), June 10th-15th, 2002.

[42] Grace, P., Blair, G.S., Samuel, S., "ReMMoC: A Reflective Middleware to Support Mobile Client Interoperability", Proc. International Symposium on Distributed Objects and Applications (DOA 2003), Catania, Sicily, Italy, November 2003.

[43] Schmid, S., "A Component-based Active Router Architecture", Lancaster University PhD Thesis, http://www.mobileipv6.net/~sschmid/PhD_Thesis.ps, 2002.

[44] Comer, D., Peterson, L., "Network Systems Design Using Network Processors", ISBN 0131417924, Prentice-Hall, 2003.

[45] Mozilla Organization, XPCOM project, 2001, http://www.mozilla.org/projects/xpcom.

[46] Bos, H., Samwel, B., "The OKE Corral: Code Organisation and Reconfiguration at Runtime using Active Linking", Proc. IWAN 2002, Zurich, Dec 2002.

[47] Solarski, M., Bossardt, M., Becker, T., "Component-based Deployment and Management of Services in Active Networks", Proc. IWAN 2002, Zurich, Dec 2002.

[48] Braden, R., Faber, T., Handley, M., "From Protocol Stack to Protocol Heap–Role-Based Architecture", ACM SIGCOMM Computer Communication Review, Vol 33 No 1, January 2003.

Workshop Report
Network Research: Exploration of Dimensions and Scope

August 25, 2003
Karlsruhe, Germany

Karen R. Sollins
Massachusetts Institute of Technology
sollins@csail.mit.edu

Introduction

The **Network Research: Exploration of Dimensions and Scope** was intended to be a next step beyond the National Academy Press report **Looking Over the Fence: A Neighbor's View of Network Research** (National Academy Press, 2001). Further, it was organized in a context in which a number of the funding agencies in the United States were also funding their own workshops and reports intended to explore the future of networking research, as seen from the perspective of each of those agencies. Among these agencies are the DoD Advanced Research Projects Agency, the Department of Energy, and the National Science Foundation. The intention of this workshop was to separate the discussion from a particular agency. It was also understood that this would be a one-day workshop, under the auspices of SIGCOMM on August 25, 2003, at Karlsruhe, Germany. This had two implications. First, clearly that was likely to have an influence on participation, although some effort was made to include participants who do not normally attend SIGCOMM and whose research fields are not typically central to SIGCOMM kinds of topics. Second, in one day we could not expect to make significant progress with consensus or conclusions, but rather encourage discussion, trying to get ideas out onto the table.

The workshop solicited 5 page position papers initially, for two reasons. The first was to raise ideas and topics that the organizing committee might not otherwise have recognized. The second was as a start on who might be invited to the workshop. Of the papers submitted, 6 were accepted and distributed to the participants of the workshops. Attendance included authors from these 6 as well as several others of the papers, and additional participants to broaden representation from the community.

In order to generate discussion, the committee identified five questions:

1. Do we have a shared meaning for "network research"?
2. Where is the science in network research?
3. Where is the research beyond the current tipping point?
4. How do we value and evaluate research? How does/should our field evolve?
5. Where do we go from here?

For each question except the last, we invited a speaker to raise some issues briefly (10-15 min) and a respondent who was given 5-10 min. At that point the session was opened up to general discussion, chaired by one or another of the committee members. Notes were taken by 3 student scribes. The final topic was only a discussion. The intetion was to expose questions and concerns; this report is a summary of those that arose during the day. It makes no claim to completeness or conclusions.

The workshop committee was Mark Allman (ICIR), Balaji Prabhakar (Stanford), Stefan Savage (University of California San Diego), and myself, Karen Sollins (MIT) as chair. The scribes were Steve Bauer (MIT), Mayank Sharma (Stanford), and Renata Teixeira (University of California San Diego). Although the initial speakers in each session are identified below, the participants are not because many of the points were part of the larger discussion and thus attribution is impossible. Without the invaluable contributions of the other committee members, scribes and participants the workshop and this report would not have been possible.

It is our intention to create a web page from the SIGCOMM site on this workshop.

Question 1: Do we have a shared meaning for "network research"?

The initial speaker Dave Cheriton and respondent Nick McKeown presented clearly different opinions on the role and value of network research, especially academic research. Cheriton made his case based on a vision of the history of networking consisting of a series of technology developments and the impact that the "research" community did not have on that history. From this he concluded that the work that is acknowledged as research by our community is evaluated more on innovation than industrial impact. One conclusion to draw from those statements is the position that network research should have reasonably direct impact on industry. Cheriton also is a strong proponent of the position that the over-riding challenging problem for network researchers is scaling. Cheriton would use this as a driving criterion for evaluating efforts in networking research.

In contrast, McKeown proposed that network research is more axiomatic in its basis. Hence he suggested that valuable network research proceeds by leaps with radical ideas that challenge previous ideas, but also progresses linearly in the absence of such non-linear transitions. This led McKeown also to consider the production of such radical ideas. One problem is that they are difficult to predict and therefore argues for supporting wide diversity, in order to learn through experimentation with new ideas. Experience also suggests that such ideas are more likely to come from younger researchers than older ones, as in many other fields of research. As an aside, and a comment to which we will return, it was noted that often younger researchers are the most critical of both themselves and others, often making it more difficult to include a diversity of ideas in peer reviewing situations (both in terms of funding review and publications review).

These two viewpoints opened the discussion to a broad cross section of opinions and viewpoints. For example, one aspect of Cheriton's position was a valuing of research utility on the basis of direct and long-term impact on industry. One of the issues that gets lost in such a metric is the more ephemeral but possibly quite significant impact on thinking that in turn may lead to yet other ideas. For example, a question arose over the value of Ethernet, in particular whether the value of it was in the idea of CSMA/CD or in the long-lived preserved interfaces. An example such as this highlights some of the breadth of differences of opinion in the room, and different ways of looking at the question of what networking research is.

Another question that arose was the present and future role of mathematics in network research. Although a position such as Cheriton's did not encompass increased rigor in network research, there was less disagreement about the value of increasing the role of mathematics. Opinions on this topic ranged along a spectrum represented by two extremes. One extreme holds that some form of mathematical expression of the phenomena we see possibly or probably requires new mathematical theory is the central problem, leading to a call for a theory comparable to information theory or thermodynamics for network complexity. A midpoint position was the opinion that we are currently making progress on developing mathematical models using currently existing mathematical techniques. At the other extreme was the position that there is significant value, often lost on students, of non-mathematical research. This last position was reflected in the comment that often the most challenging problems we face in networking are the ones we don't yet know how to express mathematically. This was accompanied by a concern that we often teach our students to undervalue this aspect of network research in favor of problems that can be expressed mathematically. Another position related to this was the concern that, to the extent we focus on modeling and explaining current phenomena mathematically we may be losing sight of the fact that the current approaches may reflect at best imperfect engineering solutions, rather than the intrinsic complexity of networking.

Closely related to the question of the relationship between mathematics and networking research was the question of to which fields we might compare networking research. One participant laid out three possibilities of the nature of the field:

1. A performance discipline, solving the problem of making "the network" increasingly faster or more efficient in some other way, followed by a clean-up

activity that involves optimization, often by means of mathematics.

2. An infrastructure field in support of applications that is embarrassed to give credit to the fact that the applications' arena is where the most challenging problems are currently arising.
3. The work that funding agencies will fund, with the clear implication of who might be driving the definition in this case.

An alternative to this was an extension of the discussion about mathematics above, in which one of the participants suggested that networking research is much more like economics than science. Science is about discovering underlying principles and rules, whereas in economics and perhaps networking research, as man-made phenomena, we can change the rules in our models and explore real, alternative possibilities. In the case of networking, this can be done by changing the actual mechanisms and the bases on which they operate, as well as doing this in our theoretical models.

One of the concluding comments in this discussion was that as with our field generally, we should allow for a variety of definitions of what we mean by network research and a rich mixture that integrates more theoretical aspects such as proofs of correctness, viability, or models with implementation and engineering. But more than that both in this abstract sense of the definition of network research, but also in our more specific thinking there should be a sense of cooperation rather than competition. One participant urged us to distinguish between styles or methodologies and actual topics. We returned to this question of topics later in the day.

Question 2: Where is the science in network research?

In this session Walter Willinger provided the initial talk, with Tony Ephremedes as the respondent. Willinger questioned the "science" in networking research in several ways. One significant concern is with the application of modeling and evaluation as it is currently practiced in the networking research community. Willinger does not believe there is "science" in such efforts as traffic modeling, topology modeling, performance evaluation, network simulation, protocol design, or network architecture. In particular, he pointed out that the majority of the curve fitting sorts of activities are not interesting because there is no possibility of failure; one can always fit a curve to a set of points, and without rigorous validation, such an activity is not interesting. The problem as Willinger sees it is that the application of the technique is more or less blind. He sees the same story in topology modeling, although the theory applied is graph theory in this case. With respect to the more design-oriented aspects of our field, such as protocol design, again, since we do not understand optimality, there is little scientific about protocol or more broadly architecture design. As part of this line of argument, Willinger addressed the question of the relationships among networking research, math/physics/statistics, and other related fields. It is his opinion that those fields have little to contribute to ours, but, if we can get it right, we can make contributions to theirs, at a minimum by means new interesting examples to challenge their tool sets.

Willinger then asked whether there is value in including "science" in networking research. He does not have a clear answer, but finds a contradiction in examples. The design of TCP was reasonably unscientific, but after the fact we can demonstrate that it is approximately optimal for what it was designed to do. This would suggest that in the business of protocol design perhaps "science" is not needed to do well. On the other hand, if one considers BGP, as a community we have no idea where it stands in relation to optimality. To this, one of the other participants suggested that the optimality of TCP derives from extensive study and design of TCP's responses to network dynamics, whereas nothing of that sort has been applied to BGP.

Willinger concluded with two significant points. First, as a research community we should not only be fitting models to measured data, but should provide or include an understanding of complex network systems. Without such an understanding, neither validation nor extension is readily possible. Furthermore, there is a need for a systematic and thorough model validation process. Without formalizing and systematizing this so that it can be trusted, as one of the tools of our field of research, the field lacks a form of rigor that ought to exist.

In responding, Ephrimedes took a different approach. As he explained, his background is in control theory. It is clear that coding, control and information theory have not been part of major contributions in networking research, but there are beginnings and significant potential there. There are now beginning to be significant contributions in network coding from Medard and others. Other areas include capacity regions and physics, such as improved understanding of network phenomena that includes a model of the physical layer. In contrast, he warned that we should be careful about including models from biocomputing and molecular biology. That may be too far afield to be usefully applicable. From his perspective, one of the key components of networking research and science is curiosity. Intellectual curiosity should be the significant driver.

These positions were representative of the nature of the discussion in this session, exploring three distinctive components of the field of networking research: measurement, formalization and validation, and the design process. Neither of the speakers discussed measurement to any significant extent, but the participants raised the issue. The issues of measurement fall into two categories, first, measurement and experimentation directed at validation or testing of particular hypotheses, and, second, time series or long-range measurement with archiving. For both styles of measurement, but especially for the second, there was a call for more effective instrumentation of the network. There was recognition that CAIDA is attempting to provide long-range archiving, but that it cannot be expected to do it alone. There was clear recognition within the group that measurements are often either impossible for legal and commercial policy reasons, or "cleansed" in order to provide privacy in such a way that relationships among the data are lost, thus lowering the long-term value of the data. As will be discussed further below, there was a call for data to be made available much more broadly, in order to allow for repetition and revalidation of published scientific conclusions.

The presentations by Willinger and Ephrimedes focused to a large extent on issues surrounding the application of formalisms to networking as part of making it more scientific. Although Willinger suggested that "curve-fitting" does not make networking research scientific, the opinion was expressed that "curve-fitting" can and should be part of the process. There was deep concern that much of the modeling and simulation work that comes from academic researchers is written off by industry because it is poorly grounded. There is little work done to validate models and little or no work has been done to address the problem that we do not understand the effects of scaling, moving from a small simulation to simulations or conclusions about much larger scale situations. There were suggestions that there is a discontinuity or at least lack of understanding in moving from small scale to large scale. Willinger and others agreed that there is a need for rigorous "model verification process" as well as following through and including as part of a model an understanding of why the model is appropriate. It is clear that there are formalisms that can valuably be applied to particular aspects of networking, many of which are only in early stages. On this latter point, the community needs to make an effort to explain not only how a formalism can fit the data, but also how it is part of a better understanding of the phenomena being observed in the measurements.

As highlighted by Willinger, it is not clear that there is "science" in the design of networks and the particular protocols that comprise a network system. It is also not clear that there should be. One of the participants suggested that there might be "pockets" of science in networking research, but not overall. For example several people expressed the opinion that science is often driven by engineering questions. Science may provide bounds on engineering problems or possible solutions where existing models and understanding are inadequate. Another participant called for basic theory, the core of networking research. As evidenced by the breadth of different opinions, it is clear that there was no unanimity among the group about what the core of networking research is or should be. One interesting characteristic discussed is the fact that in networking, a researcher can imagine something and simply program it, while in physical sciences the researcher is limited to phenomena in the real world. (Hence, the physicists are led to arguments over whether string theory is physics or philosophy as long as it remains unobservable.) Although there was some agreement that our work needs to be based on some intrinsic principles or invariants, we are left with questions of identifying a small number of elemental ones.

Question 3: Where is the research beyond the tipping point?

The initial speaker in this section was John Wroclawski, with Tolga Uzuner responding. Wroclawski explored what he has identified as the tipping point in networking research. Network research began in a period of comfortable funding that allowed curiosity to be the driver. Over time, workable technologies were developed that provided useful functionality. With further support networking, devolved into an often critical role in achieving other goals. Networking took on a social and economic role, that has led, as with tipping points in other fields, to the point at which the economic investment in not changing outweighs the economic incentive to change or provide new services. This brings us to the point at which success has bred a resistance to change, which in turn means that newer technologies will not be accepted, despite the improvements they may bring. This leads to questions such as whether one can design to understand or select the tipping point and whether we should be teaching about the evolution of the process, to explain this tipping point to students. More specifically, we can consider possibilities for responding to the idea of a tipping point in several ways. First, we can try to explain and quantify the effects. Second, we could incorporate the concept into our design principles, by recognizing that there will be pull in several contradictory directions, and design specifically to enable and isolate some of these tussles. Third, we can intentionally design the playing fields for these tussle spaces, so that they will or will not tip at certain points.

Uzuner responded with the position that research is driven by economics. Innovation may occur in either the process itself or specific product innovation. He believes we are at or past the tipping point with networking technology, so that further interesting research and development will be somewhat limited and bounded by economics. The areas in which research can continue to have impact are theory (understanding and perhaps bounding complexity, noting that minimizing complexity may not be optimal), product strategies and yield management, and finance. In considering the different sorts of commercial players in the field, smaller companies will often benefit the most from research, to which one of the other participants responded that that is often by necessity. Uzuner also proposed that late comers to a technology are often the least likely to succeed; Uzuner was not contradicting Wroclawski, but rather agreeing and suggesting the directions in which research may still have an impact.

One of the participants brought up the term "network externalities". One of these is economics. One participant suggested that innovation is driven by need rather than economics. Once the solution is "good enough" then innovation stops. Another suggested that economics drives innovation in order to allow for "lock-in", although this participant also suggested that one needs to include the network architect in this analysis and for this person economics may not be the driver.

Wroclawski pointed out that he was calling for something more significant than network researchers becoming economists, in reaction to the view that economists are generally analysts, modeling existing phenomena. Rather, one of the roles the network researcher can play is as the shaper or molder of evolution, and as such we should do research on ecosystems in order to understand the interconnections. The network is something that is architected, designed and built. Understanding at many levels of abstraction how the network researcher can influence these is important. Wroclawski used an S-shaped curve; a number of the participants found interesting points with respect to these curves. One of the researchers pointed out that the transition points in such a curve are important, especially, the point at which innovation stops and product development becomes dominant. Another way of saying this is that at different points on such a curve different kinds of research may be done. There is some research that explains, other research that expands the possibilities and their benefits and costs. Then there is research that goes the next step beyond where we know how to go at present. For example, there was a point in time when it was understood that routers needed to be speeded up by orders of magnitude. The research on this topic was focused on engineering that speedup. Another participant pointed out that such curves can be seen in many other disciplines as well. It was also suggested that families of such curves allow for an exploration of the "evolution of evolvability". In

response, one of the other participants suggested that often it will be companies that stay on the existing path, while researchers are more likely to lead the way to a paradigm shift. These paradigm shifts are what move one from one curve to another. Although there was not unanimity on this subject there was further discussion about whether or not "research' should be limited to the curiosity, or early stages of such an S-curve, considering the rest to be something else. There was also a strong point made that there should be funding support for radical ideas.

Question 4: How do we value and evaluate research? How does/should our field evolve?

The initial speaker in this session was Craig Partridge, followed by Steve Wolff responding. Partridge considered the influences on research, especially environmental. One issue is the physical environment, which affects both the sorts of people involved and the roles they play. He considered a set of somewhat different kinds of facilities including: universities, not-for-profit research labs, for-profit labs, government labs (although he had little to say on this topic because it is outside his experience), and subsidized labs. The average cost per person in the academic environment (faculty and students) is about one third that of not-for-profit labs, in part because the faculty member is typically raising only two to three months of salary and graduate students are much less expensive. One of the clear distinctions is that in the not-for-profit the typical researcher is working full time. On the other hand the faculty member is much more of a small entrepreneur, raising money, producing output, mostly through students, and leveraging that to raise more. The difference in cost for researchers in the other sorts of labs is less different from the not-for-profit, although typically, the researcher in a not-for-profit and often in government labs is also raising money as the academic is, although in these cases for full salary.

One of the other clear distinctions in environment that is reflected in the nature of the research is the presence or absence of students and other kinds of staff. Typically, in a research lab, there are senior researchers with many years experience and a large number of recent PhDs doing the bulk of the research. There are few people in between. In labs, which are increasingly commercial, there are increasing numbers of support and administrative staff. The faculty member does the bulk of the management of funding and projects alone. In a laboratory there is likely to be fiscal, technical, and administrative staff as well as other support for the research operation.

Another difference arises from the sources of funding. There tend to be larger amounts of money for research that is expected to have more direct product results. In addition, there are increasing amounts of money for increasingly classified work. That said, corporate research labs are in deep trouble to the extent they still exist at all. There are three problems. The first is that they often were not doing things useful to the company. Second, often the company does not understand how to take advantage of possibly useful results. Third, these labs often do not know how to stop projects when their usefulness is past.

Wolff addressed questions of how we evaluate research. One distinction is between basic and applied research, which distinguishes based on whether "we" care about ownership of the intellectual property involved; if "we" care, then we can categorize the work as applied. Wolff also noted three distinct scales that may form an evaluation: prestige, funding, and academic peer.

The discussion fell into several major topics: the effects of funding raising requirements, peer and other reviewing, motivations for research, in what ways are we training our graduate students.

Beyond the questions raised by the speakers, several of the participants discussed the influence of needing to raise money in the academic and not-for-profit lab environment. The problem, especially for non-faculty is the need to raise funds continuously, to cover salaries. A faculty member can simply take a break now and then, and teach. In addition, the faculty member gets a sabbatical on a regular basis. The researcher gets no such break, but in exchange can work on projects full time. One of the effects on the researcher is that the need for continuous funding leads to incremental proposals, in order to increase the probability of

success in receiving funding. The only suggested path out of this dilemma is for the researcher to run several projects simultaneously so that they are at different points of advancement, allowing for some degree of exploration at any given time.

A second concern raised by a number of participants was that decision-making, especially with respect to paper selection for the most prestigious conferences (e.g. SIGCOMM), has had a stifling effect at least on research reporting, and possibly on research output more broadly. Two dimensions of this were discussed, the limitation to certain kinds of topics and the limitation to certain styles of papers. One of the younger members of the community expressed a degree of self-denial with respect to research reported in order not to violate "sacred cows". Another commented there is a certain amount of pressure to publish, which tends to drive at least some of the choice of subject matter for research and size of efforts into publishable units. It is important to notice that comments earlier in the day pointed out that it is often the younger members of the community who are the most critical of others and least tolerant of breadth of ideas and risk. There were some questions about whether this has any relationship to the fact that young faculty cannot afford to take risks themselves in their research, at least until they achieve tenure. This was followed by a discussion of the nature of the political structure that makes value judgments about research, questioning whether or not a democracy can be more effective. Those involved in NSF reviewing pointed out that more reviews do not generally reflect more distinct opinions. With reasonably broad reviewing representation, beyond three or four opinions on a proposal or paper, additional comments generally do not increase the number of distinct opinions.

At several points in the discussion participants brought up questions of what does and what ought to motivate researchers. Clearly, as Partridge and Wolff pointed out, for some researchers the motivators fall into such categories as promotion within one's organization, peer acceptance (often through publication), success in acquiring funding. Several participants suggested that their research was motivated at least in part by education. Another suggested that the intellectual exercise of the research itself was the motivator. One participant pointed out that even within the academic community this is dependent on the nature of the university. Researchers at top tier universities are more likely to have significantly more freedom in directing their own research. Those at lower tier schools find many limitations including less funding, heavier teaching loads, students who require less challenging projects and so forth.

Questions about students appeared in a number of the topics above, but one concern was discussed more fully, the question of what students are being taught broadly about the quality of research to which they should aspire. There was a sense that the bulk of the research done by students, at least in the USA, particularly because the larger numbers are not in top tier schools, is weak at best. The group was not clear about cause and effect with respect to this problem, but there was deep concern that by not setting the research standards high enough, students are not taught to set the standards high enough for themselves, a lesson that they will need later in life if they are to become researchers themselves.

Question 5: Where do we go from here?

In this last brief session, there was less discussion and more of simply throwing out ideas. They are reported here with no value judgment or particular ordering.

- As a community we should identify key foundational questions (as the mathematics community does). These may require "tools" (e.g. mathematics and other theory) that do not now exist. One response to this was that at least some of the basic understanding may not be expressible mathematically.
- We should change the model of evaluation, especially for program committees such as SIGCOMM, to make them either more democratic or more populist. One suggestion was to make reviewing not anonymous, but rather signed, allowing for better evaluation of the reviewer. Another was to post submissions publicly, allowing anyone who wanted to comment on them. There was some

discussion about finding a conference on which to experiment with a very different model of evaluation. Note that earlier discussions explored questions of who might be either conservative or overly critical of others' work.

- There needs to be significantly more participation in the process of evaluating our field. Two additional and related issues were raised. First, small groups are significantly more effective for discussion. Second, more than one day is important, in order to do more than lay out problems as was done to some extent in this workshop. One suggestion was to run several parallel small workshops of a couple of days. One might do some coalescing of results and conclusions.
- An alternative, less radical, suggestion was to encourage much more breadth and churn in program and other reviewing committees. Committees should include people from both traditional and newer (perhaps more radical) research directions. They should regularly include many more junior faculty. Perhaps there should be instituted maximum terms or number of terms within a longer period of participation on an individual committee.
- One participant was quite worried about the suggestion that some directions of research are "good" or worthwhile and others "bad" or less worthwhile.
- As a community we should make a much more significant commitment to cross-disciplinary, high risk, and disruptive ideas.

There seemed to be agreement that the discussions were only preliminary and need to be broadened to include more people and a broader set of people.

Workshop on Network-I/O Convergence: Experience, Lessons, Implications (NICELI)

Workshop Summary

Vinay Aggarwal
Olaf Maennel
TU München
Institut für Informatik
Boltzmannstr. 3
85748 Garching b. München
vinay@net.in.tum.de
info@olafm.de

Jeffrey Mogul
HP Labs
1501 Page Mill Road
Palo Alto, CA 94304
JeffMogul@acm.org

Allyn Romanow
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134
allyn@cisco.com

ABSTRACT

This is a summary of the NICELI workshop, based on scribe reports written by Olaf Maennel and Vinay Aggarwal, and edited by Jeffrey Mogul and Allyn Romanow with help from the NICELI attendees. The workshop was held in conjunction with SIGCOMM 2003 on 27 August 2003 in Karlsruhe, Germany.

Papers and presentations from the workshop are available on the Web at `http://www.acm.org/sigs/sigcomm/sigcomm2003/workshop/niceli/`

Note-taking during an active discussion is a fallible process, so these notes may contain errors. We have encouraged participants to help us find these errors. Corrections made after this article goes to press may be found at the NICELI Web site.

1. INTRODUCTIONS

Allyn Romanow welcomed the participants to this workshop about high speed I/O and thanked the speakers for their contributions.

2. KEYNOTE: DAVID R. CHERITON, STANFORD UNIVERSITY

Network-I/O Convergence in "Too Fast" Networks: Threats and Countermeasures

The workshop began with an 1-hour invited talk by David Cheriton from Stanford University. He observed that network I/O convergence is an old story. Giving a brief introduction to I/O development and its history right from the 50/60s to the present age, he focussed on the current problems facing the field – reordering, forged packets, replay, attack by peripherals. He expressed that the performance of the host should not be degraded by attacks.

The new problem is "too fast" networks, eg. 10 Gbps. He explained the term "too fast" to mean that "it is very expensive not to protect, and not feasible to do in software."

He then briefly commented on zero copy – corruption and compromise, and on receiver authentication – how to do efficiently and safely.

Then, he referred to Moore's law, inferring that the too-fast networks were at the limit of their memory speeds.

He observed that there is a collision between I/O and processor for the hardware resources. There is a contention for pins, on-chip state, and on-chip logic. If the contention is not in the process, it is way across in the I/O network. He humorously stated that everybody is trying to push the other off-chip!

He went on to highlight the threat perception of Infiniband. After explaining what is Infiniband, he stated – "fix IP for storage or else lose to Infiniband."

On the issue of multi-layer solutions, he observed that the more the layers, the more complex it will be for hardware. In the case of meta-protocols, he believed that the standards are too flexible to design hardware to a good standard. Hence, hard choices are required.

He later shifted attention to RPC, saying that since network is as fast as memory, not just RDMA, but RPC needs to be handled too.

He then proposed the solution of refactoring the transport layer protocol, based on the theory of refactoring the protocol design problem between hardware and non-hardware level.

Then he arrived on his final solution – an RDMA based protocol. He explained the term "region" as a collection of packet frames to/from which a sequence of packets of particular flow are mapped. Showing a diagram of a region structure, he talked about its delivery conditions and its pros. Tackling the issue of control, he explained the ROP control level. Then, he explained the working of File Write. For connection setup, he said that the channel manager is a mechanism to create or setup new channels.

To conclude, he reiterated the main points of his talk, and stopped with an open question – the counter-measure exists, but can the IP-Ethernet community respond?

Questions & Answers:

The first argument came from Steph Bailey: one community says TCP is good, another says TCP offload is the hardest part, not RDMA. But he believed that stable recovery and window management are the most challenging parts of TCP. What about congestion avoidance? To this, Cheriton replied that he did not address this issue in his talk. He said, perhaps, it is a good approximation.

Then, Jim Pinkerton from Microsoft said that TCP has a buffer to control that rate. The definition of the problem is interesting. He disagreed with Cheriton's statements about Infiniband. In Infiniband, the layers work differently. While the transport layer is very similar, the reliable layer is quite different. The problem is about congestion control and new algorithms.

Cheriton responded, all that has been learned about/from TCP can be applied again. All the mechanisms and problems about TCP can be solved by designing a new protocol. However, one needs to keep an eye on cost. The implementation of stacks on hardware needs to comply with cost constraints: it needs to fit on the chip.

Then Renato Recio made 2 comments: the discussion is specific to 1 protocol, the problems are not at the layer that the speaker is talking about, marshalling-demarshalling happens in Java, C#, etc. The answer was that RPC parameters must tie into RDMA for success. The dynamics here look similar to TCP. Due to the amount of memory wasted, the same ideas would have been comical 20 years ago, but not anymore, as memory is cheap now.

Jeff Mogul, HP Labs made the next point. He said that the frames are based on MTU size. If MTU changes, configuration changes will be required which the attacker can exploit. Cheriton agreed with this.

Next, there was an extended interaction between Jeff Chase of Duke University, Jeff Mogul, and David Cheriton. Jeff Chase began by observing that there is a collision between research and industry view. The research community believes everything can be built from the ground-up, but industry does not like this. It wants a standardized IETF blessed protocol. They would prefer to build on top of an older protocol. To this, Cheriton interjected that betting on TCP is bordering on the absurd. Then Jeff Mogul chipped in saying that problem lies in integration with the OS, not with TCP. Cheriton said that demands of what we build are colliding with the hardware reality. He then discussed TCP and SCTP. He remarked that the tipping point has arrived, and hence, change is essential.

Brent Callaghan from Sun Microsystems referred to the simplified RPC protocol. When RPC runs over any transport protocol, it is not so good. However, running over RDMA is quite a refreshing change.

Dave followed this up by saying that much of the proposal is for file access. When RDMA token is passed to the server, and server turns it back. This is an alternative way of flow control, and does not commit so much memory. He insisted that this is only a comment.

Cheriton concluded the questions by announcing that one may add to something, but, it is more complicated and expensive.

3. SESSION 1: PROMISES AND REALITY

3.1 Renato Recio, IBM

Server I/O Networks – Past, Present and Future

The speaker is associated with Infiniband, RDMA and IBM eServers at IBM Research. Recio said that the agenda of his talk was server I/O – network types, requirements, I/O attachments, etc. After highlighting the purpose of server I/O networks, he talked briefly about server I/O network requirements – standardization, performance, high availability, low cost – to name a few. Then he added to this list – virtualization, security, and service differentiation. Then he went on to summarize the server I/O network history and the network evolution timeline.

The speaker then switched to PCI. The strategy – add evolutionary technology enhancements to the standard. He maintained that there are 2 contenders: PCI-x and PCI-express.

After comparing the various I/O attachments, he spoke at length about Infiniband (IB) model and its strategy. When he compared IB with PCI-express, Raj Yavatkar objected saying that the comparison was inappropriate, as both have different uses.

The next part of his talk was on server scale-up technology options, server IOA outlook, attachment and expansion. Mentioning the problems with sockets over TCP/IP, he outlined the basic mechanisms of network offload. After explaining in detail the IB network stack offload, he showed the same for iONICs.

He then extolled the network offload benefits from the middleware's view. He went to say, rather artistically, "TCP/IP/Ethernet are kings of LANs!" Then, he devoted some attention to LAN issues, before discussing cluster network contenders, proprietary and standard networks. In the end, he briefly gave an overview on HPC cluster network outlook.

To summarize, he observed that I/O server adapters will likely attach through PCI family. He then gave a brief outlook on what each kind of network will likely use and why.

Questions & Answers:

Joerg Micheel (Endace) expressed interest in latency requirements, and desired to know where these came from. The reply was that there is no paging and congestion to hold that off.

David Cheriton was surprised that latency was pushing down through switches. He was not convinced that locking, interaction, etc. should have that kind of overhead. He believed that the logic for latency and overhead was not good, as disks and other old-fashioned devices were being used for paging, which were not appropriate. Recio contended that each device in the path had latency, and that lowers the number of operations. When questioned why, Recio referred to utilization of path, throughput and latency function. He reflected that lower latency implies less hardware and lesser cost.

Donald Newell from Intel remarked that this is a data-free argument. There are ways to hid latency with software. He was looking for a concrete data on this argument. He would like, for demo purposes, a set of applications for latency-sensitive productivity. Recio said, he would provide the required data.

3.2 Piyush Shivam, Duke University

On the Elusive Benefits of Protocol Offload

Co-author: Jeff Chase

Piyush began by introducing the offload controversy and NIC cards. He outlined recent technology trends and Moore's law, with an illustrative graph. Some minor doubts were raised on the correctness of the graph, which Jonathan Smith strengthened by declaring that the graph data is incorrect. Piyush offered an explanation, which was accepted!

The speaker continued with application trends, and prepared the ground for LAWS model. When he presented the LAWS ratios, Renato Recio objected that 1 ratio was missing, that for offload. Piyush said it will come later.

Piyush observed that LAWS captures application trends. Analyzing LAWS, he pushed for ignoring latency, and laying stress on

throughput speed. He went further to present some algebra to press his point.

When he pointed out the benefits of host-limited case, Raj Yavatkar wondered if it really mattered for network-intensive cases. The answer was deferred till forthcoming slides.

He then explained the benefits of network-limited cases with a graph. His cardinal argument was – for very fast networks, this case have very good benefits, but they are valid for a very few applications only.

He went on to counter the question – will network bandwidth outrun Moore's law?

When he approached the NIC-limited case, Renato Recio pointed out that some NIC designs can do what high-end systems do. Recio's question was if this particular case was covered by their suggestion/theory (of lag ratio). Piyush replied saying, one has to be limited either by host, network or NIC in any case, and anyway, that is covered.

After giving an overall picture, Piyush concluded that applications need to be understood to understand the role of TCP, IP offload, RDMA, etc. He also believed that point studies are misleading. He stopped after giving a brief LAWS analysis.

Questions & Answers:

Joerg Micheel from Endace started off by announcing that he is designing high-speed network interface for PCs. He operates on the parameter that the capability of CPU to process packets is the main factor. He then stated that the presented model is totally incorrect, as the factors are wrong. He believed that it does not matter what processor is present, the limiting problem is the bandwidth of the machine. Hence, the model does not fit. Piyush replied that he has considered the end-to-end throughput, and hence Jeorg's concerns are covered. The parameters do suffice. Joerg again objected that it has nothing to do with CPU speed, but with I/O problems! At this point, Jeff Chase intervened and said that the model is a simplification. the problem manifests itself in CPU speed. There exists no assertion that, by doubling the CPU speed, one can cut the latency. Hence, the model works. Steph Bailey seconded the arguments in favour of the presentation throughout.

3.3 Samuel Fineberg, HP NonStop Labs

Performance Measurements of a User-Space DAFS Server with a Database Workload

Co-author: Don Wilson

Sam began by explaining DAFS. After giving the characteristics of direct access transport, he gave some DAFS details. He compared inline I/O with direct I/O. He then briefly went over Oracle disk manager, prototype client/server, test system configuration, his experiments, ODM blast, its read and write comparison, ODM latency test and performance.

The speaker then gave some Oracle-based results and the Oracle TPC-H performance. He also pointed out that in the operation distribution, read operation was the largest component.

He later concluded that local I/O is still faster, that DAFS still has more capabilities than local I/O, and that memory registration is yet a problem with DAT.

Questions & Answers:

Jim Pinkerton queried if the speaker had a feel for memory registration bottleneck. Sam replied that the implementation was in hardware. When Jim offered that good caching algorithms exist, the speaker expressed his reservation saying, the problem with caching is that VM tables change often. OS cooperation is also required.

4. INVITED TALK: WU-CHUN FENG, LOS ALAMOS NATIONAL LABORATORY AND OHIO STATE UNIVERSITY

Bridging the Disconnect between the Network and Large-Scale Scientific Applications

The speaker started off by introducing the Green Destiny supercomputer. He presented the grand end-scheme: you put a CD (containing the software you want you install in the whole network) into your cluster/network, the software figures out everything automatically, and installs a clustered software in your network. Then he asked the question: does our network possess similar virtues?

Then he made a very humorous reference to the issue of how many staff was required to get a simple Internet connection started at SIGCOMM 2003 conference. He enumerated the various system failures and correlated them to non-transparency in Internet, using terms like NAT, DNS, etc. His main point was, even getting a simple Internet connection running involved knowing so much technical terminology, that it was very difficult for a layman to get it done on his own. The ironical comparison of the problem with the conference Internet connection was well-taken by the audience!

He then elaborated the argument that complete transparency is missing in the network. He observed: “Why can't I just plug my cable into the wall and get my Internet connection running?”

He explained what he meant by “disconnect.”

At this point, Renato Recio stated that the requirements of Lawrence Livermore are much more complex than the speaker's. He asked if the speaker had investigated that case. Wu replied that he had not considered that data with respect to the nuclear program. He further stated his belief that in all applications that he came across, the latency issue can be cushioned if it is not ... To this, Recio countered that latency interacts with throughput. As Feng was about to reply, Jeff Mogul intervened and terminated the discussion due to time constraints.

Feng went on to explain the wizard-gap problem. He concluded his solution by another humorous comparison of its performance with that of FedEx, and emphatically stated that his solution beats FedEx by speed and throughput, even though it requires around 30,000 professionals working at 4 different sites.

He finished his talk by briefly talking about the dynamic right-sizing.

5. SESSION 2: STORAGE PROTOCOL DESIGNS

5.1 Brent Callaghan, Sun Microsystems

NFS over RDMA

Co-authors: Theresa Lingutla-Raj, Alex Chiu, Peter Staubach, Omer Asad

The speaker commenced by explaining the need for RDMA as a transport layer protocol. He commented that NFS is an RDMA sweet-spot. He promoted RDMA as a new RPC transport. After briefly explaining some small RPC messages, he explained the moving of NFS data with RDMA. He later compared NFS throughput with that of TCP and RDMA. He concluded his talk by explaining the extended RDMA transport header.

Questions & Answers:

David Cheriton reflected that the speaker started with read-read protocol, while he himself had started with write-write protocol.

Combining the two, we now have a read-write protocol. He personally hated RTTs.

Brent replied that he too had started with write-write, but found it very complicated, he experienced problems with the server. Hence, he moved over to read-read. The main problem was that priority buffers could not be figured out. While read-read seemed much simpler, write-write is definitely more practical.

Jim Pinkerton asked why NFS/TCP performance appeared better than NFS/RDMA performance for small transfers. Brent explained that RDMA memory registration impact is greater for small transfers. A new read-write protocol will allow the client to reduce memory registration overhead.

5.2 Mallikarjun Chadalapaka

A Study of iSCSI Extensions for RDMA (iSER)

Co-authors: Uri Elzur (Broadcom), Michael Ko (IBM Almaden Research Center), Hemal Shah (Intel), Patricia Thaler (Agilent Technologies)

Questions & Answers:

Austin Donnelly from Microsoft queried if the proposal does not open an attack.

Mallikarjun replied that there are other iWarp mechanisms to deal with it. One example would be invalidating the S-tags. The damage is localized, some other I/O will be aborted, but it stops at that.

At this point, Jim Pinkerton, the Session Chair, intervened saying that there is a detailed analysis of security for RDMA in an IETF Internet-Draft.

David Cheriton raised the next question. The copy overhead does not register, there is a long history of dealing with virtual memory systems. We do not force revalidation, and some other such things. Then why not use the VM model?

The speaker replied that they do not have command parts, only data movement is involved. When response comes back, buffer associated with transaction will not be minimum. VM can be used locally.

Pinkerton said, should we incorporate iSER into VM?

Cheriton asked, why not use the same VM? There are a lot of synergies, and common software and techniques.

Pinkerton offered, it is functionally equivalent to VM, but not synchronized with local system. That is the main problem.

Ted Kim changed the topic by asking if the speaker had considered using IB. The speaker replied that iSCSI is defined over TCP. He wasn't sure if iSCSI can run over IB. He explained his position by stating his design goal – to allow iSCSI to run on generic RNICs specifically. One can always do vendor-add ops. Efficiency on generic cards is important.

Renato then commented that integration of this chunk with the processor may allow for pre-activity: processor with paging or virtual and physical mappings.

Pinkerton concluded, we saw new innovations in IB. We observed that the scope of the protocol expanded. The new development is that RDMA over TCP runs in the kernel in an optimized way. It would be interesting to note how more applications map with RDMA.

6. SESSION 3: NOVEL APPROACHES

6.1 Angelos Keromytis, Columbia University

High-Speed I/O: The OS as a Signalling Mechanism

Co-author: Matthew Burnside

Before commencing his talk, the speaker elucidated that he hails from a cryptography background, hence his paper/talk constantly refers to cryptography, and is based for the most part on it.

Questions & Answers:

Jim Pinkerton offered that memory is a concern because it is a peripheral. If we resort to using a buffer, the problem will dissolve. This suggestion was backed by many other people in the audience. The speaker admitted it partly.

Donald Newell from Intel said, complete parallelism has been achieved in the proposed method by moving memory to other devices, and then employing additional complexity to manage it. A better solution would be to employ memory parallelism closer to the process. More random external accesses will be available, and better performance will result. There are better methods available, more scalable and cheaper.

The speaker replied that he looked at programmable FPGA's, they weren't that complicated. However, he confessed that he didn't have the figures for performance comparison.

Newell then asked, what kind of applications the speaker would be interested in demonstrating his work – databases or simpler applications?

The speaker countered that Newell was thinking of too high a level of complexity. He does not want any SQL queries, but a much simpler process.

Renato then commented that other control operations would need to be appended. The speaker would need some kind of specialized control, etc. thereby making the whole scheme very complex. In other words, he would not be able to escape a certain degree of complexity anyway.

David Cheriton said, what the speaker described for peer-peer is all transforming into a network, and then, the speaker walked into another discussion. But Cheriton questioned if it runs over Ethernet, for he believed, in the end, its all just Ethernet and some devices.

The speaker agreed that the next step indeed would be to see Ethernet, as there were many interesting issues involved there. However, he maintained that all the devices are interconnected, and one may run IP over it.

6.2 Kieran Mansley, University of Cambridge

Engineering a User-Level TCP for the CLAN Network

The speaker started off by explaining his problem – the networks have become faster, and so have the transmission speeds of packets, but the overhead to deal with the packets is still the same. And this requires CPU cycles. How do we tackle this?

After referring to some possible solutions, he arrived at CLAN networks, and explained in depth its user-level stack architecture.

Jim Pinkerton asked for some figures regarding bandwidth and latency, which the speaker promptly supplied.

Then the speaker spoke on true zero-copy transmission.

He later summarized his talk saying that the TCP/IP stack should be moved to the user level, and that the retransmission should be handled by the gateway.

Questions & Answers:

Renato Recio argued that the speaker might still need a copy operation to remove headers from the stream, only that now it would be middleware headers rather than TCP/IP headers. The speaker agreed.

Jim Pinkerton first commented that only the transmit path has been considered. Then he asked what the speaker meant by an "asynchronous API." The speaker replied that the protocol processing will have to be done at some point, either immediately at the time when asked, or sometime later, even if the application is blocked now, or later.

Stephen Bailey doubted if the socket API really required the copying. He expressed that the speaker had not changed the socket API per se, but only the interface (in a rough sense), according to people's expectations. He summarily believed that what the speaker had done was not really required. The speaker replied that the kernel could spend a lot of time servicing the queue, and as a result would have to perform a copy without these changes.

6.3 Rolf Neugebauer, Intel Research Cambridge

A Case for Virtual Channel Processors
Co-author: Derek McAuley

Questions & Answers:

Mallikarjun Chadalapaka raised the first question. He surmised that constructing the VCP would involve a tremendous amount of effort – to take the TCP stack and iSCSI and then make the VCP. He asked if the speaker could specify what work was involved in defining a VCP.

The speaker agreed and said that he was currently in the process of doing it. He confirmed that it did involve a TCP stack, a device driver, and putting it together with an API. However, he believed that a minimal OS running on top of a VCP is enough to get a TCP implementation running.

Chadalapaka further queried if in the case of iSCSI resources, one needed separate NICs for TCP and SCSI. The speaker replied that there exist NICs that multiplex on a large level. Many new NICs have been introduced of late. They later demux to different functions.

Jeff Chase then reflected that the first talk proposed to add a bunch of data copies, that would certainly require a lot of support in VM. The speaker agreed, but said that the VM already does it, it even has an interface for this function. When Chase asked if he had full control of VM between paging, he replied in the positive.

7. GENERAL DISCUSSION

All attendees were requested to air their views on appropriate issues.

Jonathan Smith said, are there any economic factors to change the bias in memory architecture towards faster memory rather than larger memory? He further said, do we always have to work around it, or can we hope for something better.

Stephen Bailey proposed stream benchmarks. He said it is faster, exponential. The number of pins is a limitation, it does not grow. And so is bandwidth. But the stream is quite good, and fast. We have 20 years of stock. Memory will be the bottleneck.

David Cheriton expressed his inability to comment on physics. However, he said he would comment on old perspectives. He would like to get rid of that copy overhead. It should be easy to transition to new, faster mechanisms. We do not notice success due to page manipulation. He has a vision to see 10 Gig Ethernet coming directly into the processor, and memory systems being accessed through VM, paging, and so on. There is not a drag behind, but it gets eaten up by little things like checking this or that. To fold in security, the complexity must be reduced, so much so that it is understandable by my mother. We still are a far way from that.

Jim Pinkerton would like to see network speed approximate memory speed. He believed that enough technology is lying around for 20 years. The network is no longer the orphan child, it gets the first technology. It has the same technology as does a CPU now.

Steph Bailey commented that disk channels have consistently delivered 10x the bandwidth of NICs to the same system. With the right hardware, NICs should be able to achieve parity with disk adapters (picking up a factor of 10 performance) without any host hardware changes.

David Cheriton said that in the network world, people talk about latency sensitive applications. This market is worth $800 billion, comprising block storage I/O, fibre channel, whatever. Data management needs to be centralized. How does one run Oracle on a Sun server 1000 miles away, when the fibre is cheap and has a capacity of 40 Gig? How do we design a protocol that works reliably, securely and does not get overloaded. That is the grand challenge, that we have to do right. After all, there will always be 1 orphan child.

Jeff Chase doubted if it really is a latency-driven world.

Cheriton reconfirmed that it is, down to the level of 100 microseconds range. When dealing with disk I/O, the numbers are in milliseconds range.

Jeff Mogul suggested that if we replace disks with MEMS devices, then we might end up on a better price curve. In that scenario, 1 ms will be important, and maybe even 100 microsecs.

Jeff Chase suggested that most systems are throughput-sensitive, not on latency. For IOP, Internet requests served per second is the sensitivity factor.

Jim Pinkerton remarked that everything is opinion-based, and not fact-based. He raised the question that what areas need more research.

Donald Newell backed him, saying that this is the most common argument. People often come up with bizarre requirements.

Renato Recio commented on the areas for additional research. According to him, segmenting out the problem is oversimplification, and that is the problem. There is a big problem with Java, C#, etc., how do we get advantage in these areas. It addresses a large chunk of space, and involves a lot of money. It might even take application changes.

Jim Pinkerton expressed that whenever application change is required, it introduces a new world. No features exist for it.

Jeff Chase referred to the talk by Piyush Shivam, and asserted it is all facts, not mere opinions. Once we can figure out the values for the parameters, we can visualize real environments. The problem is that we do not know what really matters, and hence estimating the parameter values is difficult.

Stephen Bailey cautioned all to be careful to go out and test something.

Jeff Chase spoke about some service benchmarks, saying that he did some work there.

Jeff Mogul then said that we all have focussed a lot on performance. The theory is that if we employ RDMA, we get better performance. But he believed that the right way to look at it was, does RDMA enable systems vendors to get acceptable performance with cheaper hardware? If noone uses hardware-RDMA adapters, is RDMA really better? What is the right question to ask – faster, faster... or cheaper, cheaper...? He firmly believed that cheaper is better than faster.

Jonathan Smith first expressed agreement for the previous comment. Then he said that our research is greatly hindered by absence of data, especially in fields of security. He urged all if we could come up with an anonymization policy to facilitate better availab-

ility of data.

Brent Callaghan drew attention to the fact that use of sockets was primarily responsible for the popularity of Ethernet. He then asked if we have the right API for RDMA as yet. It may be RPC, or it may be MPI. We do not know for sure.

Renato Recio then said that it takes time for applications to develop.

Jeff Chase commented that in the short term, it is more economical to buy more servers rather than hire people.

Lastly, Kieran asked how to make the API better for applications.

Revisiting IP QoS: Why do we care, what have we learned?
ACM SIGCOMM 2003 RIPQOS Workshop Report

Grenville J. Armitage, Workshop Chair
Swinburne University of Technology
Melbourne, Australia
garmitage@swin.edu.au

***Abstract*- ACM SIGCOMM 2003 included a number of workshops, including the all-day workshop "Revisiting IP QoS: Why do we care, what have we learned? (RIPQOS)." The goal of RIPQOS was to critique the evolution and deployment of IP quality of service (QoS) mechanisms, from both the research and operational community perspectives. The workshop's name was a challenge to all interested communities to reflect on whether IP QoS has lived up to the hype or whether it is simply misunderstood. The workshop saw 6 papers, 2 short papers, a discussion panel, a range of opinions and lots of questions. This report attempts to capture the essence of our workshop's discussions, presentations and experiences.**

Keywords- IP, Sigcomm, Workshop, RIPQOS

I. Introduction

There are few topics in the IP networking research and operational communities that elict as much inconsistent opinion as "quality of service" (QoS). The very premise of QoS appears, on the face of it, to contradict the guiding principles of "best effort" service, a service model that has seemingly underpinned IP network engineering since the very beginning. Segments of the research community have taken the complexity and apparent contradiction as a challenge, and produced a substantial body of strong theoretical work showing how IP networking can evolve to support a variety of QoS schemes. Large segments of the operational community simply cannot see the point of adding QoS to networks that are humming along quite nicely as they are. A broad spectrum of people can't entirely agree on what QoS actually is. What's going on here?

The RIPQOS Call for Papers deliberately began with a provocative statement:

"*For over a decade the Internet engineering and research community has debated, designed, and ignored IP Quality of Service tools and techniques. There's a sense that something might be needed, but little agreement on why and who will pay. At times the very notion of QoS has seemed to be a pointless waste of time, almost a solution waiting for a problem. This workshop is an opportunity for researchers and practitioners to discuss the history of IP QoS research and development, review what could have been done better, and perhaps develop a new focus going forward.*"

We went on to give some specific questions that this workshop might consider:

"Papers are invited that provide well-argued opinion, speculation, or contrary positions. For example:

- *IP QoS schemes never quite seem complete. Is this just a great research game for academics?*
- *Where's the money? How do we make IP QoS pay when typical Internet applications don't care, and the user's don't know any better?*
- *Will online, multi-player games be the market segment that justifies end-user/access ISP investment in IP QoS tools and solutions?*
- *Isn't more bandwidth the answer?*

Of particular interest are papers that critique the evolution of IP QoS solutions to date and/or explain what sort of applications and user mindset will need to emerge before IP QoS solutions become cost-effective for ISPs to deploy."

A bit pointed? Yes. Totally unfair to the research community? Well, no, not really. Our goal with RIPQOS was to start dialog on how IP QoS techniques and methodologies fare in the operational world outside of simulations and testbeds. The research community has taken IP QoS a long way. The question we wanted to consider at RIPQOS was whether we're entering a brave new world of QoS deployment or whether IP QoS should simply Rest In Peace.

In the end RIPQOS enjoyed 6 full papers, 2 partially developed papers and an invited panel discussion to wrap up the day's proceedings. The first four papers were grouped into two morning sessions under the heading "Challenges" - opinions on where QoS research should be going, a question of whether QoS has "failed to thrive", observations on how QoS deployment must attend to operational and commercial realities, and a review of how DiffServ is solving real-world problems for real ISPs and customers today. After lunch we had two papers under the "Lateral Thinking" session, looking at two quite diverse topics of QoS and Denial of Service, and the potential for networked Games to emerge as an important QoS-sensitive application. The "Short Papers" session saw a discussion about QoS as a risk management tool and a brief proposal to add variable congestion control algorithms to existing transport mechanisms. Our discussion panelists wrapped up the day by considering the question of where the research community should go next in order to expand the role of IP QoS in operational IP networks.

The workshop proceedings are available through the ACM Digital Library [1]. What follows here is a summary of each session.

II. Challenges

Two sessions covered the Challenges, four papers articulating some different perspectives on IP QoS past, present and future.

A. *QoS`s Downfall: At the bottom, or not at all!*

Jon Crowcroft (University of Cambridge) opened the day with a walk down memory lane to the Cambridge Ring, a "bogograph" with broad sweep armwaving, a reminder that QoS deployment means meeting the needs ot stakeholders, and an argument that some minimal degree of QoS mechanism needs to be embedded in the very lowest levels of our networks [2] .

Jon began by clarifying that, clearly, all networks deliver some sort of "quality". But in the sense we're discussing QoS here, the issue is about a network's ability to offer different (and differentiable) levels of service quality over a shared infrastructure.

More importantly, there's a real problem for researchers in this field that isn't always obvious. Jon argued that we become trapped at particular points in the cyclical nature of the problem statement. The ratio of access network and core network capacities change over time, moving the congestion (and hence QoS) problem back and forth. QoS research tracks resource contraints in networks. Hence any given researcher's work tends to be a trailing indicator of where the access/core ratio stood at the time they embarked on their particular QoS scheme.

Many of the different viewpoints held firmly by people across our networking community may perhaps be understood in the context of the prevailing edge/core capacity ratio at the time each person embarked on their network engineering and research careers. Failing to realize that we're victims of the "wheel of time" will ensure we do not break out of the cycle. [There was also a bogograph ("bogus graph") to back it all up, showing a sinusoid of "core to access capacity ratio" with a period of roughly twenty years.... and, as Jon readily admitted, scaled primarily to suit his talk.] Nevertheless, an intriguing position that our research is blinkered by historical trends.

Another issue identified in Jon's talk is the question of knowing who your stakeholders are when discussing, designing and proposing QoS schemes. Computing people, telecoms people, service operators, users/consumers.... they all have different timescales over which they evaluate the cost-benefit trade offs of new schemes and services. The drive to "converged" networking means a larger and more diverse range of applications, which has created demands for a multi-service (and hence QoS-enabled) IP layer. But each stakeholder has arrived at different points in the historical capacity ratio cycle, and thus perceive the technological problems differently.

Jon"s fundamental points came towards the end:

- We need QoS at the lowest layer, below IP, and it only needs to be simple – a two-level (one bit) scheme will suffice. Overlay schemes cannot support "better" QoS control if the underlying links (networks) don't have at least two levels of service.
- The lowest level QoS mechanism needs to be exceedingly cheap to implement and deploy, encouraging innovative use with minimal inconvenience.
- There's no need for more "QoS architecture" work at higher levels, its been done. We need to map QoS to marketable services, close the gap between mechanisms and revenue.
- We need QoS mechanism in the core. The current core/access ratio is trending towards the core being a congested resource, despite all current evidence to the contrary. Aim for the future, don't chase the near-term issue. Put two-level QoS into the optical core.

In other words, QoS needs to be deployed "bottom up", yet the mechanisms at the lowest levels need not be complex at all. QoS needs to be "sold" correctly to the various stakeholders – if we hang around at "layer 8" (the political layer of the OSI stack) we'll continue to solve for congestion/capacity limits that have changed by the time our solutions appear for deployment.

B. *Failure to Thrive: QoS and the Culture of Operational Networking*

Gregory Bell (Lawrence Berkeley National Laboratory) took us on a different tack – he provided an operational engineer's contemplations on why IP QoS schemes have failed to thrive in enterprise networks, and the entrenched R&D methodologies that may yet ensure no complex IP QoS schemes ever manage to take off [3]. He observed that his insights are taken from supporting a research institution with about 80 subnets and 10,000 hosts – an enterprise rather than ISP environment.

"Failure to thrive" comes from the growth of children - thriving is the norm and anything else is cause for alarm. In networking it might be argued that withering is the normal course of events for most protocols and architectures. Yet Greg observed that IP QoS should be thriving by all accounts, considering the stature of the researchers who have done significant work in the area over the past decade and the abundance of literature generated in the area.

So why has IP QoS failed to thrive?

There appears to be a structural rift between the designers of protocols and QoS architectures, and those whose job it would be to operate such QoS-enabled networks. There's a disconnect in how researchers and operations people view the relationship between system complexities and system failures. Both communities recognize that complexity multiplies the potential for system failures. What the research community doesn't often internalize is the fact that failures in deployed equipment are often due to buggy implementation rather

than unintended operation of a perfectly implemented protocol. Operations engineers learn the hard way to assume failures will occur and act to avoid deployment of anything that increases the chance they'll be working weekends to fix their network.

So the real question faced by IP QoS community is "is this deployable?" Greg gave us some history of a related service with a checkered past – IP multicast. At LBNL the enterprise networking team had a range of bad experiences with routers running shipped, yet buggy, IP multicast code. Greg's main point is that the IP QoS research community needs to recognize that operations engineers are as afraid of buggy QoS implementations as they are of the architectural issues surrounding QoS-enabled service deployment.

Deployment thus depends on a variety of factors - QA by vendors, critical mass of users, debugging tools, enterprise knowledge, trust between neighbouring domains, and business case. For the operational staff the existence of debugging tools is crucial – if I deploy a new technology to offer a new service, where are tools I can use to cover myself when things go wrong (network failure) or users complain they aren't getting the XYZ service level they thought they would? And a related broad concern, can this new technology be deployed incrementally in such a way that it doesn't disrupt existing best-effort IP service?

Faced with these concerns it is hardly surprising that many network administrators will seek out the safer route of just adding more bandwidth. Greg sounds a cautionary note to the QoS community not to dismiss "throwing bandwidth at the problem", because in reality this is often a reasonable and pragmatic engineering response to network congestion. It certainly seems more appealing to network engineers than "throwing protocols" at the problem! The reality in many enterprise networks is that adding links is a well understood and low-risk action that usually solves impending congestion issues. (Greg acknowledged that the economic and practical constraints facing ISPs may differ greatly from those in enterprise networks, he discussed some examples where long haul telco links simply couldn't be upgraded easily.)

Perhaps the essence of Greg's cautionary tale is contained in this quote from his paper:

"Attempting to architect QoS without taking into account its economic and institutional context is roughly analogous to designing a city with reference to local culture, climate or geography."

Greg's talk concluded with some animated discussion of his conclusion that unless the research and development communities start paying attention to the deployment issues, QoS will continue to suffer from a failure to thrive.

C. *Beyond Technology: The Missing Pieces for QoS Success*

After the morning break Carlos Macian (University of Stuttgart) gave us a view from the other side [4] – the issues that affect ISPs more so than the enterprise network focus on Greg's talk.

Carlos began by observing that the ITU and IETF have different perspectives on QoS - the former seeing it as the collective service performance affectign the user's satisfaction, the latter seeing it as a set of end to end performance metrics to be met along a network path. Yet there's a common issue – QoS is a "*...measurable performance that satisfies some(one's) demand*".

Although reliability, availability and security are also metrics by which QoS can be evaluated, Carlos specifically focused on the timescales of packet loss and jitter. He also noted that QoS is not simply a network issue – the end user applications have a significant impact on the end user's sense of service quality. Thus QoS is truely and end to end issue.

And the main motivation for ISPs to deploy QoS is money. How to provide priority service to someone who would be willing to pay for better than best effort?

Overprovisioning is often not a viable option for ISPs, ultimately boiling down to economic realities. Links of suitable capacity may be available and yet priced too high to justify, or there may simply be no links available between different sites on a provider's IP network. Alternative mechanisms are then deployed to share the existing links in a controlled and differentiated manner (using the more general notion of "differentiated", not the IETF's Differentiated Services model necessarily).

To be successful QoS needs a mixture of technical mechanisms (buffer allocation, capacity, protocols) and economic consideration (price differentiation, market segmentation, service bundling, perhaps auctions).

Although the technological pieces may be largely in place, the business environment still has a long way to go. The solution needs to be completely end to end or not at all, which implies inter-domain relationships to support QoS across provider boundaries. This creates a hugely complex problem for relationships between ISPs – indeed, the traffic exchange relationships need to be more explicit and formalized than they are today. Network operators may need to expose internal performance data about their networks, which is understandably sensitive information. Will we end up with market consolidation? Enforced interconnect rules through regulation? Stagnation by simply letting overcapacity solve QoS where it can and ignore the rest of the network?

So back to a critical question – how can money be earned? The QoS area seriously lacks a billing and accounting model that can work inter-provider and inter-domain. The telephone industry is not a good guide because it essentially offers only one level of service – either the phone call is there, or it is not. Billing and accounting are thus simplified by aggregating call durations. There's no simply analogy to IP networking. And since telephony providers are already making far more profit than ISPs, why would telephony migrate to IP? Some IP architects have proposed "brokers" to handle accounting and billing between domains, but this is still very much work-in-progress and doesn't address the need for compelling business reasons for domains to play nice with each other.

In the end Carlos summarised that we have no integrated QoS architecture, but even more importantly the community needs to address business models and trust models between providers.

D. Deployment Experience with Differentiated Services

Rounding off the morning was Bruce Davie (Cisco Systems) with an entertaining and far more positive position on IP QoS – despite the doom and gloom from certain quarters there is a thriving deployment of basic Differentiated Services (DiffServ) in real-world networks solving real-world problems today [5].

Bruce started off with a fair observation that the very premise of RIPQOS was somewhat confrontational to the QoS community. He felt obliged to step forward with a DiffServ success story and to ask whether there were lessons here for the broader deployment of IP QoS mechanisms.

There are at least two valid definitions of QoS – an application-based "what can i get from the network?" or mechanism-based "what technologies can I utilize?" Bruce's talk would focus on the use of certain mechanisms on the presumption that overprovisioning was not the solution (otherwise we should simply stop all QoS R&D work). Traffic enginering was not defined as a QoS mechanism for this talk. (Bruce also observed that if the network offers good-enough service without additional service levels this could be considered "QoS".)

The first critical clarification Bruce made was to observe that most DiffServ deployment today was in private networks or managed VPN contexts. It is commonly deployed by ISPs supporting customers who have high expectations of mixing Voice over IP (VoIP) and regular IP traffic over shared links running at very high utilizations. Usually the customer is driven to run their external links at high load because such links are exceedingly expensive. There's little to no DiffServ deployed in the core of the public internet.

VPN scenarios are tractable because they avoid the issues of maintaining QoS across inter-domain or inter-provider boundaries, or having to mix IP traffic from uncontrolled and uncontrollable sources. The VoIP sources and sinks are usually under the control of the cusomer directly asking for QoS, or the provider offering the QoS-enhanced service.

Typically the VPN customer has a number of remote sites connected to the VPN provider'score network through low bandwidth links. VoIP traffic is marked with the DiffServ "Expedited Forwarding" (EF) bit, and routers on the customer premises edge and provider edge provide priority treatment to packets with the EF bit set.

DiffServ is an exceedingly useful tool when edge bandwidth is an expensive commodity. It is especially useful when migrating customers from older private networks - such customers expect their voice trunks to operate much as they did when using e.g. Frame Relay, even though they're now sharing their external links with other intra-VPN IP traffic.

At least one national carrier (Telecom Italia) has deployed an entirely self-contained VoIP network for public telephone calls, with DiffServ turned on. Since the VoIP gateways are under the carrier's control, provisioning and policing are tractable problems.

Bruce then considered the situations where DiffServ is not attractive, or very difficult to use. For example, many ISP customers do not have the same expectation of service quality as someone who migrates from traditional circuits like Frame Relay. Regular IP access is a difficult environment to offer QoS assurances because we run into inter-provider issues, and it is hard (or impossible) to establish a-priori the set of communicating sites (and thus almost impossible for the ISP to internally provision their network correctly).

It is also hard to sell QoS mechanisms into environments where Best Effort service appears to be working just fine (e.g. where bandwidth is not a problem) – what is the benefit to a customer of a premium service if BE is fine? Who would want to be the first ISP to downgrade their BE service below their competitor'sjust to offer a "premium" service that's about as good as their competitor's BE service?

DiffServ is also a hard sell when the customer only expresses their needs in terms of end to end performance, rather than per-hop behaviors (PHBs). And finally, inter-provider QoS agreements are <u>really</u> hard!

Bruce wrapped up by observing that future QoS research needs to look at deployment issues rather than develop new mechanisms. The catalysts for future deployment of QoS will include a subsiding of the bandwidth glut, development of customer demand, regional ISPs co-ordinating together, and the development of standard service definitions (to allow comparison shopping by customers).

III. Lateral Thinking

Coming back from lunch we jumped into the Lateral Thinking session with two quite diverse topics - Denial of Service and the potential for networked Games to emerge as an important QoS-sensitive application.

A. Quality of Service and Denial of Service

Ben Teitelbaum (Internet 2) introduced an interesting argument that QoS solutions will never see meaningful deployment if they are not designed for worst-case conditions – i.e. when network components are suffering from denial of service (DoS) attacks. In other words, most QoS researchers should be designing QoS schemes that protect against (and work in the presence of) adversarial conditions. However, if QoS is deployed to protect against DoS then how do customers ever verify they're getting DoS protection?

Ben'sfundamental definition of QoS is the regulation of the impact of congestion. The reality of today'sIP networks is that "best effort" is generally pretty good, and most of the time operators find "adding bandwidth" to be the operationally and financially pragmatic solution to growth in traffic loads. QoS schemes are attractive only insofar as they offer protection against

service degradation during worst-case network traffic load conditions. QoS is not attractive if deployed primarily to optimize an operator'suse of their fibre capacity.

Ben identified two broad classes of QoS - "elevated priority service" and "non-elevated priority service". The former delivers treatment equal or better than the default best-effort service (e.g. based on the DiffServ Expedited Forwarding or Assured Forwarding models), while the latter provides something equal or lesser than default best-effort service (e.g. scavenger services that make use of spare capacity without disrupting existing best effort traffic). Ben observed that only elevated priority services can protect against adversarial traffic overload conditions. Although easy to deploy, non-elevated and default best effort can both collapse to zero service under suitably severe congestion conditions.

Defining a DoS attack is also problematic. Is it a security or resource management issue? How does an operator discern the difference between legitimate and adversarial increases in traffic load? If I'ma researcher pushing the limits of a gigabit link am I "attacking" people along the path during my test? The mere fact that a network resource begins to congest or become over-loaded cannot be used as a criteria, so it becomes a question of intent – a difficult concept to infer with completely automated tools. Ben suggests that operators instead look to protecting certain traffic known to be legitimate rather than attempting to automate the detection of illegitimate traffic. This is QoS by any other name.

Elevated priority schemes may or may not provide protection against DoS (e.g. against a third party attempting to disrupt the service guarantees offered between any two hosts by pushing the network conditions outside normal parameters). Unfortunately, any elevated priority scheme that does not protect against DoS will be essentially undeployable - it adds cost and complexity while offering almost no benefits over best-effort during good periods and no guarantees when the network is overloaded.

The costs and complexities of elevated priority schemes that can protect against DoS are similar to ones that do not, yet the benefits are tangible during times of network overload. Ben observed that this makes such schemes potentially deployable if the costs are low enough. In other words, a QoS scheme will only be deployable and attractive in the field if it inherently protects legitimate traffic from all other sources of traffic competing for network resources – legitimate or otherwise.

In essence DoS management is a QoS problem. Ben identified a number of ways that QoS researchers need to re-orient their thinking on this point. First, start thinking like an adversary. Consider the various ways an adversary can disrupt traffic flows along your statistically provisioned network paths (e.g. compromised hosts elsewhere on your network) and ask how existing QoS technologies could mitigate the impact of a DoS attack. Develop a minimally complex QoS architecture that protects against DoS, then we'll have a better chance to encourage deployment.

Of course, Ben then pointed out the obvious problem – customer verification of the offered service. How can a customer confirm they'regetting protection from DoS? Generating a DoS attack on their own ISP is unlikely to be met with much delight in other quarters.

Ben'stalk left us with more questions than answers. He definitely challenged us to consider DoS-protection as the most viable selling point for QoS technologies, because under "normal" network conditions best effort service and engineering practices are usually the simpler solution.

B. Networked games --- a QoS-sensitive application for QoS-insensitive users?

An entirely different talk was presented by Tristan Henderson (University College London) on the issue of whether networked games really demand QoS as many people have asserted. Tristan presented the results of some research he'ddone into people'ssatisfaction with playing Half-Life/Counterstrike online.

Tristan first observed that online games could be broken into three broad categories:

- First Person Shooters (FPS, such as Quake3, Half-Life, or Doom)
- Massively Multiplayer Online Role Playing Games (MMORPG, such as Everquest or Star Wars: Galaxies)
- Real Time Strategy (RTS, such as Civilization or Age of Empires)

Typically these games would use UDP-based client -server communication. Network latency is broadly a major concern for each category, more concerning for fast-paced interactive games. A range of studies unrelated to online gaming have shown 100ms to 300ms as the acceptable range for round trip delay in human interactions. Some game related studies have claimed limits down around 100-250ms.

Tristan'squestion was basically whether players actually really cared about QoS, and in what terms did they actually articulate their thoughts about QoS. He made the observation that although we intuit a need for QoS to make games popular, online games appear to be remarkably successful anyway in today'sIP networks. So is there really a need for "good" QoS for games?

Tristan set up two Half-Life/Counterstrike servers in London, equivalently connected to the Internet except that additional controlled latency could be added to one or the other server. The servers both ran for a few months to build up their popularity, and then Tristan began adding nominal latency of 50ms to one and then the other server. Usage patterns clearly showed that simply adding 50ms would discourage people from even joining their server (client software has a method for potential players to rank servers according to network latency before playing).

The server logs also showed that players tended to stay on a server once they'd joined even when network latency got worse for short periods of time. Interestingly, the relative increase in delay didn't seem to affect the likelyhood of a player leaving. Players who'd already been playing a long time seemed to be more tolerant of brief bursts of additional delay (perhaps tolerance increases with immersion in the game). But note that players who play regularly (as opposed to players who've been logged into a single playing session for a long time) are no less likely to leave when delay gets worse than a player is not a regular on the server.

Finally, there was clear evidence that a players "success" (in terms of kills, or "frags" per minute) was adversely affected by increased latency, as was the likelyhood of the player dying frequently.

Tristan used these insights to ask an unsettling question – if players are relatively insensitive to network delay once they'd decided to join a game server, how will an ISP attract these typically price-sensitive customers to a premium-cost IP service with improved QoS? Or put another way, will a player pay for QoS improvements when they appear insensitive to QoS degradation? (Tristan also commented that in some earlier work he'd surveyed game players who indicated a strong unwillingness to pay for network QoS on principle.)

Tristan noted that his study only considered absolute delay, and did not test player's sensitivity to jitter or packet loss. There's more research to be done to better understand if those QoS parameters have more influence on player satisfaction than raw latency. There's also some interesting questions about how an ISP might vary the QoS delivered to any particular customer based on how far "into" a game they are (e.g. good at the beginning (to attract players) and then gradually declining to a tolerable level as the game progresses).

IV. Short Papers

Two short presentations came after the afternoon break, representing ideas under development at the time of review.

A. *What QoS Research Hasn`t Understood About Risk*

Ben Teitelbaum (Internet 2) came back to the stage with a short discussion that he acknowledged wasn't necessarily consistent with his first presentation. (The chair also noted that it was a quirk of double-blind reviewing that led us to have two papers by Ben!)

Ben's basic thesis is that neither customers nor ISPs need or want hard performance guarantees. Rather, each wants tools and understand and manage risk. He observed that the design goals for much QoS research can be captured by words attributed to S.Keshav, "*The Holy Grail of computer networking is to design a network that has the flexibility and low cost of the Internet, yet offers the end-to-end quality-of-service guarantees of the telephone network*". Ben's contention is that this design goal completely misunderstands the market's real use for IP QoS, and that engineering for true end-to-end QoS destroys the apparent cost advantage of the Internet.

QoS is essentially a cost/benefit proposition. Many technical solutions for IP QoS focus on congestion-management or congestion-avoidance. However, avoiding congestion entirely (or close to entirely) can be a rather expensive proposition. What other tools can be brought to bear on the process of managing the risks associated with using a congested network?

Customers are essentially rational, although they have complex and divergent utility functions. They typically want the ability to trade-off between a number of service criteria, including good performance, low costs, simple pricing, low transaction overheads, and means to manage exposure to worst-case performance. Technical QoS schemes can only remove a component of risk exposure, and at significant cost.

Perhaps an alternate goal for QoS research is "*How can network services offer customers and providers flexible management of exposure to poor network performance?*"

A complete risk management solution needs to include economic tools. There are few (if any) markets where customers actually demand infallible service. Typical systems use a mix of technical and economic/regulatory mechanisms (e.g. warranties, insurance, and certification). Warranted performance appeals to customer's desire for simplicity in costs, but demands new tools for providers to correctly estimate their ability to warrantee any particular service agreement. The likelyhood of customers demanding compensation for performance degrading outside warranteed levels leads to opportunities for third-party performance insurance. Insurance is a monetary risk management tool, there's no reason it cannot be applied to the network service provision industry. Yes, this would introduce a whole new insurance industry and require government involvement to establish rules for liability, etc. But a necessary evil. And yes there will be costs to administer, but in most places insurance acts to lower risks for goods and services. And finally, certification processes and procedures would complete the tool-kit, allowing customers to compare ISPs and requiring performance monitoring and reporting by ISPs along the lines of the phone companies today.

QoS needs a multi-disciplinary approach from this point forward if we are to see any truly deployed solutions at all.

B. *Internet Service Differentiation using Transport Options:the case for policy-aware congestion control*

Panos Gevros (University of Cambridge) presented a rather different discussion, on technical means to optimise TCP's congestion control behavior in a dynamic fashion.

Panos first observed that QoS allows us to differentiate between traffic belonging to different users and to guarantee quantifiable performance levels. Most mechanisms discussed so far have been network-centric, e.g. end-to-end guarantees, router mechanisms such as

DiffServ, etc. Such mechanisms may be OK for small scope environments, but there is a combinatiorial explosion when we try and use QoS for the whole Internet.

Panos's basic thesis is that we should engage the endpoints in the control loop – modifying each endpoint's congestion control behaviour to influence performance. Many networks are well-provisioned. So if there is no congestion, users might be able to transmit more aggressively (compared to regular/default TCP as currently deployed). We could modify TCP's slow start, congestion avoidance or retransmission timeouts.

The idea behind "transport options" is that an ISP could offer a service that changed the congestion control behaviour of endpoints dynamically, on behalf of the user, according to certain classes of service offered by the ISP to the user. Classes of service are offered in relative terms (no guarantees) - so you know that a class is worse than another, but that is all.

This work on transport options is still in progress, and has a number of open issues. It offers to put ISP congestion policy control further out to the network edges and into the operating systems of end users.

V. Discussion Panel

We wrapped up the day with a short, informal panel discussion involving Jon Crowcroft (University of Cambridge), Bruce Davie (Cisco Systems), Jennifer Rexford (AT&T Research) and Ran Atkinson (Extreme networks). As chair I basically let each person comment on what their main thoughts were after participating in the day's event.

Jon: Control planes are complex - a signalling network is as complex as the network itself. We need a distributed partition to partition BE from EF/AF. But how to bootstrap? Network is ever-evolving - a moving target. Need a new model of the evolution of the network so we can predict where this moving target will be?

Bruce: Range of opinions today, from DiffServ is a success to DiffServ is a fundamental failure to QoS is a solution in search of a problem. Is the glass half-full or half-empty? How can we get Internet-wide QoS?

Jennifer: How can you have QoS when:

- A typo by a network operator can bring down service? Half of network outages come from misconfiguration
- Routing anomalies might throw away your traffic?
- Users don't know how to predict their bill?
- You don't know who to blame for a QoS violation?

Ran: Universities are more interested in degrading traffic – e.g. degrading game or p2p traffic. Degrading is better than ACLs since applications will port-hop if blocked. 5-10% of universities employ some sort of ACL to downgrade undesirable traffic. Some universities use quota-based systems, e.g. a rate-limiting up to a quota per IP address.

Extreme's products are QoS feature-rich, but most customers don't switch those features on. Ran thinks there is less QoS out there than Bruce does.

In the late 80s the DDN (Defence Data Network) in the US, based on T1s used John Moy's OSPF which supported the TOS bit. So e.g. file transfers would have lower precedence and be routed via a satellite (which had higher capacity). TOS bit was used as in RFC791.

Sprint will (for a fee) use CBQ between customer routers and the aggregating router. DSCP bits are cleared in the core because in Sprint's core there is a funny DiffServ setup for prioritising BGP traffic (as opposed to customer traffic). Also Sprint core is overprovisioned.

The IAB Network Management workshop had two points: operators desperately need better config/management tools (most ISPs use a set of Perl/MySQL/Tcl/Expect scripts to manage routers), and reducing operations costs.

Bandwidth is increasing faster than our ability to encrypt or authenticate data. This is a problem for interdomain QoS since ISPs will be unwilling to forward marked packets unless they can verify that it is not a DoS.

Maybe ISPs could just negotiate levels of EF between each other. e.g. renegotiate every day, if a customer sends too much traffic, that is their problem. Need to be able to assign blame.

As to each panel member's favourite research problem:

Jon: a model of the complexity of network architectures (not just the complexity of components)

Bruce: interprovider QoS

Jennifer: configuration management - models of protocol configuration state

Ran: configuration management - how do I configure a network (not just a single box)

kc: how to get funding? The NSF won't consider a lot of these problems as research. Maybe Dave Clark's knowledge plane could be used as a platform for research.

What is left for "traditional" QoS research? Bruce: no more queuing algorithms!

VI. Conclusions

It would be inappropriate to conclude that RIPQOS has answered everyone's questions about QoS. But the day did see two broad themes emerge:

- There's lot of respect for the complex theoretical work that has been done on device-level congestion management schemes in bursty, best-effort IP networks
- If IP QoS is to be truly deployable the research and development communities need to shift gears and begin answering the market's actual questions – use

the technical device-level mechanisms to develop systems-wide toolkits for monitoring and managing QoS schemes, and recognize that QoS is only part of what customers demand from their ISPs.

The key message from RIPQOS is that QoS is not dead, but as an IP QoS R&D community we need to reach out and include business, systems control, and marketing expertise in our efforts to get IP QoS meaningfully deployed and used.

Acknowledgments

This report relied heavily on our RIPQOS scribe, Tristan Henderson, and informal post-workshop discussions with kc claffy, to supplement my flakey memory. All errors of transcribing are, naturally enough, mine.

The workshop itself could not have materialised if it were not for the RIPQOS program committee - Mark Allman, kc claffy, Tristan Henderson, Geoff Huston, Derek Mcauley, Kathie Nichols, John Wroclawski, and Sebastian Zander. The ACM SIGCOMM2003 Workshop organizers (and in particular Craig Partridge) also deserve thanks for first believing in this workshop and then providing logistical support for workshop advertising, website and registrations.

And finally, the workshop would have been nothing without our presenters (listed below) and our discussion panelists at the end of the day – Jon Crowcroft, Bruce Davie, Jennifer Rexford and Ran Atkinson.

References

[1] "Proceedings of the ACM SIGCOMM workshop on Revisiting IP QoS: What have we learned, why do we care?," August 2003, http://portal.acm.org/citation.cfm?id=944592&coll=ACM&dl=ACM&CFID=13079983&CFTOKEN=66295760 (URL as of October 2003)

[2] Jon Crowcroft, Steven Hand , Richard Mortier,Timothy Roscoe, Andrew Warfield , "QoS`s Downfall: At the bottom, or not at all!" Proc. ACM SIGCOMM 2003 Workshops, p. 109, August 2003

[3] Gregory Bell, "Failure to Thrive: QoS and the Culture of Operational Networking," Proc. ACM SIGCOMM 2003 Workshops, p. 115, August 2003

[4] Carlos Macian, Lars Burgstahler, Wolfgang Payer, Sascha Junghans, Christian Hauser, Juergen Jaehnert, "Beyond Technology: The Missing Pieces for QoS Success," Proc. ACM SIGCOMM 2003 Workshops, p. 121, August 2003

[5] Bruce Davie, "Deployment Experience with Differentiated Services," Proc. ACM SIGCOMM 2003 Workshops, p. 131, August 2003

[6] Stanislav Shalunov, Benjamin Teitelbaum, "Quality of Service and Denial of Service," Proc. ACM SIGCOMM 2003 Workshops, p. 137, August 2003

[7] Tristan Henderson, Saleem Bhatti, "Networked games --- a QoS-sensitive application for QoS-insensitive users?," Proc. ACM SIGCOMM 2003 Workshops, p. 141, August 2003

[8] Ben Teitelbaum, Stanislav Shalunov, "What QoS Research Hasn`t Understood About Risk," Proc. ACM SIGCOMM 2003 Workshops, p. 148, August 2003

[9] Panos Gevros, "Internet Service Differentiation using Transport Options:the case for policy-aware congestion control," Proc. ACM SIGCOMM 2003 Workshops, p. 151, August 2003

Workshop Report: Future Directions in Network Architecture (FDNA-03)

Steven Bauer, Xiaowei Yang
{bauer,yxw}@mit.edu

Introduction

The Future Directions in Network Architecture (FDNA) Workshop, a one day workshop held in conjunction with the ACM Sigcomm 2003, provided a forum for participants to predict and consider the architectural underpinnings of future networks and the evolving Internet. The workshop was very well attended, attracting over eighty participants. A total of 48 papers were submitted to the workshop. Joint submission to the workshop and the general Sigcomm 2003 conference was permitted, and roughly half of the papers were dual submissions. Of these submissions, nine full papers and six short talks were selected for presentation. Speakers of the full papers gave 20-25 minute talks, with 5-10 minutes for questions. Short talk speakers were allotted 10 minutes for their talks with 5 minutes for questions. The presented full papers are published in the Sigcomm 2003 consolidated workshop proceedings, available from the ACM Digital Library.

Workshop Purpose

The purpose of the workshop was to convene researchers interested in the future of network architecture. The architecture of a network specifies the high level principles and structures that guide its design, especially the engineering of protocols and algorithms, and the interaction of different functional components. Advances in architecture come along two distinct fronts: 1) identification of new fundamental structuring principles and 2) new guidelines for making decisions about the functional decomposition and modularity of a system.

The current Internet architecture has been remarkably successful as the underpinning of a global, general-purpose, decentralized data communication network. Architectural decisions made 30 years ago have allowed the Internet to quickly support new applications and adapt to dramatic changes in technology. The requirements underlying the Internet architecture, however, have changed significantly since the 1970's.

New architectural requirements reflect ever growing government and commercial demands, increasingly complex security concerns, and changing user expectations and needs. Further, new classes of networks - sensor-nets, highly mobile ad-hoc nets, overlays, and others - have come into existence. These networks have very different design goals, operating requirements, and implementation environments than those imagined for traditional network architectures.

As the network community works to accommodate these new requirements, all too often the coherence of the Internet's architectural design is eroded by a patchwork of narrow technical embellishments. The result is an ever increasing complexity accompanied by a loss of functionality and extensibility. Given these pressures, revisiting the architectural principles of large general-purpose networks is appropriate and necessary.

The workshop addressed these issues in a series of four sessions. Talks included ideas on new routing architectures, new network abstractions, and new techniques and approaches for a variety of existing network problems. Each session sparked numerous interesting discussions and debates. The sessions are summarized in the following sections. We outline the talks and the discussions that followed.

Session 1: Motivations

This session addressed ways to design architectures that flexibly accommodate diverse concerns and requirements. Recognizing that various parties participate in a network for diverse reasons, this session's speakers considered how different participant's motivations can be leveraged in architectural designs to more efficiently accomplish system goals. The session introduced attempts to reason about and analyze different requirements in making architectural decisions. Finally, architectural influences of the changing nature of network technologies was considered.

Addressing Reality: An Architectural Response to Demands on the Evolving Internet

Presented by David Clark (MIT LCS)

Clark presented three key tenets that he contended should guide the evolution of the Internet in its next generation: 1) design for change 2) design for controlled transparency and trust and 3) respect the centrality of the tussle space by accepting conflicts of interest in the technology. Clark reviewed each of these tenets and then discussed their implications for future networks.

Clark explained that designing for change implies taking explicit architected action to preserve the ability to change and evolve a technology. This may imply making sacrifices in performance and efficiency. The challenge for architects is how to preserve generality and evolvability while minimizing costs.

Designing for controlled transparency is a tenet born from the recognition that users often have different requirements for the network. Sometimes the network should appear completely transparent and pass all packets but other times the network should mediate some communications. Clark postulated that the requirements often may be dictated by how much users trust each other. Users want the network itself to prevent certain traffic from reaching them if they don't trust the source. The key to this design principle is that the transparency of the network is controllable by the end user.

The third tenet was respecting the centrality of the "tussle space" in designing architectures. This tenet prescribes that architects should not attempt to resolve inevitable conflicts of interest that arise in a technology, but should instead architect to allow the conflicts to occur naturally within the coherently designed structure.

Clark explored implications of these design tenets in detail in the rest of the talk. The examples he employed came from a reexamination of the Internet and consideration of future requirements such as mobility, non-general purpose networks such as sensor nets, and evolving security needs.

Clark considered the issue of packet-switched versus circuit-switched networks and concluded that neither model is ideal. He contended that the fine-grained multiplexing achievable with packets has passed the test of time, however, we are missing an architecture for aggregates. This missing piece often triggers the erroneous call to replace packets with circuit abstractions. The lack of aggregates has lead to naming and managing them using lower level mechanisms such as MPLS. Aggregates instead should be first class objects in the network architecture.

Clark argued for a reconsideration of the "stateless faith" of Internet architects. He argued that we must accept that non-general networks will be attached to the edge of a general Internet core in the future. We must architect for this, likely necessitating application level state in the network. Clark argued that we need to "design the future" not "drift from the present."

The discussion after the talk initially focused on how architectural evolution occurs. Clark observed that incentives often control this process but technologies can be positioned appropriately by observing the interests of key players. Clark emphasized that we must only standardize the things seen on the wire and we can not and should not dictate how applications are built. He stated thought that it is crucial that network architects provide better guidance to application designers than has been done in the past.

The next series of questions probed at why Clark had limited his focus to the type of networks that he had. He was asked why multicast was not discussed. Clark indicated that the research group did not reach any consensus on multicast.

Other audience members questioned how Clark saw more exotic networks like Delay Tolerant Networks (DTNs). One person, for instance, wanted to know why Clark had assumed that the sender and receiver were static entities. A different network model could allow the networks to actively find appropriate or available recipients that were unknown or maybe didn't even exist when the packet was sent. Clark questioned the level of the networking stack that DTN style problems actually manifest themselves and suggested that the problems posed by DTNs might better be addressed at higher levels of the stack.

A Real Options Framework Illustrating the Economic Value of the End-2-End Argument

Presented by Mark Gaynor (Boston University)

Gaynor's talk presented a real options framework for understanding the economic value of competing system designs. He applied his techniques to network architectures, since, he noted, architecting networks in the past has relied more on "art" than "science". Gaynor is attempting to change this by providing a framework in which architectural questions can be reasoned about formally. He stated that he is seeking to model the long term impact of architectural design decisions.

He conducted his analysis using an extensions of options called real options, which he described briefly. Options are an economic construct that models uncertainty, flexibility and choice. Real options extends the theory of options to non-financial assets. Gaynor has employed real

options to evaluate the end-to-end architecture of the Internet and compare the value of competing technologies such as packet-switched to circuit-switched networks, of SIP compared to Megaco for VoIP, and WiFi compared to cellular networks.

Assuming that centralized and non-modular systems are ”cheaper” than distributed modular ones, Gaynor accessed the relative worth of systems in a given market context. He demonstrated quantitatively that the experimentation allowed by modular system is worthwhile when market uncertainty is sufficiently high. The intuition behind his results is that the benefits of a modular system outweigh the costs during periods when the market requirements of a system are not sufficiently well understood.

The discussion first focused on the models he employed to justify his conclusions. Gaynor acknowledged that putting real numbers into his models is difficult, but argued that his work provided a framework for a structured debate to occur. A questioner asked if he employed the Black-Scholes model of valuing options. Gaynor indicated that he did not, instead using a binominal model.

The discussion then centered on whether flexibility was actually good for providers, since it often seemingly empowers consumers. Gaynor’s response was to argue that the flexibility was indeed in the long term interest of providers. Even in the short term there are market opportunities if the flexibility exists. I-mode services in Japan were discussed as an example of modularity and flexibility being beneficial for providers.

Peer-to-Peer Network Architectures: The Next Step (Harnessing the Symbiosis of Altruism and Selfishness)

Presented by Peter Triantafillou (University of Patras)

In this talk, Triantafillou argued that too often peer-to-peer networks are viewed as networks of homogeneous nodes. However, the computational, bandwidth, and storage resources available to nodes often varies considerably. The heterogeneous nodes are distinguished by their behaviors; this is the key distinction between Triantafillou’s work and others. Some nodes are ”selfish” while other nodes are ”altruistic” and work for the benefit of the larger network. Whether or not this is true ”altruism” or nodes are in actuality deriving some benefit from their behavior is inconsequential. The point is that differences in behavior exist and can be exploited to the benefit of the network.

Triantafillou’s premise is that one can identify altruistic peers and leverage them to improve network functions. He suggested, for instance, improving routing lookups and routing functions by concentrating them at the altruistic nodes. He also suggested that one can isolate nodes behaving selfishly if the behavioral differences can be effectively identified.

While he did not have time in the short talk to provide the algorithmic or protocol details of the architecture, Triantafillou argued the general case for his design. His larger point was that the inherent structure and behavior of network nodes were important to how future peer-to-peer architectures are built.

After the talk, the discussion focused on the need to carefully balance exploiting good behavior with the potential that such exploitation could have the negative impact of discouraging desired behavior. An alternative hypothesis could be that one should architect the network to encourage the desired behavior. Triantafillou restated his position that behavioral differences exist for a variety of reasons and could be actively leveraged to improve a network.

Beyond Hosts and Routers: Some Architectural Principles for Future Mobile Networks

Presented by Robert Hancock (Siemens/Roke Manor Research, U.K.)

Hancock contended that current network architectures exhibit very high rigidity; they have fixed function modularity and asymmetric interfaces. In the future, he expects arbitrarily complex user scenarios where the ”host vs. infrastructure” division is no longer appropriate. Instead multiple nested/joined groups of cooperating devices will exist. He argued that the rigid approach does not scale to accommodate the requirements that mobile and multihomed nodes will have.

For these reasons, Hancock argued that multihoming and network composition are important organizing principles for future mobile networks. However, there are many challenging technical issues remaining in addressing these areas. The fundamental challenge in multihomed hosts is the need to separate identity and addresses. Nodes also need to negotiate for services to satisfy their diverse set of requirements.

To address these future problems, Hancock argued that control plane protocols will need an architectural framework of their own. Further the requirement for the architecture to support multihoming and composition will both blur the host/router distinction and spur a need for networks, rather than just nodes, to be nameable first class objects. Such advances will also serve to encourage more universal bi-directional interfaces.

The questions during the discussion time focused on clarifying Hancock’s position. One questioner wanted to

know if there was a fundamental new challenge as he felt that systems already successfully employed multiple radios for a variety of tasks. Hancock responded that the new challenges were 1) the methods to sew together identities and transport and 2) mechanisms to discover and negotiate services available in other networks. He stated that it is really difficult to get the interfaces right.

The discussion then turned to the properties that such networks would have. One participant noted that resilience to failure is an interesting property of such future networks; Hancock agreed.

Reconsidering the Wireless LAN Platform with Multiple Radios

Presented by Victor Bahl (Microsoft Research)

This talk was a demonstration of how changing technologies are influencing future networking architectures. The evolving technology in this case was radios for mobile or wireless devices. Radio technology is becoming increasingly cheap with predictions for even more price drops in the future. Some radios can be bought for under $5.00 apiece according to Bahl.

This evolution has lead Bahl's research group to consider how to make wireless networks more robust by exploiting multiple radios on a device. Most often this is multiple different radios that have different performance characteristics for different operations. Bahl noted that many people would claim that the real future lay in software radios. He preemptively countered that software radios are not a viable option even a number of years out due to the significant cost of the hardware required.

Bahl demonstrated a number of different ways that multiple radios can be used to significantly improve performance metrics such as power consumption. He addressed the question of how one could take a systems level approach to determine how to use multiple radios.

A variety of specific technical questions were asked of Bahl after his talk. One participant wondered whether Bahl was considering multiple and different radios only or were their applications for multiple and not different radios? Bahl responded that they were most often leveraging the differences between types of radios to accomplish their goals. Questioning where Bahl had placed the virtualization layer for the radios, one participant complained that he would lose valuable information that he would need to make certain decisions. Bahl indicated that the questioner could simply put his logic at the virtualization layer instead of above it.

Session 2: New Abstractions

Conventional networking relies upon a well-understood set of abstractions. These abstractions simplify the reasoning about network architectures and provide a common understanding and way of discussing networks. When the networking community considers new functionality, we often rely upon our conventional abstract models. During this session presenters challenged this conventional wisdom by presenting new abstractions for organizing network concepts. These abstractions potentially allow us to conceive of different ways of organizing networks, thus enabling new functionality and better prospects for evolving the architecture in the future.

Plutarch: An Argument for Network Pluralism

Presented by Andrew Warfield (University of Cambridge)

Warfield explained that IP's philosophy is to enable inter-networking by homogenizing the network and transport layers. By standardizing the middle, layers above and below are free to evolve. The fast growth of IP has demonstrated the power of this hourglass-shaped architecture. Warfield then proceeded to challenge this underlying philosophy. He listed problems associated with the current IP architecture, including, functional deficiencies (e.g., lack of support for mobility), scaling issues, the lack of freedom to innovate, and the ever increasing heterogeneity of networks with different capabilities, e.g., sensor networks, ad hoc networks, and wireless networks, he argued that it is less and less obvious that a ubiquitous transport and name resolution protocol is the right solution for the future.

In contrast, Plutarch emphasizes less homogeneity, and states that an inter-networking architecture must allow communications between dissimilar networks without mandating a standardized data path. There are two fundamental concepts in Plutarch: Contexts and Interstitial Functions (IFs). A Context is an area of the network that is homogeneous to some extent. It serves two purposes. First, contexts serves as descriptors for composing end-to-end services; Second, contexts describe communication mechanisms with which end points might use for session setups. Interstitial Functions exist at the borders between contexts. The primary function of IFs is to allow data with different naming and addressing schemes to cross contexts. The contemporary realizations of IFs include NAT boxes.

Warfield gave an example to illustrate the concept of

Plutarch. In the example, a user attempted to connect from a GPRS laptop to a sensor net via the Internet. In Plutarch, three stages happen before communication. First, a distributed and decentralized search service, not DNS, is used to resolve name=value pairs for addresses. Second, as the query results may include chained contexts, two IFs, one at the border of the sensor net and the Internet, and the other at the border of the Internet and the GPRS network, would be instantiated and installed. Finally, applications bind to the newly created chained context. Communication can then be established.

Warfield described several future directions for his research, including scalable name lookup services, correct semantics of IFs, fault tolerance, and garbage collection.

During the discussion time, concerns were raised about the degree of complexity in Plutarch since Plutarch requires complicated IFs to translate between contexts. The solution to avoid the complexity seems to be to standardize data representation.

The next comment came from a audience member who shared his experience in the design process of Application Level Framing. He stated that the idea of having states in the middle of the net and allowing asynchronous data transfer starts to break down when mechanisms must be added to handle failures. For example, what happens if an intermediate buffer is filled up? The complexity gradually builds up. Finally, he stated that he gave up the idea and decided to let applications handle end-to-end state.

Another participant observed that there is a commonality between DTNs and Plutarch. Warfield agreed but noted that the Plutarch approach is to make the IFs generic and not architect the system to handle disconnected networks. Finally, in response to a security question, Warfield noted that security remained an interesting, but largely unexplored issue and was something they hoped to explore more thoroughly in the future.

Designing for Scale and Differentiation

Presented by Karen Sollins (MIT LCS)

In her talk, Sollins formalized the idea of grouping and subdivision using an architectural abstraction called a ”region”. The region abstraction is being explored as a general organizing principle that can be leveraged to improve network functionality. The need for this new organizing principle arises because the network is becoming increasingly heterogeneous.

Sollins explained a region is defined by a set of characteristics or invariants, for example addresses or AS numbers, but perhaps much more complicated groupings. Regions come into existence in many ways. An entity acquires a region membership through explicit insertion. Insertion implies applying invariants to the entity. Insertion into a region may fail if the invariants on the entity do not hold. This does not imply human involvement and may be automated. Since a region has a logical boundary, when an entity cross a region boundary, states may be changed, and interested parties may be notified.

Instantiations of regions include Autonomous Systems, DNS, and security regions defined by firewalls. Sollins and her students have been working on a variety of region-related projects, including garbage collection in regions, inter-region information exchange, region adaptation, and security. Several important lessons have been learned from these projects. First, manageable exchange of information about different abstractions of region invariants can have significant performance benefit. Second, understanding where tradeoffs are necessary is critical. Third, much more work is needed on questions of region creation and membership.

During the discussion Sollins was asked why people have recently started to think philosophically about large scale heterogeneous problems after the community has been silent for so long on this topic. Sollins posited that it was a political problem; the community needed to get the basic things done first. Before, we focused on TCP performance and multicast, now Sollins claimed is a time to step back and think about what we are trying to do in a much more general way.

The next questioner asked how communication between regions would occur. Was the right idea is to pick the least common denominator at the translation points or to have complicated translation points. Sollins answered that in reality, both approaches would exist. Perhaps the best that could be done was to architect for the two approaches to coexist and make them as efficient as possible. Sollins further elaborated that the two approaches were different and could not be compared directly.

A questioner challenged Sollins whether we really want all these ”sandboxes” (regions), or would common standards be better. Sollins responded that the question might be answered by economic arguments. She did not think we can predict what will be the shared functionality. What we want to build are systems where we can have architected heterogeneity.

Building an Internet Control Plane

Presented by Tim Gibson (US Army/DARPA)

Gibson addressed the need for an Internet control plane. He argued that the original design assumptions are not true for today’s Internet. He argued that we should not

treat the network as a black box any more.

Gibson's solution is to have a control plane in the network that end hosts can employ to configure network resources. His control plane has three building blocks. First, the infrastructure provides performance information and end hosts can query the infrastructure for the information. Second, end hosts are authoritatively identified. Third, end hosts pass instructions to infrastructure for their traffic, such as requesting alternate network paths. Gibson emphasized that the control plane is not a knowledge plane as it does not assume applications that rely upon cognitive or AI techniques.

The discussion centered on questions of scale and functionality required from such a control plane. Gibson was asked whether he has considered scale issues; if end hosts all send queries, will the system scale? Gibson said the research is a work-in-progress and he is more focused on providing services to end hosts and has not given much thought to scalability issues.

One audience member observed that ultimately what we learn from the data plane is often better than what we can get from a control plane. So maybe we do not actually need a control plane at all in the conventional sense of the word.

Finally, a comment was made on the spectrum of service semantics covered by the control plane. The audience member noted that there are some things we can easily know and some things we simply cannot know. If the control plan has to provide an absolute guarantee on the answers it provides, then the protocol would likely become overly complex.

A Virtual Internet Architecture

Presented by Joe Touch (UCS ISI)

Touch presented a short talk on a virtual Internet architecture (VIA). A virtual Internet is a network composed of virtual hosts (VHs), virtual routers (VRs), and virtual links (VLs). VHs and VRs are connected by encapsulated tunnels (VLs). It provides at least the same services as the Internet Architecture but in a virtual context. Touch emphasized that this is a first principle extension to the existing Internet, and not just a patch, or an interim solution. A virtual Internet facilitates the incremental deployment of new services. As an example, Touch argued if we are going to build a different kind of forwarding scheme, without a Virtual Internet, we have to rebuild everything to support the forwarding.

A VIA emulates the Internet, yet decouples services from their base networks. It supports recursion, i.e., some of VRs are VS networks themselves. A combination of BGP and ARP, called "BARP", enables recursion. It also supports revisitation, i.e., a node can be in the same overlay more than one time. Currently, running code on FreeBSD, Linux, and Cisco exists to demonstrate the concept of VIA.

During the discussion time, Touch was questioned as to why revisitation is needed. His response was that that it is needed for the same reason we need multiple processes. This point was debated further in more detailed discussion. Another member asked Touch how resource partitioning was accomplished in a VIA. Touch said that a node would first ask the operating system and then ask the network for resources.

Session 3: Routing

In this session, two papers presented new routing architectures. These new routing architectures potentially support functionality that cannot be achieved in the current Internet architecture. The other talk of the session presented work ongoing in the development of a logic for global routing that would facilitate reasoning about protocols and policies.

BANANAS: An Evolutionary Framework for Explicit and Multipath Routing in the Internet

Presented by Shivkumar Kalyanaraman (Rensselaer Polytechnic Institute)

Kalyanaraman opened the routing session with a talk on "BANANAS", an evolutionary framework for explicit and multipath routing. BANANAS is not an acronym, but an analogy, adapted from the comedy "Herbie goes Bananas". Kalyanaraman observed that Internet paths have much multiplicity, and proposed to use a single mechanism to exploit various forms of multiplicity (such as intra-domain multiplicity and inter-domain multiplicity). Although a lot of work has been done in multipath routing, BANANAS addressed the question of whether we can do multipath and explicit routing without signaling, without variable and large per-packet overhead, being backward compatible with OSPF and BGP, and allowing incremental network upgrades.

The key idea of BANANAS is to compute global path identifiers from well-known global variables such as routers' IP addresses and Autonomous System Numbers in order to avoid explicit signaling for path setup. Kalyanaraman explained that ATM and MPLS encode paths using local labels, thus requiring a signaling protocol to map global identifiers to local labels. The canonical method

to avoid the signaling cost in the BANANAS approach is to compute MD5 hashing of the sequence of node identifiers followed by a CRC-32 checksum to get a 32-bit hash value as the global path identifier. Kalyanaraman showed that simple extensions can be added to OSPF and BGP to implement BANANAS. He also showed examples for partial deployment.

The discussion after the talk focused on the scalability of the framework. It was observed that BGP only maintains a single path but is already very complicated, and a router already has too many routing entries. Kalyanaraman was asked how BANANAS scaled and is their per-flow state in the routers? He responded that the framework provides tools and mechanisms to control how much state a router maintains.

Towards a Logic for Wide-Area Internet Routing

Presented by Nick Feamster (MIT LCS)

In this talk, Feamster argued that a framework for reasoning about wide-area Internet routing protocols is required. Protocol designers and network operators need a way to describe and reason about protocol behavior. He described his framework as a routing logic but emphasized that his focus is on practical improvements for routing protocols.

According to Feamster, a routing logic includes two components: properties and rules. First, high-level properties describe the behavior of a routing protocol. Second, these properties are defined in terms of rules (i.e., sufficient conditions) that are easier to reason about. Feamster demonstrated the practical use of a routing logic using BGP, a deceptively simple protocol. Feamster listed the five key properties BGP should have: validity, it advertises valid routes; visibility, every valid path has a corresponding route; safety, it will converge to a unique stable answer given a set of choices; determinism, the converged state is not affected by the order of messages or the set of available routes; and information flow control, it should not expose more information than intended. Feamster showed examples of rules that define the validity property and the information flow control property.

Feamster briefly addressed open problems at the end of his talk. First, protocol timing related issues are not covered in the logic yet. Second, configuration validation is sometimes about verifying intent of an operator (e.g. aggregation), which makes some things difficult to address with a routing logic. Third, more work is required to figure out how to apply the routing logic framework to routing protocols other than BGP.

The discussion session focused initially on the capabilities of the routing logic. It was observed that the related work is all about reasoning about negative properties. Similarly the audience member asked, it seems like this may be only able to prove that things are going to go bad; can one prove that things are going to go right? Feamster said that constructive results are shown in the paper. However, he cannot promise that it is possible to say the configuration is guaranteed correct.

Feamster next was asked whether the logic is accessible for a regular network operator. Feamster said they are trying to move towards that direction. Ideally, one can take a configuration and ask if it is going to go on a router without causing a problem using the routing logic.

Feamster was then asked to contrast and compare his work on papers presented the day before (i.e., one on a BGP algebra, and one on BGP policy languages). Feamster said both papers focused on convergence aspects, but his work does not touch on convergence at all. He explained his is more about a formal reasoning framework and protocol design.

NIRA: A New Internet Routing Architecture

Presented by Xiaowei Yang (MIT LCS)

Yang presented a talk on the design of a new Internet routing architecture (NIRA). The author observed that in today's Internet, users can pick their own ISPs, but once the packets have entered the network, users have no control over the routes their packets take. The author argued that it would be a better alternative to let users pick their domain-level routes because user choice fosters competition. The author used the telephone network as an example. In the telephone system, users are able to choose long distance providers separate from their local providers. This user choice has created a competitive market for long distance providers and has significantly reduced the cost for long distance phone calls. The author hypothesized that the stagnation in introducing new services, e.g. QoS, in today's Internet was a signal of the lack of competition in the form of user selected routes.

The design of NIRA focuses on the domain-level choices because choices at this level encourages ISP competition. It leaves router-level choices unspecified. The design is optimized for the special structure of the Internet. In the Internet, business relationships determines transit policies between domains. Common transit policies state that providers would provide transit service for customers, but not vice versa; peers only provider transit service for each other's customers. According to these policies, domain-level routes are said to be "valley-free". There exists a densely connected "core" of the Internet.

Based on this structure, the author proposed to use

a fixed length and provider-rooted hierarchical addressing scheme to optimize route representation. Top-level providers obtain unique address prefixes and recursively allocate subdivisions of the addresses to their customers. As a result, the route segment from a domain to a top-level provider can be uniquely identified by an address prefix. Two addresses can be used to represent a valley-free route. Source routing header is used for representing non-valley-free routes. The assumption is that valley-free route is the common case so that this route representation scheme is efficient.

The design of NIRA separates the task of route discovery into two halves. Each user only needs to know his part of the network as it grows. The infrastructure service, Topology Information Propagation Protocol (TIPP), tells users topology information of his providers, and the corresponding address allocation information. TIPP is a policy-based, link-state like protocol and is less likely to suffer from the slow convergence problem of BGP.

The user looks up the destination's addresses, and optionally the topology information of the providers of the destination , using another infrastructure service, Name-to-Route Resolution Service (NRRS). Similar to DNS, NRRS uses a hierarchical namespace and hard-coded addresses for bootstrapping. However, because addresses in NIRA are topology-sensitive, a fundamental tradeoff is that topology changes would cause address changes. The author argued that only static topology changes, such as providers changing peering agreements, would cause address changes. Recent Internet measurement results shows that the static Internet topology changes at a relative low rate, and therefore NRRS server updates are likely to be manageable.

Route failure handling in NIRA is a combination of proactive notification and reactive discovery. TIPP proactively notifies a user of route failures regarding to his part of the network. When a packet traverses across the destination part of the network, a router finds the route specified in a packet header is unavailable, the router will try its best to send a control message to the sender of the packet, notifying him of the route failure. In cases such router feedbacks are unavailable, users shall always use timeout to discover route failures.

The author also briefly discussed provider compensation problem. Similar to today's Internet, NIRA's provider compensation model is based on contractual agreements between providers and customers. As users can specify arbitrary routes in their packet headers, providers shall install policy filters to prevent illegitimate route usage. In cases where business relationships exist only in directly connected entities, policy filter checking for valley-free routes becomes verifying that the source address in a packet header matches the interface the packet comes from. For more complicated routes, policy filter checking may require matching source routing header against policy filters. For indirect business relationships, policy checking becomes verifying the identify of the sender of a packet at forwarding time. The author pointed out that it was an open but general problem. Overlay providers, or second-hop providers that want to sell QoS to end customers face the same problem. Any solution to this general problem can be plugged into NIRA.

Yang was asked during the discussion why projects such as NIMROD (mentioned in her related work) had failed. Yang disagreed that these routing architectures were a failure. She commented that as a research project, NIMROD has influenced the design of a number of protocols. She noted that research is much like fashion design; people take ideas from research projects and made on-street versions.

Yang was questioned next about the motivation for providers to implement NIRA. Providers lose control over routes, and the market would become more competitive for them. Yang explained that providers that are already in monopoly position probably do not want competition. But for smaller providers, if they want to enter the market, they probably would welcome user choices. Yang explained that employing user choices to discipline the ISP market is a hypothesis; only real experiment could justify it. She believed it was a worthy experiment and this work is the first step towards it.

The final question dealt with whether NIRA would work well in situations where peering agreements become more dynamic. Yang said she expected the peering agreements would change at a manageable time-scale in the real Internet.

Session 4: New Techniques and Approaches

The final session of the day contained a set of papers presenting new techniques and approaches for addressing various networking problems. The presenters considered new principles for organizing addressing and the ensuing implications for the Internet. An argument was presented for why denial of service attacks should be addressed at the TCP/IP level of the end users network stack. Finally, a case was made for radically changing the service model of the Internet to expose network storage primitives to enable programmable networking.

FARA: Reorganizing the Addressing Architecture

Presented by Robert Braden (USC ISI)

Braden's talk presented a new addressing architecture for networks. It was a product of the NewArch Project that explored whether abstract architectural reasoning can help in creating a better technical design for the Internet to meet today's and future requirements. This talk presented one exercise in exploring whether or not that premise was actually true. Braden emphasized that this is not a "rerun" of the design effort of the original Internet. The original design effort was largely bottom up, finding one approach that met the apparent requirements guided by some abstract thinking about protocol modularity. In contrast, this research was a top down approach.

The authors went through a three stage process. First, they defined an abstract architectural model for addressing that encompassed an interesting part of the design space but left many of the details unconstrained. They then defined an architecture that instantiates this addressing model, and finally they built a prototype system.

The abstract model for addressing they developed rested on the core principle of cleanly decoupling end-system identity from network layer forwarding functions. The motivation for this split was the traditional location identity split. The implications of this in their work were the need for a remodularization of function into entities and associations. Braden explained each of the concepts in detail in the talk.

To demonstrate the consistency of the addressing model they developed, Braden then presented an instance of it, M-FARA. This represented one point in the spectrum of choices laid out by their addressing model. He also discussed the relationship between their addressing architecture and the IPv4 architecture.

During the discussion that followed people primarily challenging the model that Braden had described of entities and associations. One person for instance pushed for the need for distributed entities (anycast). Braden responded that this was the wrong model if entities were that distributed. Another person questioned whether Braden thought it would be important to be able to change entities that were involved in an association. Braden responded that this would be a fundamentally different association.

The Case for TCP Puzzles

Presented by Wu-chang Feng (Oregon Graduate Institute)

Distributed denial-of-service (DDos) attacks via worms and viruses continually disrupt the Internet. Proposals for how to deal with such attacks abound. One mitigation approach is to require that senders do a small amount of work for each network operation that they initiate. This work often involves solving some puzzle that is designed to take a small amount of time. Malicious sources that initiate large amounts of attack traffic would be slowed while "regular" senders would be only minimally delayed.

Feng argued during his talk that puzzles must be placed within the TCP/IP protocol stack layer in order to provide protection. The key reason for this is that denial-of-service activity can happen at any layer and only needs to break one link in the end-to-end chain in order to be successful. He argued that there is a corollary to the end-to-end principles that one must put security functions in a common waistline layer if the security property is otherwise destroyed unless implemented universally across a higher and or lower layer.

Feng discussed implementing the network layer puzzles in the form of a "push-back" firewall that required senders to do some amount of work before allowing a connection through the firewall to be opened. The difficulty of the puzzles presented to senders is dynamically adjustable. Feng discussed the research challenges presented by his approach. These included how to make the puzzle system itself resistant to denial of service attacks, how to ensure that the mechanism is tamper resistant to replay and spoofing attacks, and how to design control algorithms and ensure that hosts with different computational power are fairly treated.

During the discussion Feng was asked to clarify various points about his system. It was noted that he must be assuming a streaming model where puzzles were valid for some number of packets. Feng agreed and proceeded to describe his various approaches for where puzzles were issued and how long the results were valid once a puzzle was solved.

There were a variety of questions about the need to do the puzzle solving at the IP layer. One participant asserted that the right layer for the puzzles was the layer that the attack was occurring. Feng elaborated on why he felt it was only appropriate to implement the puzzles at the IP layer. He argued we did not need multiple puzzle mechanisms at each layer of the network stack.

An End-to-End Approach to Global Scalable Programmable Networking

Presented by Micah Beck (University of Tennessee)

Beck's talk was a presentation on a new architecture for a global network that exposes storage and computation primitives. The goal of the new architecture is to support globally scalable programmable networking by exposing network primitives that are generically useful to advanced

applications.

Beck appealed to the end-to-end principles in designing the network compute services. He emphasized that scalability can only be achieved by adhering to the principles that have guided network architecture design. His appeal rested on the premise that the arguments dictate not where functionality should be placed but rather the scalable nature of that functionality.

He claimed that the most important consequence of requiring scalability is that the semantics of the services offered in the network must be simple and weak. If the semantics are too complex, the services will fail the requirement that services at intermediate nodes in the network be generic. If the services are too strong then they will not compose with a scalable network without breaking.

Building on this philosophy his research group has built Logistical Networking (LoN), which exposes network storage primitives. In his talk, Beck reviewed the LoN research and discussed how to extended it to supply an abstraction of the execution layer resources. He presented an abstraction for time-sliced operation system services and discussed how they could be employed to create network services.

Beck was questioned on how his system would change the legal status of network operators. Network operators are now considered common carriers because they do not actually store or manipulate the data they transport. If they actually stored and operated on bits, then they would potentially expose themselves to legal liability. Beck responded by citing a number of cases that he claimed established precedence that the storage of bits did not create any legal liability for network operators.

Beck was then questioned about the relationship between his work and active networking. He emphasized that while the goals of active networking are similar, his approach is dramatically different. Activate networking, he claimed, does not fundamentally concern itself with scalability, while scalability is his primary concern.

Acknowledgements

The workshop organizers are tremendously grateful to the scribes, Steve Bauer and Xiaoewi Yang, for their critical role in capturing and reporting on a full day of rapid, nonstop presentations and discussion.

The workshop itself could not have materialized without the (unanticipatedly large) efforts of the program committee - Andrew Campbell, Ted Faber, Mark Handley, John Heidemann, Larry Peterson, James Sterbenz, and John Wroclawski. Thanks go to the ACM SIGCOMM2003 Workshop organizers (and in particular Craig Partridge) for providing advice, assistance and logistical support for workshop advertising, website and registrations.

Finally, and most importantly, we thank the presenters and participants in the workshop for their interesting and thoughtful discussions. The organizers are most grateful for their time, interest, and thoughtful energy.

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:

Mark Crovella
Department of Computer Science
Boston University
111 Cummings St.
Boston, MA 02215
Voice: +1 617 353 8923
Fax: +1 617 353 6457
Email: crovella@cs.bu.edu