Computer Communication Review

Special Interest Group on Data Communication

An ACM SIGCOMM Publication

Volume 35, Number 2
April 2005

acm sigcomm

Editorial Philosophy

CCR (the Computer Communication Review) is published by ACM's Special Interest Group on Data Communication (SIGCOMM). The publication is the primary vehicle of communication to the members, presenting articles of practical research significance and relating experiences in the area of communications and computer networks, including technical design and engineering, regulation and operations, and the social implications of computer networking, with a special interest in the systems engineering and architectural questions of communication.

We seek to provide an accessible, flexible, and timely forum for information on both well-researched topics and those less common, perhaps leading-edge, topics which may not yet have found their way into the literature. We welcome the opportunity to publish materials that are controversial or which may as yet under development. We also publish short editorial notices, as well as readers' comments and reactions. Please visit

http://www.acm.org/sigs/sigcomm/ccr/ for details.

Submissions

One electronic pdf copy of the manuscript that follows the CCR-SIGCOMM templates

(http://www.acm.org/sigs/pubs/proceed/template.html), should be submitted to the CCR Editor:

Christophe Diot

INTEL Research

ccr-editor@intel-research.net

+44-1223-763-444 (voice)

Submissions that do not follow the templates and formatting instructions below will not even be considered for review.

Manuscript Format

All manuscripts should follow the Word or LaTex templates and formatting instructions available at:
http://www.acm.org/sigs/sigcomm/ccr/CCR-author-info.html

Be sure that your submission includes:

1. Author’s name(s) and complete mailing address, phone number, fax number, Internet e-mail address, and a short (no more than 100 words) biographical sketch on a separate page
2. Abstract (200-300 words)
3. ACM Classification (Categories & Subject Descriptors): ACM categories from the schedule used by the Communications of the ACM. See http://www.acm.org/class
4. ACM General Terms
5. Personal keywords or phrases for indexing (up to 15)

Citations

Bibliographic references in the body of the text should be formatted as follows: (author name[s], year). Entries in the References should be ordered alphabetically according to authors' or editors' names. Journal references should take the form: Author last name, initials. (Year). “Title,” Journal, Vol., No., pages. Book and report references should take the form: Author last name, initials. (Year). Title, publisher city: publisher name, pages, if applicable. Conference proceedings should take the form: Author last name, initials. (Year). “Title,” Proceedings name, conference location, pages.

The Review Process

Articles are handled by an area editor and then assigned to reviewers. When the reviews are completed, the manuscript will be accepted or rejected. Revisions will be requested under the supervision of the CCR area editor. The review process takes at most 3 months. Authors will be informed of the status of their manuscripts as they pass through the various stages.

Editor’s Message

CCR is entering a new era. In order to make it even more obvious, we have decided to change the cover. Not much though; just a color inversion so that our readers notice something has changed and start reading it with increased interest :-)

The new CCR will build on changes successfully begun by John Wroclawski. The goal is for CCR to strengthen its role as the network community newsletter in which we are all involved and contribute. CCR wants to be responsive to research topics and to readers opinions. Therefore, my first objective will be to implement a short turn-around: We will have four implicit deadlines (every quarter — see CCR online for details) and papers submitted before a given deadline will be published or rejected by the next deadline.

On the content side, CCR will keep publishing papers in the general field of data communications characterized primarily by quality and innovation of the work and timeliness of publication. CCR will encourage publication of “works in progress,” provided they are of good caliber and introduce new material, whether it is a new methodology, a new idea, or a new result. Experience papers, system papers, and papers reproducing others’ results, will find a home in CCR, as long as the quality is high enough and useful to the community. Each paper will be published with a public review that will synthesize the reviewers’ opinions and explain our motivation for publishing the paper. In addition, CCR does not retain copyright. Therefore, authors are welcome to publish more advanced or complete version of their work in journals and conferences after publication in CCR.

But this is not the only type of contributions that we will publish! Each CCR issue will have a news-like section which will publish short editorial notes. The scope for such notes will span topics such as interviews of research scientists and industry leaders, editorials, conference reports, summary standards, NSF or European Commission user manuals, arguments on a research topic, outrageous opinion column, favorite hits, What's Worth Reading, and so on. We even intend to publish reader's comments on articles and editorial notes, or on any other topic of interest to CCR’s readers.

Everyone is welcome to contribute and we hope to receive input from faculty members, junior researchers, students, industry researchers, etc.We also have a new editorial board. The role of the area editors is not only to harrass reviewers to get timely decision. It is also to contribute editorial notes and to help make a timely, exciting, hot magazine. Let me introduce you the new editorial board; it tries to mix different areas of expertize, backgrounds, cultures, etc. It is slightly biaised (I let you find the bias :-). But i believe this bias will just add more discussions and arguments to the selection process :-)

Chadi Barakat, *INRIA Sophia Antipolis, France*
Ernst Biersack, *Eurecom, Sophia Antipolis, France*
Mark Crovella, *Boston University, USA*
Jon Crowcroft, *University of Cambridge, UK*
Constantinos Dovrolis, *GaTech, Atlanta, USA*
Serge Fdida, *Universite Pierre et Marie Curie, Paris, France*
Steve Gribble, *University of Washington at Seattle, USA*
Roch Guerin, *University of Pennsylvania, Philadelphia, USA*
Dina Katabi, *MIT, Cambridge, USA*
Srinisivan Keshav, *University of Waterloo, Canada*
Venkat Padmanabhan, *Microsoft Research, Redmond, USA*
Matt Roughan, *University of Adelaide, Australia*

In addition to the editorial team, please welcome Angela Baretto from Intel Research and Lisa Tolles from Sheridan Printing. Angela is helping me manage CCR submissions and make sure deadlines are met. Lisa assembles each issues and manages the production phase (putting the issue together, printing, and mailing). Both Angela and Lisa are critical to the success of CCR.

Let’s talk about the April issue now. You might not find much changes in this issue. It is mostly because (1) it is a transition issue and in order to implement first the short turn-around, we decided to postpone some of the changes to July, and (2) changes comes from the readers and the community, and we can not make changes if you do not send us exciting articles, editorial notes and mails to change CCR the way we want to change it.

In the April issue, you will find six technical articles and three editorial contributions. We address a wide range of topics that we hope you will find useful. You can also comment on these articles in the next CCR issues. This issue also contains a reprint of the pioneering paper: “A Protocol for Packet Network Intercommunication” (originally published in 1974 in IEEE Transactions on Communications), in honor of Vint Cerf and Bob Kahn being selected as this year’s winner of ACM’s prestigious Turing Award.

To conclude, we ask all networking researchers (and SIGCOMM members in particular) to take to heart that CCR is your magzine and its content is made by you! Please send your editorial notes, articles, emails, etc. We will review them and strive our best to publish them as soon as we can.

Christophe Diot
CCR Editor
Intel Corporation

acm sigcomm

Stability of End-to-End Algorithms for Joint Routing and Rate Control

Frank Kelly
Statistical Laboratory
University of Cambridge
Cambridge, U.K.
f.p.kelly@statslab.cam.ac.uk

Thomas Voice
Statistical Laboratory
University of Cambridge
Cambridge, U.K.
t.d.voice@statslab.cam.ac.uk

ABSTRACT

Dynamic multi-path routing has the potential to improve the reliability and performance of a communication network, but carries a risk. Routing needs to respond quickly to achieve the potential benefits, but not so quickly that the network is destabilized. This paper studies how rapidly routing can respond, without compromising stability.

We present a sufficient condition for the local stability of end-to-end algorithms for joint routing and rate control. The network model considered allows an arbitrary interconnection of sources and resources, and heterogeneous propagation delays. The sufficient condition we present is decentralized: the responsiveness of each route is restricted by the round-trip time of that route alone, and not by the round-trip times of other routes. Our results suggest that stable, scalable load-sharing across paths, based on end-to-end measurements, can be achieved on the same rapid time-scale as rate control, namely the time-scale of round-trip times.

Categories and Subject Descriptors

C.2.2 [**Computer-Communication Networks**]: Network Protocols—*congestion control, routing protocols*

General Terms

Algorithms, Theory

Keywords

Internet, dynamic routing, scalable TCP

1. INTRODUCTION

Historically, the primary purpose of IP routing has been to maintain connectivity in the presence of topology changes and network failures. IP routing typically chooses the shortest path to the destination, based on simple metrics like hop count or distance. While the simplicity of this approach has made IP routing highly scalable, there has long been a desire to improve the reliability and performance of the Internet through the use of routing metrics that are more sensitive to congestion [28]. More recently there has also been increasing interest in multi-path routing, motivated by applications to ad-hoc networks [4, 11] and overlay TCP [7], and by perceived problems with current routing protocols [21, 29]. But despite the potential advantages of dynamic routing, it has in the past been difficult to deploy in packet-based networks like the Internet because of potential instability, manifested as routing oscillations [26].

In recent years theoreticians have developed a framework that allows a congestion control algorithm such as Jacobson's TCP [9] to be interpreted as a distributed mechanism solving a global optimization problem: for reviews see [12, 19, 20, 22]. The framework is based on fluid-flow models, and the form of the optimization problem makes explicit the equilibrium resource allocation policy of the algorithm, which can often be restated in terms of a fairness criterion. And the dynamics of the fluid-flow models allow the machinery of control theory to be used to study stability, and to develop rate control algorithms that scale to arbitrary capacities [14, 19, 22].

Han *et al.* [7] have used the framework to study multi-path routing in the Internet. They have presented an algorithm that can be implemented at sources to optimally split the flow between each source-destination pair, and they develop a sufficient condition for the local stability of the algorithm. The condition is decentralized in the sense that the gain parameters for the routes serving a particular source-destination pair are restricted by the round-trip times of those routes, and not by round-trip times elsewhere in the network.

In this paper we improve on this result, and present an algorithm with a sufficient condition for local stability that is decentralized in the stronger sense that the gain parameter for each route is restricted by the round-trip time of that route, but not by the round-trip times of other routes, even those other routes serving the same source-destination pair. The novel feature of our scheme is that the control exerted by a source over its available route flow rates is treated similarly to link congestion feedback. This allows us to apply established single-path techniques to our multi-path model. The condition we derive is conceptually simpler than that of [7], and is less demanding in the case where the round-trip times associated with a source-destination pair are highly heterogeneous. The sufficient condition is a generalization of Vinnicombe's [23] original condition for the single path case, and depends explicitly on the fairness criterion implemented by the algorithm. The sufficient condition constrains the speed of routing adaptation to essentially the same time-scale as is allowed for rate control.

In this paper we are modelling networks with *path diversity*, i.e. systems where at least some source-destination pairs have access to two or more different routes. At the minimum, we assume that, for these source-destination pairs, the source can send different flows addressed to the same destination over different routes, for example, through different Internet service providers, or different initial wireless links. We note that explicit support for edge routing is not currently available in the Internet, but it is enough for our purpose that by some means or other it is possible to create some path diversity.

A helpful distinction, developed by Zhu *et al.* [29], is to view routing information as separated into its structural and dynamic components, the former being concerned with the existence of links, and the latter with the quality of paths across the network. We suppose that the routes available, however discovered, are fixed on the time-scale we are considering. Our focus in this paper is the stability or otherwise of the system's response to dynamic information.

More formally, we suppose the network comprises an interconnection of a set of *sources*, S, with a set of *resources*, J. Each source $s \in S$ identifies a unique source-destination pair. Associated with each source is a collection of *routes*, each route being a set of resources. If a source s transmits along a route r, then we write $r \in s$. Likewise, if a route r uses a resource j we write $j \in r$. For a route r we let $s(r)$ be the (unique) source such that $r \in s(r)$. We let R denote the set of all routes.

We make no assumptions about whether the routes $r \in s$ are disjoint. Clearly the ability to generate resource disjoint routes will assist in the construction of highly robust end-to-end communication for the source-destination pair labelled by s, but our model also covers the case where some or all of the routes $r \in s$ share some path segments.

In our model a route r has associated with it a flow rate $x_r(t) \geq 0$, which represents a dynamic fluid approximation to the rate at which the source $s(r)$ is sending packets along route r at time t.

For each route r and resource $j \in r$, let T_{rj} denote the propagation delay from $s(r)$ to j, *i.e.* the length of time it takes for a packet to travel from source $s(r)$ to resource j along route r. Let T_{jr} denote the propagation delay from j to $s(r)$, *i.e.* the length of time it takes for congestion control feedback to reach $s(r)$ from resource j along route r. In the protocols we shall be considering, a packet must reach its destination before an acknowledgement containing congestion feedback is returned to its source. Further, we assume queueing delays are negligible. Thus for all $j \in r$, $T_{rj} + T_{jr} = T_r$, the round trip time for route r.

Use the notation $a = (b)^+_c$ to mean $a = b$ if $c > 0$ and $a = \max(0, b)$ if $c = 0$.

We are now ready to introduce our fluid-flow model of joint routing and rate control:

$$\frac{d}{dt}\, x_r(t) = \kappa_r x_r(t) \left(1 - \frac{\lambda_r(t)}{U'_{s(r)}(y_{s(r)}(t))}\right)^+_{x_r(t)} \qquad (1)$$

where

$$\lambda_r(t) = \sum_{j \in r} \mu_j(t - T_{jr}), \qquad (2)$$

$$\mu_j(t) = p_j(z_j(t)), \quad z_j(t) = \sum_{r: j \in r} x_r(t - T_{rj}) \qquad (3)$$

and

$$y_s(t) = \sum_{r \in s} x_r(t - T_r). \qquad (4)$$

Here and throughout we assume that, unless otherwise specified, j ranges over the set J, r ranges over the set R, and s ranges over the set S.

We motivate (1-4) as follows. The flow through resource j at time t, $z_j(t)$, comes from routes r that pass through resource j; and the flow that resource j sees at time t on route r left its source a time T_{rj} earlier. If we suppose that resource j adds a price $p_j(z_j)$ onto packets when the total flow through resource j is z_j, then we obtain (3). The total price accumulated by a single packet on route r, and returned to the source $s(r)$ via an acknowledgement received at time t, is given by (2). Finally (1) corresponds to a rate control algorithm for the flow on route r that comprises two components: a steady increase at rate proportional to $\kappa_r x_r(t)$; and a steady decrease at a rate depending upon both the price signals arriving back from route r, and the total rate of acknowledgements $y_{s(r)}(t)$ over all routes serving the source for route r. We shall see that the functions $U_s, s \in S$, appearing in (1) determine how resources are shared. And later, in Section 3, we shall reinterpret p_j as the drop or mark probability at resource j rather than a price.

Although $y_s(t)$ can be interpreted as the rate of acknowledgements received by source s, an alternative, and possibly more practical, implementation of (4) would be that each packet sent along a route $r \in s$ is marked with the flow rate or window size for r at time of sending. The source then computes $y_s(t)$, according to (4), from the values recorded in returning acknowledgements. Later we shall consider an alternative scheme, where each packet sent by s is marked with the total flow rate over all $r \in s$, and where, when an acknowledgement packet is returned, x_r is updated according to

$$\frac{d}{dt}\, x_r(t) = \kappa_r x_r(t) \left(1 - \frac{\lambda_r(t)}{U'_{s(r)}\left(\sum_{a \in s(r)} x_a(t - T_r)\right)}\right)^+_{x_r(t)}; \qquad (5)$$

here the sum $\sum_{a \in s(r)} x_a(t - T_r)$ is just the total flow rate recorded in a returning acknowledgement. We shall see that this alternative scheme has similar stability properties as those we prove for (1-4).

Under mild assumptions we shall establish that the system (1-4) is locally stable about an equilibrium point, provided the gain parameter κ_r on each route $r \in R$ satisfies a simple sufficient condition.

As an example of the results in Section 2, suppose that

$$U_s(y_s) = \frac{w_s y_s^{1-\alpha}}{1 - \alpha},$$

so that the resource shares obtained by different sources are weighted α-fair [18]. When $w_s = 1, s \in S$, the cases $\alpha \to 0$, $\alpha \to 1$ and $\alpha \to \infty$ correspond respectively to an allocation which achieves maximum throughput, is *proportionally fair* or is *max-min fair* [18, 22]. TCP fairness, in the case where each source has just a single route, corresponds to the choice $\alpha = 2$ with w_s the reciprocal of the square of the (single) round trip time for source s [17, 22].

Further suppose that

$$p_j(z_j) = \left(\frac{z_j}{C_j}\right)^{\beta}, \tag{6}$$

for constant C_j representing the capacity of resource j. Then the sufficient condition for local stability that we obtain is satisfied if, for each $r \in R$,

$$\kappa_r T_r(\alpha + \beta) < \frac{\pi}{2}. \tag{7}$$

Thus we have a sufficient condition for local stability that restricts the gain parameter κ_r on route r by the round-trip time T_r of route r, but not by other round-trip times, even those of other routes serving the same source. The condition has a straightforward dependence on the fairness criterion, described by the parameter α, as well as the resource responsiveness, described by the parameter β. Notably, the condition does not depend upon the size of the flow rates x or the congestion feedback λ, the number of resources on routes, the number of flows on routes or the network topology.

The organization of the paper is as follows. In Section 2 we present the formal analysis of the above model. In particular, we show that the use of delayed information described in (4) corresponds, under linearization, to the introduction of a fictitious link for each source-destination pair into the single route model, and hence leads to a straightforward sufficient condition for stability. The model analysed in Section 2 allows a fairly general form for the functions U_s, but is, in some respects, oversimplified. Section 3 considers a more specialized model that better approximates a network like the Internet, where congestion is indicated by a dropped or marked packet. We propose a routing extension of scalable TCP [14], a variant of TCP with attractive scaling properties that we show are inherited by our multi-path extension. Section 4 concludes with a brief discussion of the implications of our results for the division of routing functionality between layers of the network architecture.

2. ANALYSIS

We shall show, in Theorem 1, that an equilibrium point of the system (1-4) solves an optimization problem. We shall then discuss the global stability of the system in the case where there are no propagation delays, before proving our main result, Theorem 2, on the local stability of the system with propagation delays.

First we introduce matrices to succinctly express the relationships between sources, routes and resources. Let $A_{jr} = 1$ if $j \in r$, so that resource j lies on route r, and set $A_{jr} = 0$ otherwise. This defines a 0–1 matrix $A = (A_{jr}, j \in J, r \in R)$. Set $H_{sr} = 1$ if $r \in s$, so that route r serves source s, and set $H_{sr} = 0$ otherwise. This defines a 0–1 matrix $H = (H_{sr}, s \in S, r \in R)$.

Next we describe our regularity assumptions. Assume that the function $U_s(y_s)$, $y_s \geq 0$, is a increasing function of y_s, twice continuously differentiable, with $U_s'(y_s) \to 0$ as $y_s \uparrow \infty$ and $U_s''(y_s) > 0$ for $y_s > 0$. Thus $U_s(\cdot)$ is strictly concave. Assume that the function $p_j(z_j)$, $z_j \geq 0$, is a non-negative function of z_j, continuously differentiable with $p_j'(z_j) > 0$ for $z_j > 0$. Let $C_j(z_j)$ be defined by

$$C_j(z_j) = \int_0^{z_j} p_j(u)\, du.$$

From our assumptions on $p_j(\cdot)$, the function $C_j(\cdot)$ is strictly convex. The function $C_j(\cdot)$ will in general be parameterised by the capacity C_j of resource j, as for example if $p_j(\cdot)$ is given by the form (6).

Theorem 1. *If the vector $x = (x_r, r \in R)$ solves the optimization problem*

$$\begin{array}{ll} \textit{maximize} & \sum_{s\in S} U_s(y_s) - \sum_{j\in J} C_j(z_j) \\ \textit{where} & y = Hx \quad z = Ax \\ \textit{over} & x \geq 0, \end{array} \tag{8}$$

then x is an equilibrium point of the system (1-4).

Proof. The objective function of the optimization problem (8) is differentiable, and so it is maximized at $(x_r, r \in R)$ if and only if, for each $r \in R$,

$$x_r \geq 0, \quad U'_{s(r)}\left(\sum_{r:r\in s} x_r\right) - \sum_{j\in r} p_j\left(\sum_{a:j\in a} x_a\right) \geq 0 \tag{9}$$

and

$$x_r \cdot \left(U'_{s(r)}\left(\sum_{r:r\in s} x_r\right) - \sum_{j\in r} p_j\left(\sum_{a:j\in a} x_a\right)\right) = 0. \tag{10}$$

But condition (10) implies that the derivative (1) is zero, after substituting $x_r(t) = x_r, t \geq 0$, into (2-4). □

Remark 1. The optimization problem (8) has a long history in connection with road transport networks [1, 27], as well as communication networks [2, 6]. By our assumptions on the functions $U_s(\cdot)$ and $C_j(\cdot)$, there exists a solution to the optimization problem. At an optimum $x = (x_r, r \in R)$ is not necessarily unique, but, by the strict concavity of the functions $U_s(\cdot)$ and the strict convexity of the functions $C_j(\cdot)$, the vectors $y = Hx$ and $z = Ax$ are unique. If X is the set of optima x, then X is the intersection of an affine space with the orthant,

$$X = \{x : Hx = y, Ax = z\} \cap \{x : x \geq 0\},$$

and is compact.

Remark 2. The set X does not exhaust the equilibria of the system (1-4). For example, $x(t) = 0, t \geq 0$, is also an equilibrium point. The difficulty is that if $x_r(\bar{t}) = 0$ then $x_r(t) = 0$ for $t > \bar{t}$: the trajectory $x(t)$ can thus become trapped in the face $\{x : x_r = 0\}$. Call an equilibrium *spurious* if it is not also an optimum of the optimization problem. To understand better the issue, consider the dynamical system, evolving on $\{x : x_r \geq \epsilon, r \in R\}$, defined by

$$\frac{d}{dt} x_r(t) = \kappa_r(x(t)) \left(U'_{s(r)}(y_{s(r)}(t)) - \lambda_r(t)\right)^+_{x_r(t)-\epsilon} \tag{11}$$

with (2-4), where $T_r = 0, r \in R$, and ϵ is a small positive constant. Assume that $\kappa_r(x)$ is a positive, continuous function bounded away from zero on the set $\{x : x_r \geq \epsilon, r \in R\}$. Let X_ϵ be the set of optima to the amended optimization problem (8), with $\{x \geq 0\}$ replaced by $\{x : x_r \geq \epsilon, r \in R\}$. Following [13], rewrite the objective function of (8) as

$$\mathcal{U}(x) = \sum_{s\in S} U_s\Big(\sum_{r\in s} x_r\Big) - \sum_{j\in J} \int_0^{\sum_{r:j\in r} x_r} p_j(u)du,$$

and, for the dynamical system (11), calculate

$$\frac{d}{dt}\mathcal{U}(x(t)) = \sum_{r\in R} \frac{\partial \mathcal{U}}{\partial x_r} \cdot \frac{d}{dt}\, x_r(t)$$

$$= \sum_{r\in R} \kappa_r(x(t)) \left(\left(U'_{s(r)}(y_{s(r)}(t)) - \lambda_r(t) \right)^2 \right)^+_{x_r(t)-\epsilon} .$$

Hence outside any neighbourhood of X_ϵ

$$\frac{d}{dt}\mathcal{U}(x(t)) > 0$$

and is bounded away from zero. This is enough to ensure that any trajectory of the dynamical system (11) converges to the set X_ϵ. Thus, at least when there are no propagation delays, the amended system (11) can avoid being trapped at faces. The amendment, which prevents $x_r(t)$ dropping below a low level ϵ, can be interpreted as follows: even if a route appears too expensive, a low level of probing should take place, in case the price of the route changes. Any low, but positive, level of probing[1] is sufficient to rule out spurious equilibria, and henceforth we do not explicitly incorporate the parameter ϵ within our model (1-4).

We now turn to our main concern, the local stability of the system (1-4).

Define an equilibrium point x to be *interior* if it satisfies (9-10), and if for each route r either one or other of the inequalities (9) is strict; we thus rule out the possible degeneracy that *both* terms in the product (10) might vanish. At an interior equilibrium point x it is possible for a route r not to be used, i.e. $x_r = 0$, but there then exists a neighbourhood of x such that within this neighbourhood $x_r > 0$ implies $\dot{x}_r < 0$.

We next establish a sufficient condition for the local stability of $(y(t), z(t))$ near any given interior equilibrium point x. Let $y = Hx, z = Ax$, $U''_s = U''_s(y_s)$, $\mu_j = p_j(z_j), p'_j = p'_j(z_j), j \in J$, and $\lambda = A\mu$. Let $T_{max} = \max(T_r, r \in R)$; this parameter is needed to describe an initial condition of the system (1-4), but will not be part of the sufficient condition for local stability.

THEOREM 2. *Let x be an interior equilibrium point, and suppose that for each $r \in R$,*

$$\frac{\kappa_r T_r}{\lambda_r} \left(-U''_{s(r)} y_{s(r)} + \sum_{j\in r} z_j p'_j \right) < \frac{\pi}{2}. \tag{12}$$

Then there exists a neighbourhood $\mathcal{N}$ of x such that for any initial trajectory $(x(t), t \in (-T_{max}, 0))$ lying within the neighbourhood $\mathcal{N}$, $(y(t), z(t))$ converge exponentially as $t \to \infty$ to the unique solution (y, z) to the optimization problem (8).

PROOF. Initially assume that $x_r > 0$ for $r \in R$, and thus that

$$U'_{s(r)} \left(\sum_{r: r\in s} x_r \right) = \sum_{j\in r} p_j \left(\sum_{a: j\in a} x_a \right) \tag{13}$$

for each $r \in R$. Later we shall see that the assumption is without loss of generality.

[1]We note that probing is likely to be important for structural, as well as dynamic, aspects of routing.

Let $x_r(t) = x_r + u_r(t)$, $y_s(t) = y_s + v_s(t)$, $z_j(t) = z_j + w_j(t)$. Then, linearizing the system (1-4) about x, and using the relation (13), we obtain the equations

$$\frac{d}{dt}\, u_r(t) = -\frac{\kappa_r x_r}{\lambda_r} \left(-U''_{s(r)}\ v_{s(r)}(t) + \sum_{j\in r} p'_j w_j(t - T_{jr}) \right), \tag{14}$$

$$v_s(t) = \sum_{r: r\in s} u_r(t - T_r), \tag{15}$$

$$w_j(t) = \sum_{r: j\in r} u_r(t - T_{rj}). \tag{16}$$

Let us overload notation and write $u_r(\omega), v_s(\omega), w_j(\omega)$ for the Laplace transforms of $u_r(t), v_s(t), w_j(t)$ respectively. We may deduce from (14-16),

$$\omega u_r(\omega) = -\frac{\kappa_r x_r}{\lambda_r} \left(-U''_{s(r)}\ v_s(\omega) + \sum_{j\in r} p'_j e^{-\omega T_{jr}} w_j(\omega) \right),$$

$$v_s(\omega) = \sum_{r: r\in s} e^{-\omega T_r} u_r(\omega),$$

$$w_j(\omega) = \sum_{r: j\in r} e^{-\omega T_{rj}} u_r(\omega).$$

We calculate that

$$\begin{pmatrix} v(\omega) \\ w(\omega) \end{pmatrix} = -P^{-1} R(-\omega)^T X(\omega) R(\omega) P \begin{pmatrix} v(\omega) \\ w(\omega) \end{pmatrix}, \tag{17}$$

where $X(\omega)$ is an $|R| \times |R|$ diagonal matrix with entries $X_{rr}(\omega) = e^{-\omega T_r}/(\omega T_r)$, and P is a $(|S| + |J|) \times (|S| + |J|)$ diagonal matrix with entries $P_{ss} = 1$, $P_{jj} = (p'_j)^{\frac{1}{2}}$, and $R(\omega)$ is a $|R| \times (|S| + |J|)$ matrix where

$$R_{rs}(\omega) = \left(-U''_{s(r)} T_r \frac{\kappa_r x_r}{\lambda_r} \right)^{\frac{1}{2}}, r \in s$$

$$R_{rj}(\omega) = e^{-\omega T_{jr}} \left(\frac{\kappa_r x_r}{\lambda_r} T_r p'_j \right)^{\frac{1}{2}},\ j \in r$$

and all other entries are 0.

The matrix $G(\omega) = P^{-1} R(-\omega)^T X(\omega) R(\omega) P$, which appears in (17) is called the *return ratio* for (v, w). From the generalized Nyquist stability criterion [5, 8] it is sufficient to prove that the eigenvalues of the return ratio $G(\omega)$ do not encircle the point -1 for $\omega = i\theta$, $-\infty < \theta < \infty$, in order to deduce that $(v(t), w(t)) \to 0$ exponentially as $t \to \infty$. Note, we are not, at this stage, interested in the asymptotic behaviour of $u(t)$.

If λ is an eigenvalue of the return ratio then we can find a unit vector z such that

$$\lambda z = R(i\theta)^\dagger X(i\theta) R(i\theta) z,$$

where $\dagger$ represents the matrix conjugate, and hence

$$\lambda = z^\dagger R(i\theta)^\dagger X(i\theta) R(i\theta) z.$$

If $d = R(i\theta)z$ then, since X is diagonal,

$$\lambda = \sum_r |d_r|^2 X_{rr}(i\theta) = \sum_r |d_r|^2 \frac{e^{-i\theta T_r}}{i\theta T_r}.$$

Hence $\lambda = K\zeta$, where $K = \|R(i\theta)z\|^2$ and ζ lies in the convex hull of

$$\Big\{\frac{e^{-i\theta T_r}}{i\theta T_r} : r \in s, s \in S\Big\}.$$

The convex hull includes the point $-2/\pi$ on its boundary (at $\theta T_r = \pi/2$), but contains no point on the real axis to the left of $-2/\pi$ [23], and hence if λ is real then $\lambda \geq (-2/\pi)K$.

Next we bound K. Let Q be the $(|S| + |J|) \times (|S| + |J|)$ diagonal matrix taking values $Q_{ss} = y_s\sqrt{-U_s''}$ and $Q_{jj} = z_j\sqrt{p_j'}$, let $\rho(\cdot)$ denote the spectral radius, and $\|\cdot\|_\infty$ the maximum row sum matrix norm. Then

$$\begin{aligned} K &= z^\dagger R(i\theta)^\dagger R(i\theta)z \\ &\leq \rho(R(i\theta)^\dagger R(i\theta)) \\ &= \rho(Q^{-1}R(i\theta)^\dagger R(i\theta)Q) \\ &\leq \|Q^{-1}R(i\theta)^\dagger R(i\theta)Q\|_\infty \\ &< \frac{\pi}{2}, \end{aligned}$$

the last inequality following from (12).

So we have that $\lambda > -1$ for any real eigenvalue λ. Thus, when the loci of the eigenvalues of $G(i\theta)$ for $-\infty < \theta < \infty$ cross the real axis, they do so to the right of -1. Hence the loci of the eigenvalues of $G(i\theta)$ cannot encircle -1, the generalized Nyquist stability criterion is satisfied and the system (14-16) is stable, in the sense that $v_s(t) \to 0$, $w_j(t) \to 0$ exponentially, for all s, j, as $t \to \infty$. There remains the possibility that $x(t)$ might hit a boundary of the positive orthant, and invalidate the linearization (14-16). To rule out this possibility, note that there exists an open neighbourhood of x, say $\mathcal{N}'$, such that whilst $x(t) \in \mathcal{N}'$, the linearization is valid, and so $y_s(t) \to y_s$ and $z_j(t) \to z_j$ exponentially. Now $(\dot{x}_r(t), t > 0)$ is defined by (1-4) as a function of $(y(t), z(t), t > -T_{max})$. Thus, whilst $x(t) \in \mathcal{N}'$, $\dot{x}_r$ decays exponentially to 0 for all r and therefore, the total distance $x_r(t)$ can travel from $x_r(0)$, whilst remaining in $\mathcal{N}'$, is bounded by

$$\gamma \max_{t\in(-T_{max},0)} \|(y(t), z(t)) - (y, z)\|$$

for some γ. Hence we can pick an open subset, $\mathcal{N} \subset \mathcal{N}'$ such that if $x(t) \in \mathcal{N}$ for $t \in (-T_{max}, 0)$ then $x(t) \in \mathcal{N}'$ for all t. Furthermore, if $x(t) \in \mathcal{N}'$ for all t then, since $\dot{x}_r$ decays exponentially to 0, $x(t)$ must be Cauchy, and must therefore tend to a limit. Thus $\mathcal{N}$ is as required.

Finally we shall relax the assumption that $x_r > 0$ for all r. Since x is an interior equilibrium point, $x_r = 0$ implies $\dot{x}_r(t) < 0$. Thus there is a neighbourhood of x, say $\mathcal{M}$, such that, on $\mathcal{M}$, the linearization of (1-4) coincides with the case where we discard all r such that $x_r = 0$. Therefore, as above, we may choose an open neighbourhood $\mathcal{N} \subset \mathcal{M}$ such that for any initial trajectory $(x(t), t \in (-T_{max}, 0))$ lying within the neighbourhood $\mathcal{N}$, $(y(t), z(t))$ converge exponentially as $t \to \infty$ to the unique solution (y, z) to the optimization problem (8). $\square$

Remark 1. The linearization (14-16) is similar to that arising in the treatment by Johari and Tan [10], Massoulié [16] and Vinnicombe [23] of the case where each source-destination pair has a single route. Comparing our linearization with theirs, it is as if we transform our model by treating each route as arising from a separate source, and for each $s \in S$ we add a fictitious link $l(s)$ to each of the routes $r \in s$, with $T_{l(s)r} = 0$ and $p_{l(s)}(y_s) = -U_s'(y_s)$. A complication is the non-uniqueness of x, and hence our need to approach the result via the convergence of (y, z).

Remark 2. Theorem 2 remains valid if equation (1) is replaced by (5). This corresponds to the added links discussed in remark 1 having the property that $T_{l(s)r} = T_r$ rather than $T_{l(s)r} = 0$: the flows from source s share a fictitious link as they *leave* the source s, rather than as they *return* to source s.

Remark 3. In [7] a system similar to (1-4), is considered, but where, instead of equation (4), $y_s(t) = \sum_{a\in s} x_a(t)$. The sufficient condition for local stability obtained in [7] restricts κ_r by round-trip times of all routes serving $s(r)$, and can be onerous if the round-trip times of the routes serving a source-destination pair are heterogeneous. We have shown that using delayed information, either $x_a(t - T_a)$ or $x_a(t - T_r)$ rather than $x_a(t)$, allows a fully decentralized sufficient condition.

Remark 4. If

$$U_s(y_s) = \frac{w_s y_s^{1-\alpha_s}}{1 - \alpha_s},$$

and

$$p_j(z_j) = \left(\frac{z_j}{C_j}\right)^{\beta_j}$$

then the condition (12) is satisfied if, for each $r \in R$,

$$\kappa_r T_r(\alpha_{s(r)} + \max(\beta_j, j \in r)) < \frac{\pi}{2}.$$

Observe that the importance of the source parameter α_s, relative to the resource parameters β_j, increases as α_s increases. The condition (7) arises as the special case where $\alpha_s = \alpha, s \in S$, and $\beta_j = \beta, j \in J$.

Remark 5. If we consider a network consisting of one source s, one route $r \in s$ and one link $l \in r$, then the linearization of this system is

$$\dot{u}(t) = -\kappa\,(\alpha + \beta)\,u(t - T). \qquad (18)$$

The Nyquist stability criterion, applied to the Laplace transform of this differential equation tells us that our system is locally stable if and only if

$$\kappa T\,(\alpha + \beta) < \frac{\pi}{2}.$$

Indeed if this inequality were replaced by equality, then $u(t) = \sin(\pi t/2T)$ solves equation (18), an oscillatory solution with period $4T$. Thus, for this example, our condition is tight.

3. AN EXTENSION OF SCALABLE TCP

In this Section we consider a refinement of the fluid-flow model used earlier. The refinement is intended to better approximate the behaviour of a network like the Internet, where congestion is indicated by a dropped or marked packet, and hence where a single packet crossing the network generates just a single bit of information concerning congestion along its route. The refinement follows [23] and we use it to develop a routing extension of scalable TCP [14], a variant of TCP with certain attractive scaling properties that we show are inherited by our multi-path extension.

The single bit of information is carried back to the source by the acknowledgement stream. The rate at which acknowledgements from route r arrive back at the source $s(r)$ at time

t is $x_r(t-T_r)$, since these acknowledgements arise from packets sent a time T_r previously. Let $\lambda_r(t)$ be the proportion of these acknowledgements that indicate congestion. These acknowledgements arise from packets that passed through resource j a time T_{jr} previously: thus

$$\lambda_r(t) = 1 - \prod_{j\in r}(1-\mu_j(t-T_{jr})). \qquad (19)$$

where $\mu_j(t)$ is the proportion of packets marked at resource j at time t, under the approximation that packet marks at different resources are independent. Observe that (2) approximates (19) when the probabilities $\mu_j, j \in J$, are small. The flow on route r that is seen at resource j at time t left the source for route r a time T_{rj} previously: hence

$$\mu_j(t) = p_j\left(\sum_{r:j\in r} x_r(t-T_{rj})\right), \qquad (20)$$

as in (3). Whereas in Section 2 the functions $p_j(\cdot)$ were not necessarily bounded, in this Section we presume they are bounded above by 1, in line with their interpretation here as a drop or mark probability rather than a price.

Suppose the sending rate $x_r(t)$ on route r at time t varies according to following algorithm: on receipt of a positive acknowledgement the sending rate is increased by $\bar{a}/T_r$, and on receipt of a negative acknowledgement indicating congestion the sending rate is decreased by $b_r x_r(t)/T_r$. This corresponds to the stable flow control algorithm of [15]: particular choices for $\bar{a}, b_r$ give scalable TCP [14].

We are now ready to define our routing extension. We suppose that the response on receiving a negative acknowledgement is altered. Now, on receipt of a negative acknowledgement the sending rate is decreased by $b_r y_{s(r)}(t)/T_r$, where $y_s(t)$ is given by equation (4). The fluid-flow model becomes

$$\frac{d}{dt}x_r(t) = \frac{x_r(t-T_r)}{T_r} \cdot \left(\bar{a}\ (1-\lambda_r(t)) - b_r\ y_{s(r)}(t)\ \lambda_r(t)\right)^+_{x_r(t)}. \qquad (21)$$

Let x be an equilibrium point. Then

$$\sum_{a\in s(r)} x_a = \frac{\bar{a}}{b_r}\cdot\frac{1-\lambda_r}{\lambda_r} \qquad (22)$$

for any route r with $x_r > 0$. Let $x_r(t) = x_r + u_r(t)$, for those routes with $x_r > 0$. Then linearizing about x and using the relation (22) we obtain

$$T_r\frac{d}{dt}u_r(t) = -\bar{a}(1-\lambda_r) \cdot \left(\frac{x_r}{\sum_{a\in s(r)} x_a}\sum_{a\in s(r)} u_a(t-T_a) + \frac{x_r}{\lambda_r}\nu_r(t)\right) \qquad (23)$$

where

$$\nu_r(t) = \sum_{j\in r}\frac{p'_j}{1-p_j}\sum_{a:j\in a} u_a(t-T_{aj}-T_{jr}). \qquad (24)$$

The method of Theorem 2 may be applied to this linearized system, and we find that local stability of (23-24) is implied by $\|R(i\theta)^\dagger R(i\theta)\|_\infty < 1$ where

$$R_{r,s}(\omega) = \left(\frac{\bar{a}x_r(1-\lambda_r)}{y_s}\right)^{\frac{1}{2}} \qquad r\in s$$

$$R_{rj}(\omega) = e^{-\omega T_{jr}}\left(\frac{\bar{a}x_r(1-\lambda_r)p'_j}{\lambda_r(1-p_j)}\right)^{\frac{1}{2}} \qquad j\in r$$

and all other entries are 0. For this $R(\omega)$, $\|R(i\theta)^\dagger R(i\theta)\|_\infty$ is less than 1 if

$$\bar{a}\frac{1-\lambda_r}{\lambda_r}\left(\lambda_r + \sum_{j\in r}\frac{z_j p'_j}{1-p_j}\right) < \frac{\pi}{2}. \qquad (25)$$

Suppose that the functions $p_j(\cdot), j\in J$, are given by equation (6). Then

$$\frac{1-\lambda_r}{\lambda_r}\sum_{j\in r}\frac{z_j p'_j}{1-p_j} \le \beta.$$

Thus, from (25), a sufficient condition for local stability is that

$$\bar{a}\,(1+\beta) < \frac{\pi}{2}. \qquad (26)$$

The corresponding condition for local stability of scalable TCP [14, 23] is $\bar{a}\beta < \frac{\pi}{2}$, and so the introduction of routing makes stability only a little harder to ensure.

Remark 1. If $b_r = 1/w_s, r\in s$, then from (22) the marking rate λ_r is the *same* on every route $r \in s$ with $x_r > 0$. Further, the total flow rate serving s is proportional to the weight w_s and approximately inversely proportional to this common value of λ_r, corresponding to a resource allocation across source-destination pairs that is approximately weighted proportionally fair [13]. In particular, if a source-destination pair s has more routes across the network then this may aid the network to balance load, and may help the resilience and reliability achieved from the network by the source-destination pair s; but the weight in the fairness criterion, namely w_s, is unaffected by the number of routes to which s has access. The marking rate on unused routes serving s is at least as high as the common value of λ_r on the routes used, provided some probing mechanism ensures a flow can escape from zero if the second term in the product (21) is positive.

Remark 2. As in Section 2, the same condition is sufficient for local stability if, in equation (21), $y_{s(r)} = \sum_{a\in s(r)} x_a(t-T_a)$ is replaced by $\sum_{a\in s(r)} x_a(t-T_r)$. The first sum, $y_{s(r)}$, is the rate of acknowledgements arriving back at the source $s(r)$ at time t, summed over all routes serving $s(r)$. An implementation might record $x_a(t)$ in a packet leaving on route a at time t, copy the value into the acknowledgement for the packet, and thus return it for use by $s(r)$ a time T_a later. In contrast, $\sum_{a\in s(r)} x_a(t-T_r)$ is the aggregate flow rate leaving the source $s(r)$ at time $t-T_r$. An implementation might record the aggregate rate $\sum_{a\in s(r)} x_a(t)$ in a packet leaving $s(r)$ at time t, copy the value into the acknowledgement for the packet, and thus return it to $s(r)$ via route r a time T_r later.

Remark 3. In the model (19), (20), (21) the equation (20) represents the marking probability at a resource as a function of the instantaneous flow through the resource. Suppose instead the marking probability at a resource is a function of an exponentially weighted average of the flow through the resource. Consider the system (19), (21) where, instead of equation (20),

$$\mu_j(t) = p_j(z_j(t)), \quad \delta_j\frac{d}{dt}z_j(t) = \sum_{a:j\in a} x_a(t-T_{aj}) - z_j(t).$$

Voice [25] establishes a decentralized sufficient condition for the local stability of this system, of the same form as that described in this paper, building upon the results of Vinnicombe [23, 24] for the case without routing.

Remark 4. The function (6) is a natural form to be implemented by active queue management [15, 24]: for example, $z_j(t)$ might be estimated as in the previous remark, and packets marked accordingly. For large buffers operating with drop tail, a more reasonable approximation for the proportion of packets overflowing the buffer is [22]

$$p_j(z_j) = [z_j - C_j]^+/z_j.$$

With this form, the sufficient condition (23) for local stability is satisfied if

$$b_r(\lambda_r + M_r) \sum_{a \in s(r)} x_a < \frac{\pi}{2}$$

where $M_r = \sum_{j \in J} I[p_j > 0] A_{jr}$, the number of saturated resources on route r. This is a much less attractive condition than (26): as well as the dependence on M_r, which may not be known at the edge of the network, its dependence on x prevents it scaling to arbitrary flow rates. The network may be stable for certain capacities and flow rates, but may become unstable with larger capacities and flow rates. In contrast the condition (26) is indeed scalable: as for scalable TCP in the single route case, the flow rates x, the marking rates λ, even the network topology, are notable by their absence from the stability condition.

4. CONCLUSION AND DISCUSSION

In this paper we have used a fluid-flow model to analyse the local stability of an end-to-end algorithm for joint routing and rate control. We have seen that stable, scalable load-sharing across paths, based on end-to-end measurements, can be achieved on the same rapid time-scale as rate control.

In the Internet there is generally a single path from a source to a destination, or a pre-determined split of traffic across a set of paths [21], mirroring the layering within the TCP/IP stack, where rate control is part the transport layer but routing is considered to be part of the network layer. The optimization and control framework of this paper sheds light on an aspect of this layering (cf. [3]), and on the possible separation (cf. [29]) of routing information into slowly varying structural information, able to provide a source-destination pair with a collection of available paths, and dynamic information, determined from end-to-end measurements and used, in our proposal, by a source-destination pair to balance load across paths. Our results suggest that while structural information may be provided by the network layer, load-balancing is more naturally part of the transport layer. In particular, we have observed that, for dynamic routing, the key constraint on the responsiveness of each route is the round-trip time of that route, information which is naturally available at sources.

5. ACKNOWLEDGEMENTS

The authors are grateful to Bob Briscoe, Jon Crowcroft, Christophe Diot, Richard Gibbens, Damon Wischik and two anonymous referees for their comments on an earlier draft of this paper. The research of the authors is supported in part by EPSRC grant GR/S86266/01.

6. REFERENCES

[1] M. Beckmann, C.B. McGuire and C.B. Winsten. *Studies in the Economics of Transportation.* Cowles Commission Monograph, Yale University Press, 1956.

[2] D. Bertsekas and R. Gallager. *Data Networks*, 2nd edition. Prentice-Hall, New Jersey, 1992.

[3] M. Chiang. Balancing transport and physical layers in wireless multihop networks: jointly optimal congestion control and power control. *IEEE J. Sel. Areas Comm.*, 23:104–116, 2005.

[4] J. Crowcroft, R. Gibbens, F. Kelly, and S. Östring. Modelling incentives for collaboration in mobile ad hoc networks. *Performance Evaluation*, 57:427–439, 2004.

[5] C.A. Desoer and Y.T. Yang. On the generalized Nyquist stability criterion. *IEEE Transactions on Automatic Control*, 25:187–196, 1980.

[6] S.J. Golestani. *A Unified Theory of Flow Control and Routing in Data Communication Networks.* PhD thesis, MIT, Dept. of Electrical Engineering and Computer Science, Cambridge, MA, 1980.

[7] H. Han, S. Shakkottai, C. Hollot, R. Srikant, and D. Towsley. Overlay TCP for multi-path routing and congestion control. In *ENS-INRIA ARC-TCP Workshop*, Paris, France, 2003.

[8] O.L.R. Jacobs. *Introduction to Control Theory.* Oxford University Press, Oxford, 1993.

[9] V. Jacobson. Congestion avoidance and control. *Computer Communication Review* 18(4): 314–329, 1988.

[10] R. Johari and D.K.H. Tan. End-to-end congestion control for the Internet: delays and stability. *IEEE/ACM Transactions on Networking*, 9:818–832, 2001.

[11] D.B. Johnson and D.A. Maltz. Dynamic source routing in ad hoc wireless networks. In T. Imielinski and H. Korth, editors, *Mobile Computing*, 153–181. Kluwer, 1996.

[12] F. Kelly. Fairness and stability of end-to-end congestion control. *European Journal of Control*, 9:159–176, 2003.

[13] F.P. Kelly, A.K. Maulloo, and D.K.H. Tan. Rate control in communication networks: shadow prices, proportional fairness and stability. *Journal of the Operational Research Society*, 49:237–252, 1998.

[14] T. Kelly. Scalable TCP: improving performance in highspeed wide area networks. *Computer Communication Review*, 32(2):83–91, 2003.

[15] T. Kelly. *Engineering Flow Controls for the Internet.* PhD thesis, Department of Engineering, University of Cambridge, 2004. http://www-lce.eng.cam.ac.uk/∼ctk21/papers/

[16] L. Massoulié. Stability of distributed congestion control with heterogeneous feedback delays. *IEEE Transactions on Automatic Control*, 47:895–902, 2002.

[17] M. Mathis, J. Semke, J. Mahdavi, and T. Ott. The macroscopic behaviour of the TCP congestion avoidance algorithm. *Computer Communication Review*, 27:67–82, 1997.

[18] J. Mo and J. Walrand. Fair end-to-end window-based congestion control. *IEEE/ACM Transactions on Networking*, 8:556–567, 2000.

[19] F. Paganini, Z. Wang, J.C. Doyle, and S.H. Low. Congestion control for high performance, stability and fairness in general networks. *IEEE/ACM Transactions on Networking*, 2005.

[20] A. Papachristodoulou, L. Li, and J.C. Doyle. Methodological frameworks for largescale network analysis and design. *Computer Communication Review*, 34(3):7–20, 2004.

[21] A. Sridharan, R. Guérin, and C. Diot. Achieving near-optimal traffic engineering solutions for current OSPF/IS-IS networks. *IEEE/ACM Transactions on Networking*, 2005.

[22] R. Srikant. *The Mathematics of Internet Congestion Control*. Birkhauser, 2004.

[23] G. Vinnicombe. On the stability of networks operating TCP-like congestion control. *Proc. IFAC World Congress*, Barcelona, Spain 2002.

[24] G. Vinnicombe. Robust congestion control for the Internet. 2002.

[25] T. Voice. Delay stability results for congestion control algorithms with multi-path routing. 2004. http://www.statslab.cam.ac.uk/∼tdv20

[26] Z. Wang and J. Crowcroft. Analysis of shortest-path routing algorithms in a dynamic network environment. *Computer Communication Review*, 22(2):63–71, 1992.

[27] J.G. Wardrop. Some theoretical aspects of road traffic research. *Proceedings of the Institution of Civil Engineers*, 1:325–378, 1952.

[28] S. Yilmaz and I. Matta. On the scalability-performance tradeoffs in MPLS and IP routing. In *Proceedings of SPIE ITCOM'2002: Scalability and Traffic Control in IP Networks*, Boston, MA, 2002.

[29] D. Zhu, M. Gritter, and D.R. Cheriton. Feedback based routing. *Computer Communication Review*, 33(1):71–76, 2003.

Granular Differentiated Queueing Services for QoS: Structure and Cost Model

Shengming Jiang
School of Electronic & Information Engineering
South China University of Technology
email: shmjiang@scut.edu.cn

Abstract— **One weakness of DiffServ is the lack of granularity for QoS guaranteed services, which makes it difficult to cost-effectively support end-to-end (e2e) QoS according to the e2e situation (e.g., path lengths) of applications. With the conventional packet-level QoS mechanisms for the regulated traffic, i.e., buffer admission control plus output schedulers in general, increasing service granularity may inevitably complicate implementation and/or impact scalability since sophisticated output schedulers seem necessary in this case. In this paper, a new structure, Differentiated Queueing Services (DQS), is discussed to handle the above issue. DQS tries to provide granular and scalable QoS guaranteed services to be selected by users according to their QoS requirements and e2e situation. Its basic idea is to convert packet delay guarantee into packet loss ratio guarantee with either dropping or marking the packets whose e2e delays are perceived unable to be guaranteed. Packets are queued according to their e2e delay requirements so that various delay bounds can be guaranteed without using sophisticated output schedulers while different packet loss ratios are mainly controlled by call admission control (CAC). To this end, the e2e QoS requirement is carried by each packet to avoid storing such information in network units for scalability. On the other hand, differentiated services should also be accomplished with a differentiated cost model for pricing, not only for the profits of both the service provider and the user, but also to prevent good services from being abused. So, a cost model for differentiated QoS services provided by DQS is also discussed with a possible CAC based on the exponentially bounded burstiness traffic.**

Keywords: Service Granularity, End-to-end QoS, Differentiated Queueing Services (DQS).

I. Introduction

A. Background

The new generation network needs to provide end-to-end (e2e) QoS cost-effectively according to the QoS requirements and the e2e situation (e.g., path lengths) of applications. How to support QoS has been studied for about two decades and lots of structures have been proposed especially for ATM and IP networks. Many proposals try to trade off between service granularity (e.g., per-flow or per-class) and simplicity & scalability of implementation, often using the following mechanisms, namely, resource reservation at the CAC [1] level and traffic policing & shaping, scheduling [2] as well as buffer admission control (BAC) at the packet-level. To provide granular QoS for different flows, schedulers has to know the QoS requirement of each flow and treat them accordingly (e.g., ATM and IntServ). Such per-flow treatment with storing voluminous information in network units is the major cause of the scalability problem and complicates implementation. To avoid this problem, the class-based approach suggests to aggregate the flows with the same QoS requirement into one class so that network units only need to handle classes instead of individual flows for QoS provisioning. DiffServ is a typical example of this approach. So far, two services, namely, premium service (PS) and assured service (AS), have been defined for DiffServ [3], [4], [5], [6]. PS provides 'a low loss, low latency, low jitter' e2e service through the DiffServ domain with assured bandwidth [3] mainly for delay-sensitive applications with virtual leased-line services. AS differentiates service classes with relative QoS assurance by using such as active buffer management, with emphasis on expected throughput and more sub-classes of services. Now four sub-classes have been defined according to the combination of various packet drop precedences [6].

There is lots of follow-up to enhance DiffServ reported in the literature. However, its lack of QoS granularity has not been addressed adequately. With the services defined so far in DiffServ, it seems only PS promises QoS guarantee while AS only provides assured QoS in terms of throughput. With aggregating multiple individual flows into one to be serviced by PS, this aggregate flow should be serviced such that the most stringent individual QoS requirement must be satisfied; otherwise, it is unclear how various e2e QoS requirements can be guaranteed with flow aggregation [4], [7]. This weakness also raises the following issues: (1) whether flow aggregation is cost-effective from the point of view of individual flows and (2) are the services defined in DiffServ sufficient to cost-effectively support new applications in future?

For both the service provider and the user, it is reasonable to differentiate charges for different services accordingly. In DiffServ, the flows aggregated into the same class are treated equally and should be also charged at the same rate accordingly. However, from the e2e point of view, such treatment may be unnecessary and the subsequent charge is unfair to some flows even running the same applications. This is because, although the e2e QoS requirements from the same applications are identical, the difficulty in provisioning such e2e QoS in a node still depends on and varies with the situation of the path that each flow is going through, particularly the number of hops along the path. In general, the less number of hops, the looser QoS support is required in each node

The author was with Institute for Infocomm Research (I^2R), 21 Heng Mui Keng Terrace, Singapore 119613.

to guarantee the same e2e QoS. With the current DiffServ structure (i.e., either PS or AS), an application may not be able to find an e2e service to satisfy its e2e QoS requirements with low cost according to its path situation. For example, a flow requiring QoS guarantee over a short path may be aggregated with another having the same QoS requirement but over a much longer path. This may lead to over-provisioning of QoS to the former while the user should pay accordingly. More discussion on this issue can be found in Section III-C.

Regarding the second issue, DiffServ suggests to aggregate the flows with identical QoS requirements into one; if there are two types of packets co-existing in the same flow, one is better serviced by PS while the other is better with AS, how such kinds of flows can be serviced cost-effectively? The diversity of QoS requirements and the mixture of QoS requirements within one flow may increase as new applications such as grid computing [8] appear in the future. Adding new DiffServ services may need upgrading existing implementation since per-hop behaviors (PHB) defined for DiffServ are realized through buffer management and output scheduler, which usually are implemented in hardware for high speed because both need to make decisions on-the-fly. Compared to the buffer management, the output scheduler is more critical to QoS provisioning since it decides which packets and when to be serviced within the packet time scale. Although DiffServ sufficiently decouples service provisioning and traffic conditioning from the forwarding behaviors [9], adding new services maybe still require changes in the above implemented components especially schedulers.

There are also some approaches on "stateless per-flow QoS guarantee" such as [10], [11], [12], which provides per-flow QoS guarantee with less scalability problem. Their basic ideas include (i) allowing packets to carry flow state information to avoid per-flow information storage [10] and per-flow CAC [11] in core routers and (ii) properly re-aggregating the aggregate flows belonging to the same class from different links in a router using the class-based scheduler [12]. These approaches still relay on sophisticated output schedulers, which themselves may become bottlenecks especially for high-speed links [13]. Furthermore, the per-flow treatment may still cause scalability problem. There are also some incremental upgrading approaches such as [14], [15], [16], [17], which either adds an additional service (e.g., better than the best-effort service in term of delay) or enhances the current best-effort service. However, they did not address sufficiently the above mentioned two issues.

B. *A proposal*

With an increased number of QoS services, a flow's e2e QoS can be supported by a combination of various services available at each node along the path which can best match the flow's QoS requirement at the lowest cost. Services at different nodes for a combination are unnecessarily identical to each other. Such a structure will be discussed below by suggesting some adjustments to typical QoS mechanisms, i.e., CAC, scheduler and BAC. Packet loss ratio is determined by CAC and further tuned by BAC while the delay and jitter are mainly determined by schedulers. For jitter, the most popular approach in the literature is adopted in this paper. That is, the jitter is handled at the end-point of the network (e.g., network egress or destination) in order to simplify implementation and reduce operations inside of networks. So, here only the e2e delay and loss ratio are discussed.

Here we suggest to convert delay guarantee into loss ratio guarantee with enforcing explicit delay guarantee to packets since delay is instantly measurable while loss ratio is statistically measurable. A node can know instantly whether or not the delay of a packet can be guaranteed if the delay requirement is explicitly expressed. The packet loss ratio cannot be known instantly and it is not straightforward for a node to control it instantly according to particular loss ratio requirements. Packet loss ratio can be mainly determined at the CAC level through such as queueing analysis. On the other hand, if a packet's e2e lifetime is confirmed expiring, it should be dropped immediately. Consequently, all packets successfully received by their destinations should be surely guaranteed in terms of their e2e delays. The dropping ratio of deadline-due packets can be analyzed jointly with packet loss ratio caused by buffer overflow at the CAC level. Therefore, with this suggestion, the mechanisms for QoS can be further simplified by avoiding the use of sophisticated output schedulers (in [13], a structure not using output scheduler was discussed, which however is based on per-flow virtual queue management), while some techniques developed to analyze 'schedulable condition' available in the literature may still be useful to analyze packet dropping ratio here. Note that in practice it is difficult for a node to perceive accurately a packet's real journey time over the remaining path. So, intermediate nodes can only estimate a packet's e2e delay validity. In this case, in stead of 'being dropped immediately', the estimated deadline-due packets can be marked as 'non-guaranteed packets', which will be serviced only when no guaranteed packet is waiting for service. More discussion on this issue can be found in Section II-B.

DQS requires the expression of the e2e delay requirement at the packet level in addition to packet dropping precedence. Traditionally, applications have been asked to express their QoS requirements (if any) at the CAC level. At the packet level, only the packet dropping precedence is often carried by packets to decide which packets are dropped first in the case of congestion. Consequently, network units have to store the information on delay requirement for packet scheduling. Storing and handling such information per-flow or per-connection lead to the scalability problem. Similar to [10], [11] which lets packets carry the flow state information, DQS also allows each packet to carry the e2e delay requirement. More discussion can be found in Section II-A.

With the above suggestions, the QoS effort at the packet level can focus on queueing packets. That is, when a newly arriving packet is admitted to the buffer, its queueing sequence in the buffer is decided according to its delay requirement, which is arranged from the shorter (i.e., at or close to the head of the queue) to the longer. This queueing order just decides the service order that a packet will receive and naturally differentiates services according to its e2e delay requirement, hence the name 'Differentiated Queueing Services' (DQS). Such service is similar to the earliest-deadline-first (EDF) scheduler [18], [19] in terms of the manner of deciding service order. However, DQS enforces explicit e2e delay guarantee without using output schedulers, which are different from EDF as discussed in Sections II-B.1 and II-B.2. Extra costs with DQS include increased packet overhead and sophisticated BAC and CAC. Intuitively, the former is expected to be compensated somewhat by the reduced cost of the more granular services offered by DQS. Regarding BAC, queueing packets according to e2e delay needs insertion operation, which can be considered jointly with active buffer management (ABM) [20] schemes to reduce the overall complexity of implementation. If the queue size is so small that FIFO will not impact delay guarantee, inserting newly arriving packets may be also replaced by FIFO queueing in this case to maintain high link rate. This is different from output schedulers which need to perform the same operation to each packets to be transmitted. On the other hand, compared to schedulers, CAC should have more space for being sophisticated since it is operated at the call level.

The paper is organized as follows. The structure of DQS is described in Section II with five subsections corresponding to the major issues arising, i.e., the format of e2e delay expression at the packet level in Section II-A, per-hop-behavior (PHB) in Section II-B, CAC in Section II-C, service selection in Section II-D and cost model in Section II-E. In Section III, more discussion on the proposed cost model with a possible CAC based on the exponentially bounded burstiness (EBB) traffic is also provided. Finally, the paper is summarized in Section IV.

II. Differentiated queueing service (DQS)

DQS is based on the following assumptions. Any application should have a maximum e2e delay ($\mathbf{D}$) and a maximum e2e packet failure ratio ($\mathbf{F}$) that it can tolerate, where $\mathbf{F}$ is the sum of dropping ratio because of missing a deadline and loss ratio due to buffer overflow. If an application can have an arbitrary $\mathbf{D}$ and/or $\mathbf{F}$, then they can be set to certain values artificially. $\mathbf{D}$ should be treated as the lifetime of the associated packet. If a packet is confirmed that its lifetime has expired, it should be dropped immediately. The following sections evaluate some possible ways to express $\mathbf{D}$ at the packet level and discuss PHB, CAC and cost model for DQS.

The following notations are used in the discussion.

- a: packet's arrival time.
- c: node's link rate in bits/s.
- d: packet's maximum delay at a node decided by CAC.
- $\mathbf{e}$: latest time for a packet's departure from a node subject to its e2e delay.
- T: packet's maximum remaining lifetime upon its arrival at a node.

Subscript 'i' is often associated with these parameters to indicate that they are for node i while association '(j)' is sometimes used to indicate the j-th packet at the same node. Furthermore, to simplify discussion, the propagation delay is not addressed since it is a constant for a given path. Therefore, for a path consisting of n nodes, we can have the following relations between the above parameters for a packet arriving at node i.

$$T_i = \mathbf{D} - \sum_{j=1}^{i-1} \tilde{d}_j = T_{i-1} - \tilde{d}_{i-1} \qquad (1)$$

with $T_1 = \mathbf{D}$, where $\tilde{d}_i$ is the actual delay that the packet experiences at node i. The effective packet's maximum delay (which is defined as the interval between the arrival of the first bit and the departure of the last bit from the node for the same packet) allowed at node i subject to its e2e delay, $\hat{d}_i$, is given by

$$\hat{d}_i \quad = \quad T_i - \sum_{j=i+1}^{n} \tilde{d}_j. \qquad (2)$$

However, it is difficult for node i to determine $\tilde{d}_j$ for $j > i$. In practice, (2) can be approximated by $\hat{d}'$ as

$$\hat{d}'_i = T_i - \sum_{j=i+1}^{n} d_j. \qquad (3)$$

Similarly, if the upstream nodes have bounded the delay as promised during CAC, then

$$\mathbf{e} \quad = \quad a + \hat{d} \qquad (4)$$

which can also be approximated by $\mathbf{e}' = a + d$. Here, subscript for $\mathbf{e}$ is reserved to indicate the group or class that a packet belongs to as used later (e.g., Section II-B.2).

A. *Formats of* $\mathbf{D}$ *carried by packets*

Here two possible ways to express $\mathbf{D}$ at the packet level are briefly discussed in terms of accuracy, packet disorder and packet overhead.

a) Format 1 (F1): Since $\mathbf{D}$ indicates the packet's lifetime, intuitively, we can let each packet first carry $\mathbf{D}$ which then is deducted once the packet leaves a node according to (1). For BAC, $\hat{d}_i$ given by (3) is used to decide $\mathbf{e}$ according to (4), which determines the packet position in the buffer. The major advantage of this format is that the margin in service time, i.e., the sum of $(\hat{d}-\tilde{d}) > 0$ given by upstream nodes, can be used by downstream nodes to accommodate some instant burst traffic.

This method may disorder packets (even with the same $\mathbf{D}$) belonging to the same flow. This is because $\hat{d}$ for an early arriving packet at a node may be still larger than that of a later arrival (perhaps due to dynamic traffic load in its upstream nodes) such that the latter is queued before the early arrival according to their $\mathbf{e}$ settings. Another disadvantage is that d_i used in (3) needs to be either stored in each node along the path (which may lead to scalability problem) or carried by the packet (which causes more overhead).

b) Format 2 (F2): To avoid packet disorder with F1, another method is to let a packet carry d (real value or code), and BAC queues the packet according to $\mathbf{e}$ given by (4) with d rather than $\hat{d}$ used for F1. So disordering packets belonging the same flow with an identical $\mathbf{D}$ can be avoided.

However, if there are packets with different settings of $\mathbf{D}$ co-existing within the same flow, packets with different $\mathbf{D}$ settings may still be disordered due to possible different settings of d at the same node. Two possible solutions are (i) numbering those packets separately according to their $\mathbf{D}$ settings and (ii) filtering those packets according to d (or $\mathbf{D}$ if it is carried by packets) at the destination since no such problem exists for the packets with the same d.

Since a path often consists of multiple hops and each of which may give different d for the same flow. Therefore, there are two variances of the above method: (a) letting packets carry d given by each hop in the form "$d_n d_{n-1} ... d_2 d_1$", where d_i is the guarantee provided by hop i, and (b) letting packets carry a typical d (e.g., $\min[d_1, d_2, ...d_n]$) or an averaged d determined according to e2e delay. The former is more accurate than the latter to reflect $\mathbf{D}$ to each hop along the path at the expense of more overhead. This overhead can be reduced by each hop removing the part related to it after the packet visits the hop, i.e., hop i removes the field carrying d_i. The second one may lead to lower network resource utilization compared to the first one due to possible unnecessary bounding effort for the e2e delay.

In the remaining discussion, we just assume that $\mathbf{e}$ be known to the nodes since the final packet format for $\mathbf{D}$ should balance performance, packet overhead and other implementation issues.

B. Per-hop-behavior (PHB)

Basically, the PHB of DQS consists of the following operations: queueing arriving packets, forwarding the first packet in the queue and changing packet header if necessary. Since each packet is queued according to its $\mathbf{e}$ which exactly defines the forwarding order as discussed below, the forwarding operation is simple. Regarding changing packet header along with the forwarding operation, it mainly depends on the adopted formats of $\mathbf{D}$ discussed above and is more an implementation issue so it will not be discussed here in detail. The crucial part for DQS is queuing arriving packets, of which, the key operation is packet insertion according to certain queueing policies to be discussed later. How fast a packet is inserted into a queue depends on algorithms and data structures adopted for implementation (e.g., binary search tree [21]), which should be addressed jointly with BAC. In addition, due to some similarity between DQS and EDF, some implementation approaches for EDF schedulers (e.g., [22]) can be borrowed for implementing DQS. Therefore, this section focuses on queueing issues.

1) Strict queueing policy (SQP) : This is a non-preemptive queueing policy. Upon a packet arriving at a node indexed by '(.)', its $\mathbf{e}$, $\mathbf{e}(.)$, is determined according to (4). This packet may be inserted into the current queue according to its $\mathbf{e}(.)$ and those of all packets present in the queue. Precisely, if this packet is going to be inserted between two consecutive packets (if any) in the queue, namely packet j and packet $(j+1)$ with $\mathbf{e}(j) \leq e(j+1)$, where $\mathbf{e}(j)$ stands for the $\mathbf{e}$ of packet j. As mentioned earlier, Packets with smaller j are at or closer to the head of the queue and will be serviced earlier accordingly. This insertion operation should satisfy the following conditions expressed in time point for an infinity buffer since for implementation, a packet's delay bound requirement is often converted into the local time upon its arrival.

1) $\mathbf{e}(j) \leq \mathbf{e}(.) < e(j+1)$. That is, the packets with different $\mathbf{e}$ are ordered according to their $\mathbf{e}$ while those with identical $\mathbf{e}$ are queued according to their arrival sequences with FIFO.
2) $\mathbf{e}(.) \geq a(.) + \frac{\tilde{l}}{c}$, where $\tilde{l} = \sum_{i=0}^{j} l(i) + l(.)$ and $l(i)$ is the length of packet i ('$i = 0$' indicates the remaining of the packet currently in the service). That is, the current situation in the queue should be able to support this packet's delay requirement $d(.)$.
3) $\mathbf{e}(j') \geq a(.) + \frac{l'(j')}{c}$, $\forall j' \geq (j+1)$, where, $l'(j') = \sum_{i=0}^{j'} l(i) + l(.)$. That is, the delay guarantee for all packets to be affected by this insertion in the queue should be assured.

Condition 2 can also be simply expressed as $d(.) \geq \frac{\tilde{l}}{c}$. Condition 3 can be further simplified into to just checking packet $j' = j+1$. That is, if the insertion can satisfy the delay bound of packet $(j+1)$, those following it can be also satisfied as proved below. Given the delay bound of packet $(j+1)$ being satisfied, we should have $\mathbf{e}(j+1) \geq a(.) + \frac{l'(j+1)}{c}$. Since $\mathbf{e}(j+2) \geq \mathbf{e}(j+1) + \frac{l'(j+2)}{c}$, replacing $\mathbf{e}(j+1)$ with the above inequality, we have $\mathbf{e}(j+2) \geq a(.) + \frac{l'(j+2)}{c}$.

With a finite buffer, conditions 2 and 3 are necessarily limited by the buffer capacity. That is, there should be enough buffer space to insert the newly arriving packet and this insertion should not kick out any existing packet. Note that condition 1 decides where to insert a packet while conditions 2 and 3 control buffer admission.

From the above definition, it is easy to have the e2e delay jitter bound (i.e., the maximum e2e delay variation) for packets with $\mathbf{D}$ over a path consisting of n hops, $\tilde{D}$, as

$$\tilde{D} = \mathbf{D} - \sum_{h=1}^{n} \frac{l_{min}}{c_h}, \tag{5}$$

where l_{min} is the shortest packet length and c_h the link rate of hop h (again the propagation delay is ignored here). The proof is very simple. Since with DQS, a packet with $\mathbf{D}$ will be dropped if its e2e delay exceeds $\mathbf{D}$, $\mathbf{D}$ is just the maximum e2e delay for the successfully received packets. Clearly, $\sum_{h=1}^{n} l_{min}/c_h$ is the minimum e2e delay.

2) Priority function of DQS: We first group packets according to d following $d_1 > d_2 > ... > d_G$, where G is the total number of groups. Every packet in group g has the same d, namely d_g. Also, a 'g-packet' means a packet from group g and the same for g-flow or g-data. DQS is a dynamic priority-based queueing system and different from some priority systems such as head-of-line (HOL) and time-dependent (TD) ones [23]. In HOL, the priority of a packet from group p is explicitly defined and fixed against time. So, the packet's service order is only determined by the pre-assigned priority p. In TD, the priority function is $q_g(t) = (t - a_g)b_g$, where $0 \leq b_1 \leq b_2 ... \leq b_G$ are constant against t and a_g is the arrival time of a packet from group g. So, the priority changes with t.

For DQS, a packet's initial priority is determined according to its arrival time a_g and its delay requirement d_g, i.e., $\mathbf{e}_g$ given by (4), as follows:

$$q_{g,0} = \frac{1}{\mathbf{e}_g} \approx \frac{1}{a_g + d_g}. \tag{6}$$

which can be determined only upon its arrival. Its priority function can be expressed as

$$q_g(t) = \frac{1}{\mathbf{e}_g - t}. \tag{7}$$

As indicated by (7), when $t \to \mathbf{e}_g$ means the packet must be serviced and leave the system immediately in order to meet the delay bound. If a packet has not been serviced after $\mathbf{e}_g$, its priority becomes negative as indicated by (7), which means this packet should be either dropped or marked as non-guaranteed packet. However, the latter does not happen to EDF, with which, a packet's priority always increases with its dwelling time in the queue. In general, DQS is similar to EDF in terms of the way to differentiate service order according to condition 1 mentioned in Section II-B.1. However, EDF is only equivalent to condition 1. Furthermore, like other scheduling schemes, EDF aims at a common delay bound for all packets using the same service. Therefore, as mentioned in [24], [25], EDF by itself cannot provide efficient e2e delay guarantees. With DQS, the delay guarantee of both a newly arriving packet and the existing packets are the conditions for a packet admission to the buffer as mentioned earlier.

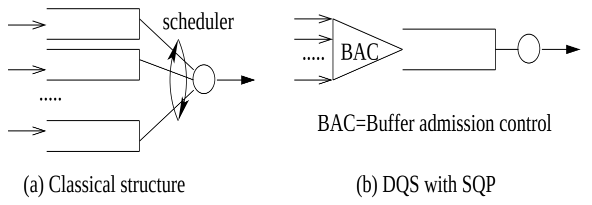


Fig. 1. Queue structures

3) Queue structure comparison: Classical network structures such as ATM, IntServ and DiffServ often adopt multiple queues linking with one output scheduler as illustrated in Fig. 1(a). For example, in ATM, even per VC/VP queue structure has been proposed while DiffServ adopts per class queue for aggregated flow. These multiple queues can be either physical (i.e., each sub-queue is allocated a physical buffer) or logical (a physical buffer is divided into sub-buffers each corresponding to one sub-queue). Given the same total size of buffer, the buffer utilization with multi-queue structure is lower than that with single queue structures since it is difficult to have inter-buffer capacity sharing with a multi-queue structure. As shown in Fig. 1(b), DQS for SQP adopts a single queue, which is more efficient to handle bursty traffic than multi-queue due to its higher buffer utilization. In addition, DQS does not use output scheduler but buffer admission control (BAC). Such structure can efficiently support queueing preemption, with which, later arriving high priority packets (e.g., voice) can preempt low priority packets (e.g., bursty data) in the queue to reduce high priority packet loss.

C. Call admission control

As discussed earlier, DQS can guarantee e2e delay ($\mathbf{D}$) and the remaining key issue is to guarantee e2e packet failure ratio ($\mathbf{F}$), which can be mainly determined by CAC. In general, a packet failure is due to either the violation of DQS's conditions mentioned in Section II-B or traffic congestion. This section is going to discuss a framework rather than detailed algorithms of CAC to guarantee **F** with SQP since for detailed algorithms, more information on traffic pattern and/or algorithms of traffic shaping & policing is needed.

The CAC here is to determine admission thresholds to be used by a hop to control flow admission such that packet failure ratio for the admitted flows requiring the same d_i will not exceed a predefined ratio Γ_i. The main issue is to calculate Γ_i, which however is difficult. As discussed in Section II-B, only if both conditions 2 and 3 are satisfied, a newly arriving packet can be admitted into the position defined by condition 1. By further considering the buffer capacity limitation, we can have the expression of Γ_i as

$$\begin{aligned}\Gamma_i &= 1 - \mathbb{P}\{\text{Conditions } 1 \sim 3 \text{ are satisfied}\} \times \\ &\quad \mathbb{P}\{\text{Successful packet insertion}\}, \end{aligned} \tag{8}$$

where, 'successful packet insertion' means an arriving packet can be inserted into the buffer according to conditions $1 \sim 3$ while this insertion does not cause buffer overflow.

It is almost impossible to directly apply existing schemes to calculate Γ_i because of the following reasons. As indicated in (8), Γ_i includes loss due to buffer overflow and dropping due to the limitation from queueing policies, and only the former is considered by most existing schemes. It is possible to refer to works on priority-queueing in the literature for this CAC, most of them are designed for HOL-alike fixed priority systems such as [26], [27]. However, as described in Section II-B.2, the priority in DQS is dynamic. Given d_i of group i, the priorities of packets from this group are further determined by their arrival times to the queue as indicated by (6)-(7), and it is impossible to distinguish between high and low priorities just based on d_i. Therefore, it is difficult to determine some important parameters used in the above mentioned algorithms, such as mean arrival rates of low and high priorities traffic. Some other works available in the literature only consider two priorities (e.g., [28]) or bufferless situation (e.g., [29]). One possible way is to borrow works on schedulablility conditions in terms of delay bound. Such a preliminary CAC is discussed in Section III.

D. Service selection

The services available in a network should be pre-defined with respect to the QoS metrics of each service and network resource utilization. These services can be selected by the user to form e2e services. Suppose a set of possible uni-cast paths, $\mathcal{P}$, is known. Generally, the steps below can be followed to select services.

1) Determine a set of paths from $\mathcal{P}$ with respect to the required e2e QoS $(\mathbf{F}, \mathbf{D})$, $\mathcal{P}_{qos}$. That is, $\mathcal{P}_{qos} = \{\varrho\}$ for all $\varrho \in \mathcal{P}$ with $\sum_{\forall h \in \varrho} d_{i(h)} \leq \mathbf{D}$ and $1 - \prod_{h=1}^{n}[1 -$

$\Gamma_{i(h)}] \leq \mathbf{F}$, where $d_{i(h)}$ and $\Gamma_{i(h)}$ are respectively d and Γ provided by hop h for service i.

2) Select the paths from $\mathcal{P}_{qos}$ with the lowest price through the on-line calculation template. This price is determined partially by the e2e cost of the provisioning of the services selected by the user, which can be estimated by Θ (11) to be discussed in Section II-E.
3) Each hop along the path needs to check whether a new admission to the selected service will affect QoS promised to both the existing and new sessions therein. Once the admission is accepted, no adjustment is required during the session lifetime.
4) If the selected service cannot pass any local CAC, the user can try other available services probably with higher cost.

Note that the above steps are necessary only for long session time applications like video, and can be set once forever for the same application if no change in the network structure happens. For short session time applications such as data, default services (e.g., best-effort) can be used.

Inevitably, the provisioning of granular services and service selection capability increases the operation complexity since more information on services and their utilization needs to be maintained while billing should be also granulated accordingly. This complexity increases with the number of provided services and may lead to additional operation cost. Therefore, although there may be many e2e service combinations with DQS, it does not mean that all combinations must be provided to the user. How many and what services to be provided are determined by the service provider with considering marketing, operation cost and other issues. For example, only services 1, 11 and 27 in Table II (Section III-C) can be provided due to their remarkable differences in e2e delays and costs. Furthermore, to simplify service selection, hops along a path can be clustered according to domains to create a list of per-domain services. Also the service provider can perform periodic searches to come up with some e2e service combinations of low cost for each path available in the network to simplify selection process. As discussed in the following section, the proposed cost model is static and e2e service combinations are stable for pre-defined per-node/domain services. The searching operation can be performed off-line and no need to modify results if no change occurs in network topologies and service capacity.

E. Cost model

Given a set of differentiated services, the price of each service should be also differentiated such that at least the luxury services should not be abused to have over-QoS-provision. Many papers in the literature discuss the leverage of pricing (e.g., [30], [31], [32]) for QoS or congestion control and the optimization of the revenue for the service provider (e.g., [33]), given the service prices. Some pricing structures based on the effective bandwidth can be also found in the literature (e.g., [34], [35]). And a cost-based pricing structure for adaptive applications in DiffServ environments is discussed in [36] (more survey can be found in [37]). Since DQS is different from DiffServ and the effective bandwidth in terms of QoS support as mentioned earlier, here a statistical cost model (θ) is discussed for each service provided by DQS, which is measured in terms of bandwidth over-provision ratio corresponding to effective services received by a flow with respect to its QoS requirement.

Let $i(h)$ denote a service $\langle d_{i(h)}, \Gamma_{i(h)} \rangle$ in hop h. The following notations are used in the discussion.

- λ: mean initial packet arrival rate of a flow under consideration.
- $\langle$QoS,T$_{spec}\rangle$: QoS requirement for a flow with the traffic characterized by T$_{spec}$ (e.g., peak rate).
- $\hat{r}_{i(h)}$: maximum average rate allowed for a flow with $\langle$QoS,T$_{spec}\rangle$ requiring service $i(h)$ if hop h uses all its link capacity (c_h) to support such service.
- $\theta_{i(h)}$: unit cost rate (per time unit) for service $i(h)$ with respect to $\langle$QoS,T$_{spec}\rangle$.
- θ_{ϱ}: unit e2e cost rate over a path ϱ.

By 'unit', we mean the related cost rate refers to a flow with $\lambda = 1$, which is also called 'unit flow' below.

The effective throughput offered by service $i(h)$ is $[1 - \Gamma_{i(h)}]/d_{i(h)}$, which is also expected by a flow using this service. However, this quantity cannot be used to measure the service cost since it does not reflect the degree of difficulties in provisioning the corresponding QoS service. For example, with a large buffer, $\Gamma_{i(h)}$ can be very small, which can lead to an effective throughput as high as that provided by very short $d_{i(h)}$. Usually, it is more difficult to provide fast services with short delay than slow ones. The degree of such difficulty can be measured simply by bandwidth over-provision ratio corresponding to a service, i.e., $c_h/\hat{r}_{i(h)}$ (called 'difficult coefficient' henceforth), which is usually larger than 1 since the over-provisioned bandwidth is usually required for QoS guarantee. Then, the product of the effective throughput and the difficult coefficient, i.e., $[1 - \Gamma_{i(h)}]c_h/[d_{i(h)}\hat{r}_{i(h)}]$, can effectively indicate the cost for a flow with $\lambda = \hat{r}_{i(h)}$ to use this service. Dividing the above cost by $\hat{r}_{i(h)}$, we can get such cost for the unit flow as follows:

$$\theta_{i(h)} = \frac{[1 - \Gamma_{i(h)}]c_h}{d_{i(h)}\hat{r}^2_{i(h)}}. \tag{9}$$

Assuming packet failure probability along each hop is independent, then,

$$\theta_{\varrho} = \sum_{\forall h \in \varrho} \Pi^{h}_{j=1}[1 - \Gamma_{i(j)}] \frac{c_h}{d_{i(h)}\hat{r}^2_{i(h)}}, \tag{10}$$

where $\Pi^{h}_{j=1}[1 - \Gamma_{i(j)}]$ indicates effective service offered by hops from 1 to h, i.e., the mean rate for successfully serviced packets of the unit flow. Here, the hops are numbered according to the packet travel sequence from 1. For a flow with λ and lasting a t time along path ϱ, the overall cost Θ can be simply estimated by

$$\Theta = t\lambda\theta_{\varrho}. \tag{11}$$

The user should be charged by the service provider at Θ, which is based on the cost of services selected by the user. The linearity of Θ to t and λ can provide a visibility for the user to estimate the overall cost. This cost model contains both the user's expectation and the effort of the service provider for the required service. It can also enable both the provider (for services provided) and the user (for services selected) to tradeoff between the cost and QoS requirements. Since this model is static, there is no need to account for every single packet. Note that, given c_h, $\hat{r}_{i(h)}$ further depends on T$_{spec}$ and the efficiency of CAC. Therefore, this model may cause unfairness to the user since it probably shifts some cost due to inefficient CAC to the user.

III. Discussions with exponentially bounded burstiness (EBB) traffic model

Here more discussions are given on service cost differentiation with a possible CAC based on EBB [38]. EBB is one of stochastically bounded traffic (SBT) models available in the literature. Compared to deterministic bounded traffic model, SBT can provide higher resource utilization [39]. With SBT, a traffic controller can be used at the ingress of a network to regulate traffic according to a given traffic bounding function. The reason of using EBB here is that a CAC scheme based on EBB reported in [40], [41] can be used for DQS. Therefore, in the following sections, more discussion on the proposed cost model based on this CAC is provided after a brief introduction to this CAC.

A. A preliminary CAC based on EBB

Here this CAC is discussed which is mainly used to investigate the proposed cost model. Lots of work is required for a practical CAC. First we discuss an approach for Γ_g estimation, which is similar to 'analyzing queue length in an infinite buffer to estimate loss in a finite buffer' such as [42]. That is, we consider a lossless queueing system, in which, the loss due to buffer overflow does not happen so that the failure probability for a g-packet with d_g is estimated by the probability that its delay exceeds d_g, i.e.,

$$\Gamma_g \approx \mathbb{P}\{\text{a } g\text{-packet's delay in a lossless queue} > d_g\}. \quad (12)$$

One possible estimation of Γ_g is based on stochastic delay bounds given by EDF schedulable conditions. Generally, a stochastic delay bound is expressed by $\mathbb{P}\{\text{delay} \geq d_g\} \leq \mathcal{D}_g$, where $\mathcal{D}_g$ is a function of d_g and other parameters (e.g., link capacity and traffic characteristics). Then let $\Gamma_i = \mathcal{D}_g$ to determine CAC conditions. Below such a CAC based on EBB [40], [41] is introduced.

The EBB traffic model is expressed as [38]

$$\mathbb{P}\{A[s,t] \geq (t-s)\rho + \sigma\} \leq \Lambda e^{-\alpha\sigma} \quad (13)$$

for all $\sigma > 0$, where $t > s \geq 0$ and $\rho > 0$ all are constants against t. Here, both α and Λ are constants against t, and ρ is the upper average rate of $A(t)$. An aggregate flow from two EBB flows is still of EBB. Let $\mathcal{G} = \{\rho_j, \Lambda_j, \alpha_j, d_j\}$ be set of groups ($j = 1, 2, ...G$), each of which is an aggregate flow from individual EBB flows with the same delay bound. Assume a group $g \in \mathcal{G}$ consist of total n_g EBB flows each with $\{\rho_{g,i}, \Lambda_{g,i}, \alpha_{g,i}\}$, $i = 1, 2, ...n_g$. Then with some algebra, this aggregate g-flow is still EBB with $\rho_g = \sum_{i=1}^{n_g} \rho_{g,i}$, $\Lambda_g = \sum_{i=1}^{n_g} \Lambda_{g,i}$ and $\alpha_g^{-1} = \sum_{i=1}^{n_g} \alpha_{g,i}^{-1}$.

An EDF schedulablility condition for EBB given by [40], [41] is (some notations here are different from those used in [41], and j or g indicate an aggregate flow from set $\mathcal{G}$)

$$\begin{aligned}\mathbb{P}\{D_g(t) \geq d_g\} \quad \leq \quad & \sum_{\hat{\tau}=0}^{t} \mathbb{P}\{\sum_{j\in\mathcal{G}} A_j[t-\hat{\tau}, t+d_g-d_j] + \\ & L_{max} > c(\hat{\tau}+d_g)\}, \end{aligned} \quad (14)$$

where $D_g(t)$ is the delay for g-packet at time t and L_{max} the maximum packet size. With discrete $\hat{\tau}$, $t-\hat{\tau}$ indicates the last time before t at which there is no traffic to be serviced before the tagged packet, i.e., g-packet here, in the queue. $A_j[s,t]$ indicates the amount of traffic arrival between s and t from flow j, which assumes traffic arrival happens only at discrete time points. That is, $A_j[s,t] = \sum_{m=s+1}^{t} A(m)$, where $A(m)$ is a discrete-time process for traffic arrival at time m. (14) can be briefly interpreted as follows: $\sum_{j\in\mathcal{G}} A_j[t-\hat{\tau}, t+d_g-d_j]$ indicates the total amount of traffic arriving from $t-\hat{\tau}$ to $t+d_g-d_j$ that is serviced before g-packet, where $t+d_g-d_j$ is the latest arrival time for j-packet to be serviced before g-packet. L_{max} is due to the untransmitted data remaining in the server. Therefore, the sum of the above two items is just the amount of data to be serviced before g-packet while $c(\hat{\tau}+d_g)$ is the maximum amount of data can be serviced before the deadline of g-packet.

Let $\mathcal{D}'_g$ denote the right hand of (14). Then a stochastic delay bound for an EBB flow with d_g at any time is [40], [41]

$$\mathcal{D}'_g \leq \Psi e^{-\omega(c-\sum_{j\in\mathcal{G}}\rho_j)d_g}, \quad (15)$$

where

$$\Psi = \frac{e^{-\omega(\sum_{j\in\mathcal{G}}\rho_j d_j - L_{max})}}{1-e^{-\omega(c-\sum_{j\in\mathcal{G}}\rho_j)}} \sum_{j\in\mathcal{G}} \Lambda_j. \quad (16)$$

The right hand of (15) is used as $\mathcal{D}_g$ in the following discussion.

B. A cost estimation for EBB

Here we use (15) to give an estimation of θ_g introduced in Section II-E by approximating $\hat{r}_g$ with ρ_g. According to the definition of $\hat{r}_g$, we only need to consider one group with (15), i.e., $G = 1$. Therefore,

$$\hat{r}_g \approx \rho_g = c + \frac{1}{\alpha_g}\ln[1 - \frac{\Lambda_g e^{-\alpha_g(cd_g - L_{max})}}{\Gamma_g}]. \quad (17)$$

Replacing $\hat{r}_{i(h)}$ in (9) with (17), we can have an estimation of θ_g for the CAC condition given by (15) as (here $i(h)$ is simplified into i)

$$\theta_g \approx \frac{(1-\Gamma_g)c}{d_g(c + \alpha_g^{-1}\ln[1 - \Lambda_g\Gamma_g^{-1}e^{-\alpha_g(cd_g - L_{max})}])^2}, \quad (18)$$

from which we can get the range of θ_g as $(\frac{1-\Gamma_g}{cd_g}, \infty)$, and

$$d_g > \frac{1}{c}[L_{max} + \frac{1}{\alpha_g}\ln\frac{\Lambda_g}{\Gamma_g(1-e^{-c\alpha_g})}] \quad (19)$$

to avoid situation $\theta_g = \infty$, and when $\Lambda_g = 0$ or $[\Lambda_g\Gamma_g^{-1}e^{-\alpha_g(cd_g - L_{max})}] \to 0$, $\theta_g \to (1-\Gamma_g)/(cd_g)$.

TABLE I

PER-HOP SERVICE, d_g AND θ_g ($\Gamma_g \approx 0$)

Service No.	hops 1,5 d_g (10^{-4})	hops 1,5 θ_g (10^{-7})	hops 2,4 d_g (10^{-5})	hops 2,4 θ_g (10^{-7})	hop 3 d_g (10^{-6})	hop 3 θ_g (10^{-7})
1	1	643	5	321.5	5	803.76
2	10	64.3	50	32.15	50	80.376
3	100	6.43	500	3.215	500	8.0376

TABLE II

END-TO-END: $\mathbf{D}$, θ_ϱ AND PER-HOP SERVICE COMBINATIONS

Ser no.	$\mathbf{D}(s)$ 10^{-4}	θ_ϱ 10^{-5}	per-hop i	j	k	Ser no.	$\mathbf{D}(s)$ 10^{-3}	θ_ϱ 10^{-6}	per-hop i	j	k
1	3.05	27.3	1	1	1	15†	10.7	130	1	3	3
2	3.5	20.1	1	1	2	16†	12	93.9	2	3	1
3	8	19.4	1	1	3	17†	12.1	21.5	2	3	2
4†	12	21.5	1	2	1	18	12.5	14.3	2	3	3
5	13	14.3	1	2	2	19†	20.1	146	3	1	1
6	17	13.6	1	2	3	20†	20.2	73.6	3	1	2
7	21.1	15.8	2	1	1	21†	20.6	66.4	3	1	3
8	22	8.52	2	1	2	22†	21	88.1	3	2	1
9	26	7.8	2	1	3	23†	21.1	15.8	3	2	2
10†	30	9.97	2	2	1	24	21.5	8.52	3	2	3
11	31	2.73	2	2	2	25†	30	82.3	3	3	1
12	35	2.01	2	2	3	26†	30.1	9.97	3	3	2
13†	102	21	1	3	1	27	3.05	2.73	3	3	3
14†	103	13.7	1	3	2						

Similarly, (10) is estimated by

$$\theta_\varrho \approx \sum_{\forall h \in \varrho} \frac{\Pi_{j=1}^{h}[1-\Gamma_{g(j)}]c_h}{d_{g(h)}}\{c_h + \frac{1}{\alpha_{g(h)}} \times \ln[1-\frac{\Lambda_{g(h)}}{\Gamma_{g(h)}}e^{-\alpha_{g(h)}(cd_{g(h)}-L_{max})}]\}^{-2}. \quad (20)$$

C. Numerical results for end-to-end unit cost θ_ϱ

Some numerical results on the CAC can be found in [40], [41]. So here we only discuss e2e service $[\mathbf{D}, \mathbf{F}]$ and θ_ϱ over a path ϱ. If unspecified otherwise, the units for c, d_g and L_{max} are *mbps*, *second* and *byte*, respectively. First, we look at the combination of per-hop services for a path consisting of five hops, namely $1 \sim 5$, with a hierarchical and symmetric c structure, i.e., OC-3 for both hops 1 and 5, OC-12 for both hops 2 and 4 while OC-48 for hop 3. Fig. 2 plots θ_g versus Γ_g for these c settings, each with three d_g services indicated therein. We can find that the difference in θ_g for those services is remarkable for the same c except in the case of $\Gamma_g \to 1$. As indicated by this figure, we can roughly divide Γ_g into four segments for each pair $[d_g, c]$ each corresponding to a narrow range of θ_g, i.e., $0 \sim 0.4$, $0.4 \sim 0.8$, $0.8 \sim 0.9$ and $0.9 \sim 1$. The last segment can be further splitted if necessary.

To simplify discussion with less e2e service combinations, here we ignore packet failure ratio by setting $\Gamma_g = 10^{-20}$ in (18). So each hop provides three services as listed in Table I. To further reduce the number of combinations for e2e services to be discussed, we also assume symmetric service selection in hops 1&5 and hops 2&4, i.e., the service selected from hop 1 is the same as that from hop 5, and the same for hops 2 and 4. Even in this case, there are still total 27 combined e2e services as listed in Table II. Note that, given c and T_{spec}, the shortest affordable d_g is limited by (19). $\mathbf{D}$ and θ_ϱ in Table II are calculated respectively by $2[d_{i(1)} + d_{j(2)}] + d_{k(3)}$ and $2[\theta_{i(1)} + \theta_{j(2)}] + \theta_{k(3)}$, where $i, j, k \in \{1, 2, 3\}$ indicate the services provided by hops 1&5, 2&4 and 3 (listed in Table I), respectively. This table is arranged according to $\mathbf{D}$ jointly with θ_ϱ in an ascending order.

As indicated in Table II, generally, θ_ϱ decreases as $\mathbf{D}$ increases except some of those with '†' indicating a service with a longer $\mathbf{D}$ failing in offering a lower θ_ϱ accordingly. For example, θ_ϱ of service 4 is higher than those of services 2∼3. It is nature for the user to avoid using such services if possible. In stead, a user can select the service providing $\mathbf{D}$ close to the e2e delay requirement ($\mathbf{D}_r$) of applications at the lowest cost. For example, an application with $\mathbf{D}_r = 3.05\ ms$ can select service 27 rather than service 1 whose cost is 100 times larger than the former's. However, whether or not a user's request can be granted by the network is still subject to the traffic load requiring the same service. Therefore, those with '†' may be still used in case. Note that the above service option cannot be easily provided by less granular services such as DiffServ, which so far one premium service for QoS guarantee.

TABLE III

$\mathbf{D}_r$, $\mathbf{D}$ AND θ_ϱ VERSUS PATH LENGTH

$\mathbf{D}_r$ (s)	length	$\mathbf{D}$ (s)	θ_ϱ	e2e service
$5{\times}10^{-4}$	5	$2.5{\times}10^{-4}$	$1.61{\times}10^{-4}$	1,1,1,1,1
	4	$2{\times}10^{-4}$	$1.29{\times}10^{-4}$	1,1,1,1
	3	$1.5{\times}10^{-4}$	$9.65{\times}10^{-5}$	1,1,1
	2	$1{\times}10^{-4}$	$6.43{\times}10^{-5}$	1,1
$1{\times}10^{-3}$	5	$7{\times}10^{-4}$	$1.32{\times}10^{-4}$	1,1,1,1,2
	4	$6.5{\times}10^{-4}$	$9.97{\times}10^{-5}$	1,1,1,2
	3	$6{\times}10^{-4}$	$6.75{\times}10^{-5}$	1,1,2
	2	$1{\times}10^{-3}$	$6.43{\times}10^{-6}$	2,2
$1{\times}10^{-2}$	5	$7{\times}10^{-3}$	$1.32{\times}10^{-5}$	2,2,2,2,3
	4	$6.5{\times}10^{-3}$	$9.97{\times}10^{-6}$	2,2,2,3
	3	$6{\times}10^{-3}$	$6.75{\times}10^{-6}$	2,2,3
	2	$1{\times}10^{-2}$	$6.43{\times}10^{-7}$	3,3

Now we return to the issue mentioned in Section I regarding the same application running over different paths measured in the number of hops that a path consists of. Here c for each hop is set to OC-12. Table III lists the minimum θ_ϱ of the offered services, i.e., $\mathbf{D}$, and per-hop service combination for e2e services, for three settings of $\mathbf{D}_r$ over paths ranging from 2 to 5 hops. As listed in this table, when $\mathbf{D}_r = 5 \times 10^{-4}$, each hop must offer service 1, which is the best service like PS in DiffServ and whose θ_ϱ is the maximum accordingly, to support such QoS requirement, no matter how long path is. When $\mathbf{D}_r = 10^{-3}$, one hop can just use service 2 (the secondly best service) instead of service 1 for paths longer than 2 hops while only service 2 from each hop is enough for the 2-hop path. For the latter, θ_ϱ is just 10% of the maximum cost offered by the best service. Again with DiffServ, there is no similar option available for such situation. This service option space given by DQS increases with $\mathbf{D}_r$ as indicated by θ_ϱ for $\mathbf{D}_r = 10^{-2}$ in this table, where θ_ϱ for the 2-hop path is just 1% of the maximum cost offered by the best service.

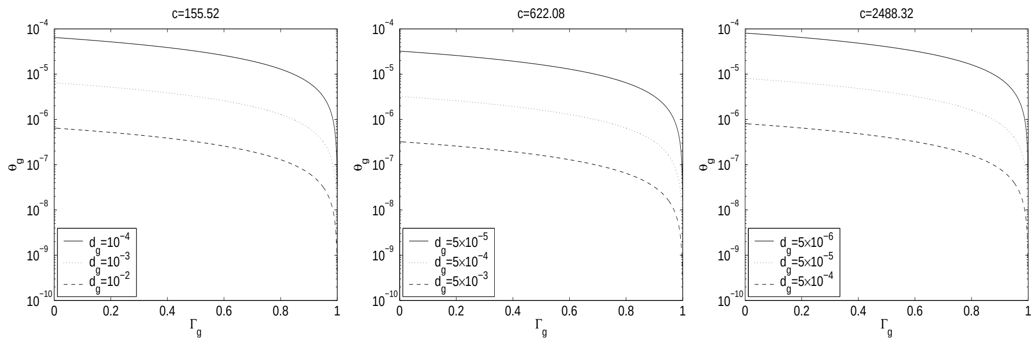


Fig. 2. θ_g versus Γ_g against d_g and c ($\alpha_g = 1, \Lambda_g = 10, L_{max} = 1000$)

IV. Summary

This paper discussed the structure of DQS, which tries to provide granular services for e2e QoS mainly relying on buffer control rather than output schedulers. DQS suggests to convert packet delay guarantee into packet loss ratio guarantee with either dropping or marking the packets whose delays are perceived unable to be guaranteed. As consequence, sophisticated output schedulers are not required to simplify implementation and improve scalability. A cost model for the differentiated services was also discussed. This model may be used to define pricing structure for differentiated service so that the network resource can be used properly to support various e2e QoS requirements. It also allows users to select or combine 'economic' services that can best satisfy their e2e QoS requirements with more service granularity.

The following issues required more studies. Without using sophisticated output schedulers, DQS mainly relies on CAC and packet queueing for packet failures (i.e., dropping due to lifetime expiration plus loss due to congestion) and delay guarantee. An ideal CAC should be able to satisfy various packet failure requirements with less packet-level intervention for simple implementation and less over-provisioned bandwidth to guarantee QoS for high bandwidth utilization. Regarding packet queueing, this operation should be fast enough so that the packet forwarding speed will not be affected. This part may be considered jointly with buffer admission control such as RED to reduce the overall implementation complexity. The discussion in this paper is based on the strict queueing policy, how the packet dropping precedence will affect the performance and how to use another queue for non-guaranteed packets to reduce the e2e packet failure are also important issues. In addition, simulation studies are also necessary to find relationship among traffic characteristics, CAC algorithms, queueing policies and e2e performances.

Acknowledgment

I would like to thank Dr. Yin Qinghe and Dr. Jiang Yuming for the fruitful discussion with them. I also take this opportunity to thank the anonymous reviewers for their patience and valuable comments as well as many recommendations.

References

[1] E.W. Knightly and N.B. Shroff, "Admission control for statistical QoS: theory and practice," *IEEE Network Mag.*, pp. 20–29, Mar./Apr. 1999.

[2] H. Zhang, "Service disciplines for guaranteed performance service in packet-switching networks," *Proc. The IEEE*, vol. 83, no. 10, pp. 1374–1396, Oct. 1995.

[3] V. Jacobson, K. Nichols, and K. Poduri, "An expedited forwarding PHB," *IETF RFC 2598*, Jun. 1999.

[4] B. Davie, A. Charny, J.C.R. Bennett, K. Benson, J.Y. Le Boudec, W. Courtney, S. Davari, V. Firoiu, and D. Stiliadis, "An expedited forwarding (PHB) per-hop behavior," *IETF RFC 3246*, Mar. 2002.

[5] A. Charny, J.C.R. Bennett, K. Benson, J.Y. Le Boudec, A. Chiu, W. Courtney, S. Davari, V. Firoiu, C. Kalmanek, and K.K. Ramakrishnan, "Supplemental information for the new definition of the EF PHB (expedited forwarding per-hop behavior)," *IETF RFC 3247*, Mar. 2002.

[6] J. Heinanen, F. Baker, W. Weiss, and J. Wroclawaski, "Assured forwarding PHB group," *IETF RFC 2597*, Jun. 1999.

[7] N. Benameur, S.B. Fredj, S. Oueslati-Boulahia, and J.W. Roberts, "Quality of service and flow level admission control in the Internet," *Computer Net.*, vol. 40, pp. 57–71, 2002.

[8] I. Foster, C. Kesselman, and S. Tuecke, "The anatomy of the grid: enabling scalable virtual organizations," *Int. J. Supercomputer Applicatins*, , no. 3, pp. 200–222, Jan. 2001.

[9] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An architecture for differentiated devices," *IETF RFC 2475*, Dec. 1998.

[10] I. Stoica and H. Zhang, "Providing guaranteed services without per flow management," in *Proc. ACM SIGCOMM*, Cambridge MA, USA, Sep. 1999.

[11] Z.L. Zhang, Z.H. Duan, L.X. Gao, and Y.W.-T. Hou, "Decoupling QoS control from core routers: a novel bandwidth broker architecture for scalable support of guaranteed services," in *Proc. ACM SIGCOMM*, Stockholm, Sweden, Aug. 2000.

[12] Y.M. Jiang, "LBFA: a stateless approach to scalable support of end-to-ed per-flow service guarantees," in *submission of INFOCOM'04*, Jul. 2003.

[13] R. Guérin, S. Kamat, V. Peris, and R. Rajan, "Scalable QoS provision through buffer management," in *Proc. ACM SIGCOMM*, Vancouver, Canada, Sep. 1998.

[14] C. Dovrolis, D. Stiliadis, and P. Ramanathan, "Proportional differentiated services: delay differentiation and packet scheduling," in *Proc. ACM SIGCOMM*, Cambridge MA, USA, Sep. 1999.

[15] J. Roberts and S. Oueslati-Boulahia, "Quality of service by flow-aware networking," *Phil Trans. R. Soc. Lond*, vol. A (2000) 358, no. 1773, pp. 2197–2207, 2000.

[16] V. Firoiu, X.H. Zhang, and Y. Gao, "Best effort differentiated services: trade-off service differentiation for elastic applications," in *Proc. IEEE Int. Conf. Telecom.*, Zagreb, Croatia, Jun. 2001.

[17] P. Hurley, K. Mourad, J.Y. Le Boudec, and P. Thiran, "ABE: providing a low-delay service within best effort," *IEEE Network Mag.*, vol. 15, no. 3, pp. 60–69, May 2001.

[18] Y.N.-E. Liu and W. Wong, "Deadline based channel scheduling," in *Proc. IEEE Globecom*, San Antonio TX, USA, Nov. 2001, vol. 4, pp. 2358–2362.

[19] Z.L. Zhang, Z.H. Duan, and Y.W.-T. Hou, "Fundamental trade-offs in aggregate scheduling," in *Proc. IEEE Int. Conf. Net. Protocols*, University of Minnesota, USA, Nov. 2001, vol. 4, pp. 129–137.

[20] B. Braden, D. Clark, J. Crowcroft, B. Davie, S. Deering, D. Estrin, S. Floyd, V. Jacobson, G. Minshall, C. Partridge, L. Peterson, K.K. Ramakrishnan, S. Shenker, J. Wroclawski, and L.X. Zhang, "Rec-

ommendations on queue management and congestion avoidance in the Internet," *IETF RFC 2309*, Apr. 1998.

[21] M.A. Weiss, *Data Structures and Algorithm Analysis in C*, The Benjamin/Cummings Publishing Company, Inc., 1992.

[22] J. Liebeherr and D. Wrege, "Design and analysis of a high-performance packet multiplexer for multiservice networks with delay guarantees," Tech. Rep. CS-94-29, University of Virginia, 1994.

[23] L. Kleinrock, *Queueing Systems, volume II: Computer Applications*, John Wiley & Sons, 1975.

[24] L. Georgiadis, R. Guerin, V. Peris, and K.N. Sivarajan, "Efficient network QoS provisioning based on per node traffic shaping," *ACM/IEEE Trans. Networking*, vol. 4, no. 4, pp. 482–501, Aug. 1996.

[25] R. Guérin and V. Peris, "Quality-of-service in packet networks: basic mechanisms and directions," *Computer Net.*, vol. 31, no. 3, pp. 169–189, Feb. 1999.

[26] A.W. Berger and W. Whitt, "Extending the effective bandwidth concept to networks with priority classes," *IEEE Commun. Mag.*, vol. 36, no. 8, pp. 78–83, Aug. 1998.

[27] A.W. Berger and W. Whitt, "Effective bandwidths with priorities," *ACM/IEEE Trans. Networking*, vol. 6, no. 4, pp. 447–460, Aug. 1998.

[28] A.I. Elwalid and D. Mitra, "Analysis approximation and admission control of a multi-service multiplexing system with priorities," in *Proc. IEEE INFOCOM*, Boston, USA, Apr. 1995, pp. 463–472.

[29] H.L. Vu and M. Zukerman, "Blocking probability for priorityclasses in optical burst switching networks," *IEEE Commun. Let.*, vol. 6, no. 5, pp. 214–216, May 2002.

[30] P.B. Key, "Resource pricing for differentiated services," http://research.microsoft.com/ ~peterkey/Papers/KIVSpbkey.pdf, Feb. 2001.

[31] P. Marbach, "Pricing differentiated services networks: bursty traffic," in *Proc. IEEE INFOCOM*, Anchorage Alaska, USA, Apr. 2001, vol. 2, pp. 650–658.

[32] C.A. Courcoubetis and A. Dimakis, "Providing bandwidth guarantees over a best-effort network: call-admission and pricing," in *Proc. IEEE INFOCOM*, Anchorage Alaska, USA, Apr. 2001, vol. 1, pp. 459–467.

[33] D. Mitra, K.G. Ramakrishnan, and Q. Wang, "Combined economic modeling and traffic engineering: joint optimization of pricing and routing in multi-service networks," in *Proc. ITC-17*, Salvador da Bahia, Brazil, Dec. 2002.

[34] S. Jordan and H. Jiang, "Connection establishment in high speed networks," *IEEE J. Selected Areas Commun.*, vol. 13, no. 7, pp. 1150–1161, Sep. 1995.

[35] H. Jiang and S. Jordan, "A pricing model for high speed networks with guaranteed quality of service," in *Proc. IEEE INFOCOM*, San Fransisco, USA, Mar. 1996, vol. 2, pp. 888–895.

[36] X. Wang and H. Schulzrinne, "Pricing network resoruce for adaptive applications in a differentiated services network," in *Proc. IEEE INFOCOM*, Anchorage Alaska, USA, Apr. 2001, vol. 2, pp. 943–952.

[37] L.A. DaSilva, "Pricing for QoS-enabled networks: a survey," *IEEE Commun. Surveys & Tutorials (on-line)*, Jan. 2000, DaSilva.pdf.

[38] O. Yaron and M. Sidi, "Performance and stability of communicatio networks via robust exponential bounds," *ACM/IEEE Trans. Networking*, vol. 1, no. 3, pp. 372–385, Jun. 1993.

[39] H. Michiel and K. Laevens, "Teletraffic engineering in a broad-band era," *Proc. The IEEE*, vol. 85, no. 12, pp. 2007–2033, Dec. 1997.

[40] E. Biton and A. Orda, "QoS provision with EDF scheduling, stochastic burtiness and stochastic guarantees," in *Proc. ITC-18*, Berlin, Germany, Aug./Sep. 2003, vol. 5b, pp. 1061–1070.

[41] E. Biton and A. Orda, "QoS provision with EDF scheduling, stochastic burtiness and stochastic guarantees (extented version)," http://comnet.technion.ac.il/~berez/BO03.pdf.

[42] D. Anick, D. Mitra, and M.M. Sondhi, "Stochastic theory of a data-handling system with multiple sources," *Bell Syst. Tech. J.*, vol. 16, no. 8, pp. 1871–1894, Oct. 1982.

A Methodology for Studying Persistency Aspects of Internet Flows

Jörg Wallerich★, Holger Dreger★, Anja Feldmann★,
Balachander Krishnamurthy◇, Walter Willinger◇

★Technische Universität München {hdreger,feldmann,jw}@net.in.tum.de *
◇AT&T Labs Research,Florham Park, NJ, USA {bala,walter}@research.att.com

ABSTRACT

We focus in this paper on Internet flows, consider their contributions to the overall traffic per time unit or bin, and perform a multi-scale and multi-protocol analysis to explore the persistency properties of those flows that contribute the most (also known as "heavy hitters" or "elephants"). Knowing the persistency features (or a lack thereof) of the heavy hitters and understanding their underlying causes is crucial when developing traffic engineering tools that focus primarily on optimizing system performance for elephant flows.

The main difficulty when studying the persistency properties of flows is that the available measurements are either too fine-grained to perform large-scale studies (i.e., packet-level traces) or too coarse-grained to extract the detailed information necessary for the purpose at hand (i.e., Netflow traces, SNMP). We deal with this problem by assuming that flows have constant throughput through their lifetime. We then check the validity of this assumption by comparing our Netflow-derived findings against those obtained from directly studying the corresponding detailed packet-level traces.

By considering different time aggregations (e.g., bin sizes between 1–10 minutes) and flow abstractions (e.g., raw IP flows, prefix flows), varying the definition of what constitutes an "elephant", and slicing by different protocols and applications, we present a methodology for studying persistency aspects exhibited by Internet flows. For example, we find that raw IP flows that are elephant flows for at least once (i.e., one bin or time unit) in their lifetimes tend to show a remarkable persistence to be elephants for much of their lifetimes, but certain aggregate flows exhibit more intricate persistency properties.

Keywords: Zipf's law, large flows, time-scales, flow-abstraction, elephants vs. mice, reservations, protocol use, Netflow

1. INTRODUCTION

This paper contributes to the nascent literature on characterizing the rates at which IP flows transmit data in the Internet [1, 2, 3]. In particular, we present the findings of an empirical study that demonstrate that Zipf's law for flow rates (i.e., number of bytes transmitted by the different flows during a given bin) holds not only for a fixed bin or time period, but applies more generally bin-by-bin[1] over time and for different flow abstractions. Recall that Zipf's law (see [4, 5] and references therein) for flow rates states that for a set of n inferred flow rates, ordered as $x_{(1)} \geq x_{(2)} \geq \ldots \geq x_{(n)}$, so we may think of $x_{(r)}$ as the rth-largest flow rate and of r as the flow rate's rank ($1 \leq r \leq n$), the relationship $rx_{(r)} = constant$ (or, more generally, $r^{\alpha}x_{(r)} = constant = c, \alpha > 0$) holds, at least approximately. When plotted on log-log scale, this *rank–size* relationship results in an (approximate) straight line with slope -1 or $-\alpha$, respectively, with the top-ranked flow rates being exceptionally large but rare and the lower-ranked rates being smaller but more common. This latter property follows directly from the *size–frequency* relationship that corresponds to Zipf's law and states that $f(x)$, the relative frequency of occurrence of a flow rate of size x satisfies the relation $f(x) = c \cdot x^{-2}$ (or, more generally, $f(x) = c \cdot x^{-(1+\alpha)}$), $x = 1, 2, \ldots$. To illustrate that Zipf's law applies bin-by-bin over time, Figure 1 shows the log-log plots of 10 size-rank relationships for flow rates of raw IP flows corresponding to 10 consecutive 1-minute bins. The plots are offset from one another by a small amount in the vertical direction to facilitate a visual assessment of Zipf's law across time, i.e., an approximate straight line behavior for each of the 10 plots. The same applies to other time periods, flow abstractions, and bin sizes (not shown). Flows whose flow rates appear in the top ranks (e.g., ranks 1–10) for at least one time interval or bin during their lifetime are referred to as "heavy hitters" or "elephants", while flows with consistently lowly-ranked flow rates (e.g., ranks beyond 100) are called "mice." We call flows exhibiting intermediate flow rates during their lifetime (e.g., ranks 11–100: never top-ranked, but not always lowly-ranked) "hybrids". In this paper we classify flows according to their ranks which are defined in terms of their absolute rate (e.g., greater than 1 Mbps) or

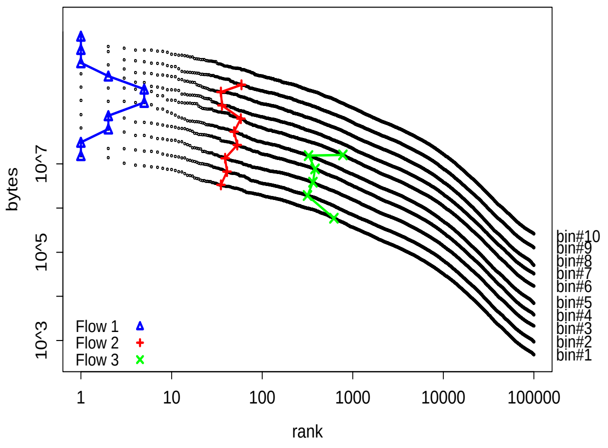


Figure 1: An illustration of Zipf's law across 10 successive time bins (raw IP flows, bin size = 1 minute) including three lines for three flows that connect the points on different size-rank curves.

*This work is partially funded by DFG, Project 1126

[1]Something applies bin-by-bin if it is applicable for each individual bin.

their relative rate (e.g., greater than 10% of the traffic). Focusing on ranks facilitates the comparison of flows across time periods because the number of elephant flows and hybrid flows are constant. It can also be expected to add an element of stability, especially when the emphasis is on investigating the temporal dynamics of flow attributes such as “being an elephant flow”.

Given that Zipf’s law for flow rates applies on a per-bin basis across time begs the question whether a flow that lasts for a number of bins and has been classified as “heavy hitter” has earned this distinction because of being top-ranked only sporadically (i.e., the flow rates associated with just one or two bins made it into the top ranks) or persistently (i.e., the flow rates in most bins are top-ranked) throughout much of its lifetime. To illustrate, reconsider Figure 1 which shows log-log plots of 10 size-rank relationships for flow rates of raw IP flows corresponding to 10 consecutive 1-minute bins. Included in this figure are also three lines that connect various points on the different size-rank curves. These lines indicate how the ranks of the flow rates of three particular raw IP flows that were active during (parts of) this 10-minute interval changed over time. Note that while the left line shows about the same amount of movement in the rankings as the right one due to the log-scale on the x-axis, the number of ranks covered is far greater for hybrid and mice flows than for the elephant flows, with no indication that these flows will ever become elephants.

Understanding the persistency properties of how much traffic individual Internet flows (especially the heavy hitters) contribute during their lifetime to the overall traffic is important for traffic engineering. A commonly-used approach in traffic engineering targets the large flows primarily and attempts to optimize system performance mainly for them. Equally important is the ability to identify the causes underlying any observed persistency properties of these large Internet flows. Examples that follow this basic approach to traffic engineering include, among others, [6] (measurement support and accounting), [7] (Web server overload control), [8] (routing), [9] (scheduling), etc. Clearly, such approaches are more viable and effective if a substantial portion of the overall traffic in a bin is due to a few heavy hitters and if roughly the same cast of heavy hitters is responsible for a significant amount of the total traffic across different bins.

Unfortunately, studying the persistency aspects of Internet flows and demonstrating that the findings are representative requires a compromise concerning the available measurements. On the one hand, carefully examining persistency-related aspects of Internet flows is only possible using detailed packet-level traces, which tend to be collected in only a few places and for limited durations only. On the other hand, Netflow traces are more widely available and are therefore more suitable for checking whether or not certain findings are representative. However, because Netflow traces are in general too coarse-grained for investigating a number of dynamic aspects of individual flows, empirical studies relying on Netflow data often make the critical assumption that raw IP flows exhibit constant throughput (computed as flow size divided by flow duration) for the duration of their lifetime.

We thus use a combination of packet-level and Netflow measurements. We first rely on packet traces and Netflow traces derived from the same packet traces and check whether the assumption necessitated by the nature of the available Netflow data—that flows have constant throughput throughout their lifetime—is valid. If this assumption is invalid we may arrive at misleading or wrong conclusions about the persistency properties of Internet flows. We then use some large Netflow traces to illustrate the kind of persistency properties of measured Internet flows that are largely insensitive to the assumption of constant throughput.

Using our methodology for the available data, our initial findings depict a wide, yet largely unexplored spectrum of persistency-related behavior of Internet flow rates. For example, we find that heavy hitters at the level of raw IP flows show remarkable persistence and are likely to be top-ranked for the duration of their entire life. Thus, at the level of raw IP flows, the notion “once an elephant, always an elephant” holds with relatively high probability and suggests a simple heuristic for identifying heavy hitters—pick the top-N in each bin and “adjust” for those that last only one or so bins. Despite this persistence property of the heavy hitters, there seems to exist an unavoidable tradeoff between the need to consider a large number of top flows to account for a substantial portion of the overall traffic and the desire to account for only a few heavy hitters to ensure strong persistency properties. However, a formal statement concerning this tradeoff is beyond the scope of this paper. We also observed that while heavy hitters at the aggregate level are often made up of heavy hitters at the level of raw IP flows, we also encountered numerous instances where aggregate elephants contain essentially no raw IP flow elephants, but consist almost exclusively of raw IP flow mice.

The rest of the paper is organized as follows. After a brief discussion of related work in Section 2, Section 3 describes our proposed methodology for studying persistency-related aspects of Internet flows. In Sections 4 and 5 we describe the data sets that are used throughout the paper and validate our approach. Some of our initial findings are described in Section 6, where we focus in particular on the observed persistency properties of Internet flows under different time aggregation (i.e., different bin sizes) and different flow abstractions. We conclude in Section 7 by summarizing our experience and suggesting future research directions.

2. RELATED WORK

A common observation found in many measurement studies is that the sizes of raw IP or aggregated flows obey a Zipf-type law in the sense that a small percentage of the flows accounts for a large percentage of the total traffic (e.g., [8, 10, 11, 12] and references therein). As far as flow rates are concerned, a number of recent papers have attempted to characterize them and determine the causes of the observed rates at which flows transmit data in the Internet. In particular, [1] provides indirect evidence that a static version of Zipf’s law for flow rates holds. In fact, Fig. 1 in [1] can be considered to express Zipf’s law, with a bin size that corresponds to the length of the underlying trace data. In [2], the authors claim that a single high-rate flow typically accounts for much of the burstiness of the aggregate link traffic, and their Fig. 1(c) in support of this claim shows Zipf’s law for flow rates (on linear-linear scale) for a single bin. While these and other papers make important contributions and improve our understanding of the nature of Internet flows and flow rates, none of them dwell on Zipf’s law as such or on whether it holds on a per-bin basis across time.

To our knowledge, the first attempt at assessing the feasibility of identifying and isolating heavy hitters for traffic engineering purposes is reported in [13]. In this paper, the authors propose a definition of heavy hitters that accounts for both their volume and their persistence in time, and they rely on packet-level traces and corresponding BGP tables to examine the effectiveness of their proposed classification schemes for a fixed flow abstraction (determined by the BGP destination network prefixes) and different time aggregation (bin sizes of 1, 5, and 30 minutes). They find that while a single-feature classification scheme is impractical due to the large number of short-lived elephant flows it produces, a simple two-feature classification scheme that accounts for short transient dips or bursts is more successful in identifying the persistent heavy hitters. Our work adds to the original findings discussed in [13] by further exploring the persistence property of heavy hitters and by considering a fuller range of useful flow abstractions. At the same time, we move beyond the issues addressed in [13] by presenting an approach that is equally applicable to a collection of the top N flows (e.g., N=1000) as it is to the top-10 and that allows for a systematic investigation of potential causes underlying the collective behavior of groups of flows that deviates from "normal" or "typical" behavior (e.g., can "unusual" behavior be associated with standard or "emerging" applications?). However, perhaps the most important original contribution of our work is that it attests to the viability of traffic engineering approaches that trade off precise but scarce measurements (i.e., exact per-bin flow rates from fine-grained packet traces) for approximate but abundant information (i.e., constant throughput assumption for coarse-grained Netflow traces).

3. METHODOLOGY

When exploring the various persistency aspects of Internet flows, we divide time into bins and the basic information consists of the number of bytes or packets transmitted by the different flows in each bin. There are two methodological aspects to our work. First, we require detailed packet-level traces that can be easily and efficiently aggregated in time and across flows. Second, we rely on an effective multi-scale and multi-protocol analysis of the available data, where multi-scale refers to an ability to analyze the data at different time aggregations (i.e., bin sizes) and different flow abstractions, while multi-protocol implies the flexibility to slice the data by different protocols and applications.

In terms of flow abstractions, the packets belonging to a flow are usually determined by two parameters: *aggregation* and *timeout* [14, 15, 16, 17, 18]. Earlier work (e.g., [17]) has shown that the specific choice of the timeout parameter rarely changes the basic characteristics of the resulting flows. However, the aggregation parameter has a significant impact. To illustrate, *raw IP flows* are defined by the protocol they use as well as by source and destination IP addresses and port numbers. *Prefix flows* are defined by the source and destination prefix. The prefix for both the source and the destination IP address is computed from the IP address by using the mask from the longest prefix match in the routing table. If such a mask is not available we compute the prefix from the source and destination IP address using a fixed-length mask of length L. Note that a larger value of L implies a smaller degree of abstraction or aggregation.

To investigate temporal persistency aspects of Internet flows and their behavior across different time scales, we consider different bin sizes. To cover a reasonable range of interesting time scales, our choice of bin sizes ranges from 60 seconds to $120, 240, 480$ and 960 seconds. Since 60 seconds is significantly larger than the typical round-trip times experienced in the Internet [1], the impact of TCP-specific features (e.g., ACK spacing, window effects) on the resulting flow rates can be expected to be minimal and decrease further as the bin size increases.

Concerning the ability to restrict our analysis to specific applications and/or protocols, we note that UDP and TCP are likely to show different features. Similarly, different applications are known to have different signatures, e.g., distribution of flow length. Unfortunately the only way to associate flows with applications is via port numbers which is somewhat problematic.

3.1 Software Infrastructure

The methodology described above centers around computing per-flow throughput for different bin sizes, flow abstractions, protocols, and applications. To this end, based on the kind of input data, the first step consists of creating appropriate flows (see Section 3.2). Next, the resulting flows may have to be filtered by protocol and/or application before they can be used to form flows at different levels of aggregation. Depending on the input data and the desired abstraction level, one or more of the previous steps can be skipped. Relying on well-formed flows as input, the third step consists of computing the per-bin ranking of all the flows based on their per-bin throughput. Since we are mainly interested in the heavy hitters, we reduce the computational effort by considering only the top 5000 per-bin flow rankings and enter them into a database. This database contains all the information necessary for our study of the (non-)persistency of Internet flows across time, abstraction levels, protocols, and applications. For more details on the software infrastructure see Appendix A.

3.2 Input Data

Basically there are three kinds of network measurement data sources available. SNMP [19] data is the most available but unfortunately also the most coarse-grained data source and thus unsuited for our purposes. Another common data source are packet level traces. These provide sufficiently detailed information, but are not easily available from ISP backbones, or they usually cover only short timespans, e.g. a few hours. The third data source are flow-based measurements like *Cisco's Netflow* [20]. Flow-based measurements provide an acceptable level of detail for our purposes, providing byte counts and duration per flow and are more easily available than packet traces, especially for longer traces on ISP backbones. However, they pose a problem for our purposes, in that the time granularity of Netflow data is too coarse and requires approximations which in turn may lead to inaccuracies in the analysis and ultimately to incorrect conclusions. Cisco's NetFlow system is designed for traffic monitoring and accounting. Packets passing router interfaces on which NetFlow is enabled are aggregated in real time to unidirectional flows defined by source/destination IP and ports, protocol, and IP TOS bits. For each flow NetFlow gathers among others byte and packet counts, start and end timestamps as well as routing related information such as prefix mask lengths and AS numbers. Each router terminates flows according to a set of heuristics and then exports them to a NetFlow collector. A flow is terminated if it had no contributing packets for a certain amount of time (default 15 seconds). Flows that have been active for more than a certain amount of time (default 30 minutes) are terminated. This ensures that online-monitoring tools can operate on current information. Furthermore TCP flows are terminated whenever a FIN or RST packet is found. Finally, when the router runs short of memory, flows are terminated using undisclosed heuristics.

Since we are interested in per-bin packet or byte counts for each flow, the very nature of the available Netflow records makes it necessary to somehow distribute the total packet or byte aggregates per flow across the flow's lifetime. A natural choice is to assume

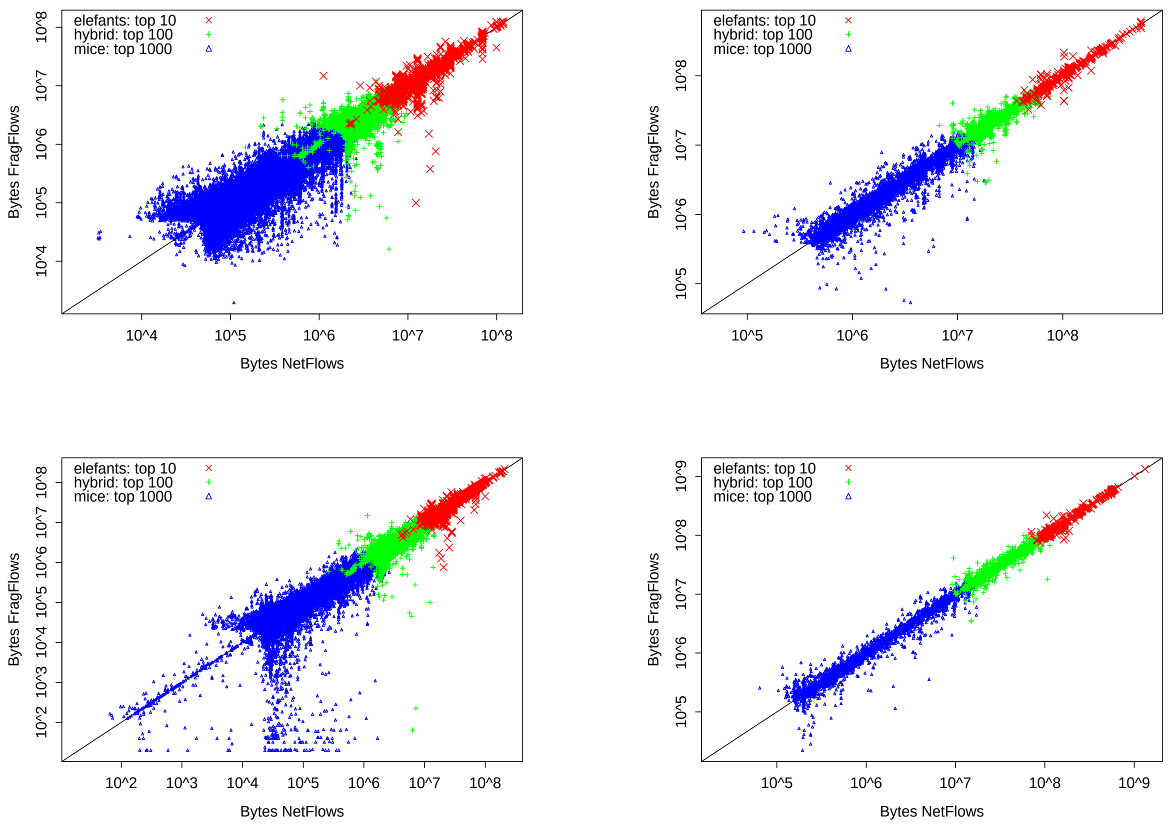


Figure 2: Scatterplots comparing the per-bin byte counts for NetFlows and FragFlows (top: raw IP flows, bottom: aggregated destination prefix flows using a fixed 16 **bit mask; left: bin size =** 60 **sec; right: bin size =** 480 **sec).**

a constant flow rate, computed as flow volume divided by flow duration. Practical experience with accounting and visualization systems suggests that this is a reasonable assumption, especially for reasonably large aggregation levels [21]. This is also consistent with the results of Barakat et al. [22] who model Internet traffic at the flow level via a Poisson shot noise process where the shape of the "shot" can be rectangular which corresponds to the assumption of constant rate. Additional complications with Netflow records are that NetFlow relies increasingly on sampling (see also Section 6.1) and that it uses several different timeout values. A router may expire a flow at any point if it needs to reclaim memory resources. The result of this process is that one flow may be split into several *raw flows*.

In this paper, we use both packet-level traces and raw Cisco Netflow traces. For the packet-level traces we reconstruct the individual flows and compute the corresponding exact per-bin rates called "fragments." We call the resulting flows *FragFlows*–they are defined in terms of the per-bin fragments which can vary across a flow's lifetime. For the Netflow traces we first recombine *raw flows* into flows and then compute their rates under the constant flow rate assumption. The resulting flows are called *NetFlows*–their per-bin flow rates are constant for the duration of a flow. Note that using packet-level traces, it is possible to reconstruct the appropriate Netflow counterparts that incorporate the constant flow rate assumption. These are also called *NetFlows*. In general the term *NetFlows* refers to flows whose per-bin rates are constant as a result of the constant flow rate assumption while *FragFlows* is synonymous with flows whose per-bin rates are computed exactly and are not likely to be constant.

Flows at different aggregation levels can be derived from both *FragFlows* and *NetFlows*. For an aggregate flow, its per-bin flow rates are simply the sum of the corresponding per-bin rates of those flows that make up the aggregate flow; for *FragFlows*, taking the sum involves the exact per-bin flow rates, while for *NetFlows*, the per-bin sum is taken over the average rates of the flows that are active during that bin and are part of the aggregate flow.

4. TRACES

We had access to several hour-long packet-level traces from the external Internet connection at the Universität des Saarlandes (UNI) and Leibnitz Rechenzentrum München (EDU). Both connections provide Internet access to a major university, some colleges, and several research institutes. The capacity of the UNI link was 155 Mbps, and the capacity of the EDU link was 622 Mbps. In both locations the recording was done via the monitoring port of a Gigabit Ethernet switch just before the traffic passes the last router to the Internet. In terms of Cisco Netflow traces we had access to several days worth of traces collected at different backbone routers of a Tier-1 ISP.

Throughout this paper we use the following data sets. The packet trace P1 was gathered at the EDU location using a packet filter that captured only traffic to and from the CS department. The trace consists of a total of 349,194,384 packets (more than 10.3 GB of compressed data) and was collected on Friday, Oct. 31, 2003, 11:17-15:22. A second data set P2 was gathered at the same location, but without the packet filter used for P1 (i.e. this trace contains all packets that crossed the monitored link). Trace collection started on Wednesday, Nov. 13, 2002, 19:10 and ended on Thursday morning at 02:43. The trace consists of 344,965,957 packets or more than 11 GB of compressed data. The third trace P3 consists of 104,583,280 packets (some 2.5 GB of compressed data) and was collected at the UNI location on Tuesday, Feb. 02, 2003, between 12:00–15:29.

The Cisco Netflow trace F1 was collected from a single backbone router within a Tier-1 ISP. The trace contains a day worth of Netflow data, collected on Dec. 11, 2001. The data set F1 contains over 211 million flow records or about 4 GB of compressed data. This Netflow trace is unsampled and has a loss rate of 9% (mostly due to the capacity limits in the monitoring infrastructure, not in the ISPs infrastructure). A second day-long sampled Netflow trace F2 was collected on Sept. 5, 2002 and consists of almost 330 million flow records or more than 5.1 GBytes of compressed data.

5. VALIDATION

Before embarking on a Netflow-based study of the persistency aspects of Internet flows, we first examine in this section the validity of the constant flow rate assumption. As discussed earlier, this assumption is unavoidable when using Netflow data in this context. To this end, we rely on our packet-level traces for which we can derive both FragFlows as well as NetFlows. After obtaining the per-bin rankings for the top 5000 flows for both NetFlows and FragFlows at various abstraction levels, as well as for different protocols and applications, we match the appropriate FragFlows and NetFlows and compare them to assess the impact of the constant throughput assumption on the validity and quality of our findings. More precisely, we select the top 1000 entries from each bin and located the matching counterpart if it existed among the top 5000 entries.

5.1 Per-bin byte differences

We start by comparing how the per-bin byte counts of the FragFlows differ from those of the NetFlows for different time aggregations and flow abstractions. The working hypothesis is that we should expect differences, but that they will diminish as we consider larger bin sizes and/or flow aggregates. In this context, one of the objectives is to try and identify the causes of and quantify to some extent the expected differences for small bin sizes and raw IP flows.

The impact of the constant flow rate assumption for NetFlows can be expected to be most dramatic when comparing how many bytes a FragFlow is contributing to a particular bin and how many bytes the corresponding NetFlow contributes. To illustrate this comparison, Fig. 2 shows scatterplots of the per-bin contributions of NetFlows (x-axis, log-scale) against the per-bin contributions of the corresponding FragFlows (y-axis, log-scale) for the trace P1. The top row is for raw IP flows and bin sizes of 60 seconds (left) and 480 seconds (right), while the bottom row is for aggregated destination prefix flows using a fixed 16 bit mask and the same two bin sizes. Note that the plot only shows distinct points; duplicates are removed before plotting. We use different symbols to indicate the ranking that a particular point is associated with. A small "$\triangle$" corresponds to a byte count that has one ranking (FragFlow or NetFlow) in the top 1000 but none in the top 100. A "+" marks those byte counts that have one ranking in the top 100 but not top 10, and a "$\times$" identifies the byte counts that have a top 10 ranking. The most pronounced feature in all of these plots is a strong concentration of the points around the diagonal, with varying degrees of deviation as we consider different bin sizes and/or flow abstractions.

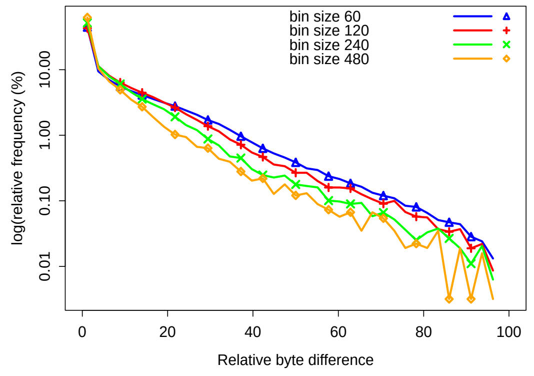


Figure 3: Histogram plot of the relative byte differences between FragFlows and NetFlows with linear x- and logarithmic y-axis (raw IP flows, bin size = 60**,** 120**,** 240**,** 480 **sec).**

In relative terms, these deviations from the diagonal seem to be smallest for byte counts with a top 10 ranking and tend to get larger as we consider more of the more frequently occurring lower-ranked byte counts. Also, as we consider either larger bin sizes (left to right in Fig. 2) or larger aggregation levels (top to bottom), or a combination of larger bins and aggregates (top left to bottom right), the concentration along the diagonal is accentuated. As far as increasing the bin size is concerned, one explanation for this observation is that more flows will completely fall within a single bin, and that this feature impacts not only the many lowly-ranked flows whose durations tend to be shorter than those of the few top-ranked flows. In terms of increasing the level of flow aggregation, the variability of the per-bin byte counts of large aggregates is bound to decrease as predicted by the Central Limit Theorem.

To quantify the degree of (in)accuracy of the approximation resulting from the constant flow rate assumption, we compute for each per-bin byte count the relative byte difference between the FragFlow and the NetFlow entries. This is done for each bin and flow by taking the absolute value of the difference between the two byte counts, dividing it by the maximum of those two values, and multiplying by 100 to get percentages. Fig. 3 shows histogram plots of the relative byte differences for different bin sizes and illustrates that the quality of the constant throughput assumption increases with bin size. Similar conclusions can be drawn when considering the same histogram plots (not shown here) for different flow aggregation levels. Overall we observe that—as expected—the accuracy of using the more widely available NetFlows instead of the hard-to-come-by FragFlows increases with bin size and with aggregation level, implying that the constant throughput assumption may be appropriate for certain flow abstractions.

While concentration around the diagonal in the plots in Fig. 2 is highly desirable, points that clearly deviate may also be informative, especially if they concern top-ranked byte counts, and deserve closer inspection[2]. For example, in the top left plot in Fig. 2, we can identify 18 bins associated with 16 flows where the difference

[2]An obvious artifact in Fig. 2 are the vertical bands that are associated with one and the same NetFlow byte count and result from having in general many different FragFlow byte counts for the same flow.

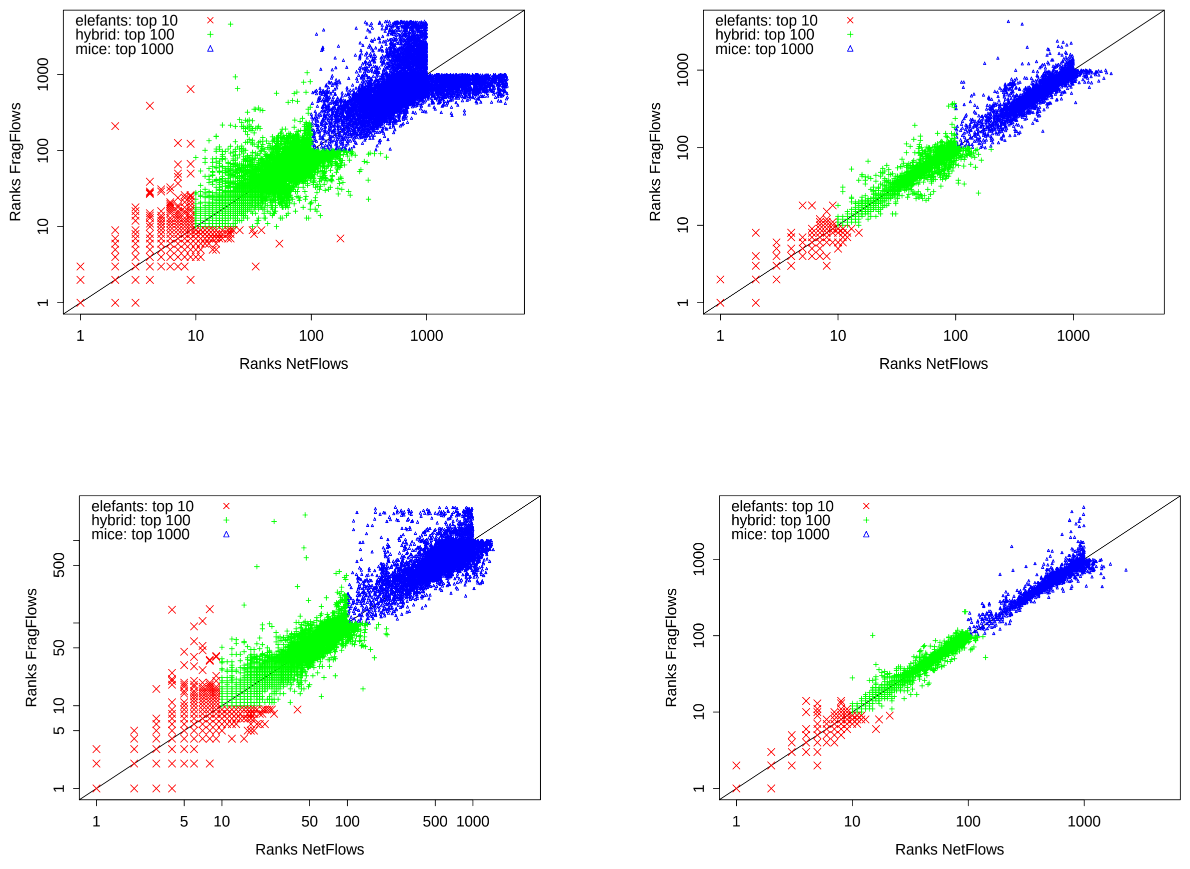


Figure 4: Scatterplots comparing the per-bin byte count ranks for NetFlows and FragFlows (top: raw IP flows, bottom: destination prefix flows using a fixed 16 **bit mask; left: bin size =** 60 **sec; right: bin size =** 480 **sec).**

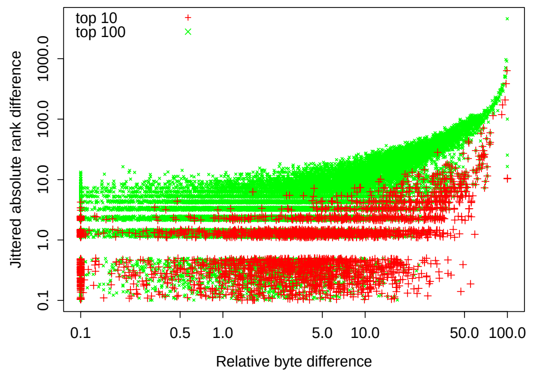
in FragFlow- vs. NetFlow-derived byte counts was large enough to cause the byte counts to be classified as top 10 for FragFlow and as top 100 for NetFlow, or vice versa. Of these 18 "outliers", 14 occurred at the start (6) or the end (8) of the flows. Possible explanations include: TCP slowstart and initial protocol overhead for those at the beginning, and timeout effects for those at the end.

5.2 Per-bin rank differences

Next we examine what impact the observed differences in per-bin byte counts that are the result of the constant throughput assumption have on the per-bin ranking of the flows.

While the previous subsection focused on the impact of the constant flow rate assumption on byte counts, we now examine the differences that are imposed by this assumption on the ranking. The raw rank data derived from the P1 trace are given in Fig. 4 which shows scatterplots of the per-bin NetFlows-derived ranks (x-axis, log-scale) against the corresponding FragFlows-derived ranks (y-axis, log-scale). As in Fig. 2, the top row is for raw IP flows and bin sizes of 60 seconds (left) and 480 seconds (right), while the bottom row is for aggregated flows using a fixed 16 bit mask and the same two bin sizes. The symbols have the same meaning as in Fig. 2, but note that now, the top-ranked per-bin flow rates are concentrated in the lower left rather than in the upper right corners of the four plots. After accounting for the artifacts caused by selecting only the top 1000 entries and by using log-scale in conjunction with ranks that can only take discrete values, the common dominant feature in these plots is again a pronounced concentration of the points around the diagonal, with some obvious deviations. However, these deviations from the diagonal diminish significantly as either larger bin sizes or larger flow aggregates are considered.

Before addressing the issue how the constant flow rate assumption impacts the ranking of individual flows, we first examine how much of a per-bin rank difference can be expected as a result of a given per-bin byte count difference. In effect, in answering this question, we combine the information from Figs. 2 and 4 to generate Fig. 5. More precisely, to generate the relevant information, we consider the relative per-bin byte count differences instead of their absolute values because generally, for elephants, larger absolute byte changes are needed to switch ranks than for mice. At the same time, in terms of rank differences, it seems more sensible to consider absolute rather than relative rank changes. To be able to examine the relative byte differences in conjunction with the smaller absolute rank differences in more detail, we manipulate the data by adding a constant offset of 0.1 to both the absolute rank differences as well as to the relative byte differences; we also introduce some jitter to the absolute rank differences by adding a uniform random amount between 0 and 0.4 to each absolute rank difference so as to avoid the situation that all points with the same (integer-valued) rank difference appear as a single point in the plot. The resulting two plots (one for raw IP flows and a bin size of 60 sec, and one for aggregated flows using a fixed 16 bit mask and a bin size of 480 sec) are shown in Fig. 5 and correspond to the top left and bottom right plots shown in Figs. 2 and 4. The clearly visible band with (jittered) rank differences between 0.1 and 0.5 cor-



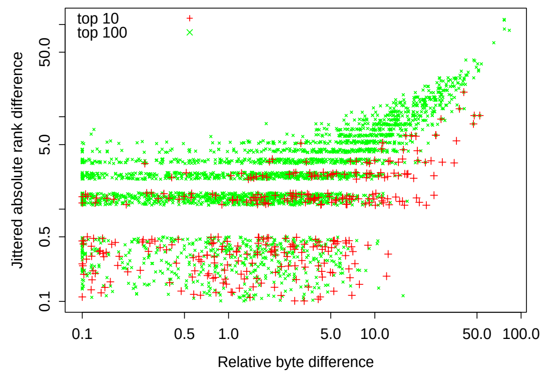
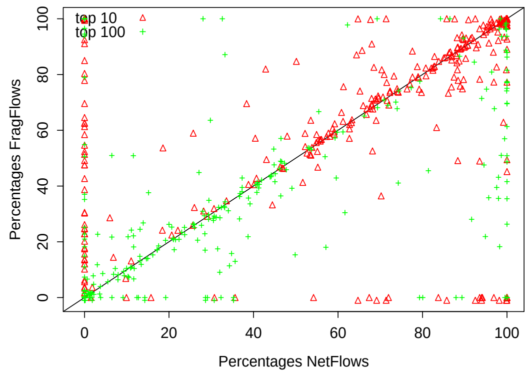


Figure 5: Scatterplots of the relative per-bin byte count differences between FragFlows and NetFlows vs. jittered absolute per-bin rank differences between FragFlows and NetFlows (top: raw IP flows, bin size = 60 **sec; bottom: aggregated destination prefix flows using a fixed** 16 **bit mask, bin size** = 480 **sec).**



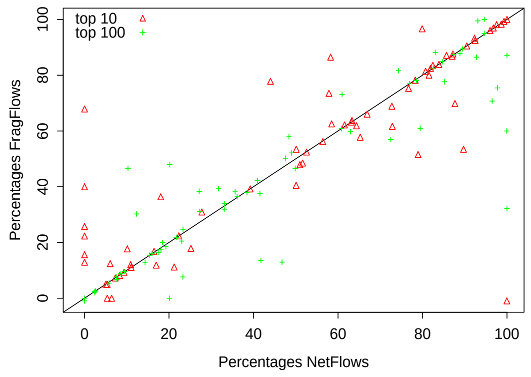


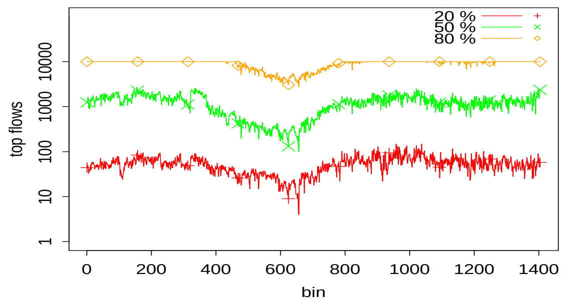
Figure 6: Scatterplots of percentages of bytes from elephant/hybrid bins for NetFlow-derived elephants/hybrids vs. percentages of bytes from elephant/hybrid bins for FragFlows-derived elephants/hybrids (top: raw IP flows, bin size = 60 **sec; bottom: aggregated destination prefix flows using a fixed** 16 **bit mask, bin size** = 480 **sec).**

responds to those bins where the FragFlows and the corresponding NetFlows entries have the same rank. It is interesting to note that in the non-aggregated case (top plot), rather large relative byte difference (up to 50%) can occur without influencing the rank too much. Once we include the next few discernible bands corresponding to rank differences of ± 1, ± 2, to ± 5 or so, the number of top-ranked bins is drastically reduced, more so for the aggregated case (bottom plot) than for the non-aggregated example. The remaining elephant bins are the ones where a large relative byte difference leads to a relatively large rank change. Fig. 5 begs the question how a flow rate can more or less keep its rank in going from NetFlows to FragFlows even if the byte count difference is relatively large. There are (at least) two arguments that can be put forward. For one, the byte difference may not be big enough to either reach the byte count of the next higher ranked entry or let it drop below the next lower ranked entry. Alternatively, another flow that was lower or higher ranked than the current one has a large relative byte difference and is therefore now ranked higher or lower than the current one. The latter argument also explains why a flow may change its rank in a bin even if the byte difference is zero or extremely small. Similar comments apply when considering different bin sizes and/or flow aggregation levels. Fig. 5 also illustrates that because the byte differences between the individual ranks are much smaller for the lower-ranked entries, the observed rank differences for the latter will be larger than for the top-ranked items.

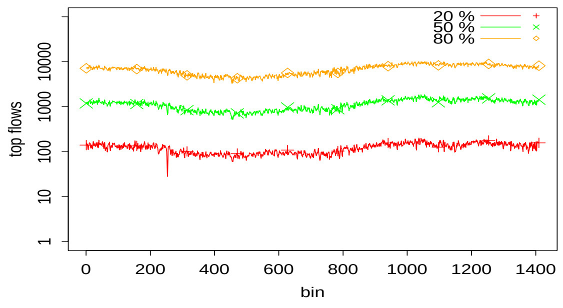
5.3 Per-flow rank differences

Until now, we have been mainly concerned with the per-bin byte count differences between FragFlows and NetFlows and with their impact on the resulting per-bin rank differences. Here we will use the insight gained so far at the per-bin level and apply it to determine the impact that the constant throughput assumption has on the flows as a whole. To this end, we focus on the heavy hitters, where we define a heavy hitter to be a flow that is ever ranked an “elephant” (i.e., in the top-10) in any bin during its lifetime. Other choices of defining an elephant (e.g., in the top-5, or top-20) yield similar results. For such flows we are interested in determining what percentage of the total bytes contributed by an “elephant” flow can be attributed to bins that are ranked within the top-10 or the top-100. Accordingly, a flow is considered a “hybrid” flow if its top ranked bin is a “hybrid” (i.e., in the top-100, but not in the top-10).

Fig. 6 shows two scatterplots of the percentage of bytes contributed by an elephant or hybrid during bins that were ranked within the top-10 (for elephants) or top-100 but not top-10 (for hybrids) using NetFlows (x-axis) against the same FragFlow-derived quantity (y-axis). The top plot deals with the non-aggregated case (i.e., raw IP flows and a bin size of 60), and the bottom plot is for the aggregated case (i.e., aggregated flows using a fixed 16 bit mask and a bin size of 480 sec).

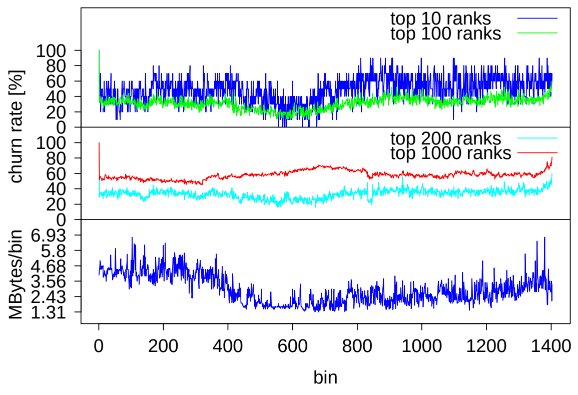


a) non-sampled

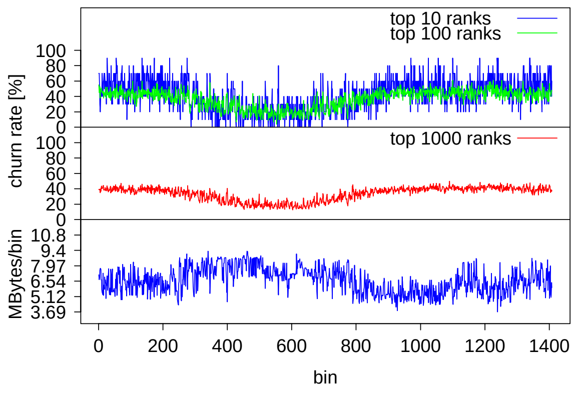


b) sampled

Figure 7: Number of top-ranked flows needed to account for a given portion of the total traffic per bin (raw IP flows, bin size = 60 **sec).**



a) non-sampled



b) sampled

Figure 8: Upper and middle parts: Churn rate processes associated with the top 10, top 100, top 200, and top 1000 flows, respectively. Lower part: Time series of flow rates needed for a newly arriving flow to move into the top 10. (Raw IP flows, bin size = 60 **sec.)**

Fig. 6 shows a number of informative properties related to relying on NetFlows as compared to FragFlows. For one, most of the points in both plots scatter around the diagonal, some are right on the diagonal (indicating a perfect match between NetFlows and FragFlows), some occupy the line $x = 0$ (vertical line through 0) and others the line $y = 0$ (horizontal line through 0). Looking first into the 24 (total points in the top plot is 524) flows satisfying $y = 0$, we find that the median distance of the FragFlow and the NetFlow ranking for the bins that cause each of these flows to be considered an elephant is 1 (mean is 1.8). This suggests that NetFlow just barely overestimated the ranking in comparison with FragFlow. Unfortunately, these edge effects cannot be avoided whenever one chooses a simple static elephant classification such as top-10 ranking, but time aggregation helps in alleviating this problem. Next the 51 flows satisfying $x = 0$ have little to do with edge effects, but represent in some sense the price one has to pay when using NetFlows instead of FragFlows in classifying Internet flows. Indeed, the reason for this drastic mismatch in this case between NetFlows and FragFlows is that while FragFlow is capable of capturing the dynamics of the within-flow data exchange, these details are invisible to NetFlows. To illustrate, a sample sequence of per-bin ranks for a FragFlow is: 135, 9, 11, 7, 15, 18, 18, 19, 17; the Netflow-derived per-bin rank sequence for that same flow is: 92, 13, 12, 15, 15, 18, 18, 19, 17. One consequence of NetFlows "mis-classifying" some elephant bins as hybrid bins is that the affected flows tend to get a large percentage of their bytes from hybrid bins when the actual (FragFlow-derived) percentage is in fact smaller. This explains the set of hybrid flows clustering around the line $x = 100\%$. In general, this problem can be alleviated with flow aggregation, which illustrates yet again that aggregation is the proper tool for achieving a desirable degree of accuracy when using NetFlows instead of FragFlows .

6. MULTI-SCALE/PROTOCOL FLOW ANALYSIS

The findings reported in the previous section justify the use of the widely available but coarse-grained unsampled Netflow data in conjunction with the constant flow rate assumption for studying persistency related aspects of Internet flows, especially of the heavy hitters. While the constant bandwidth assumption inherently gives raise to inaccuracies and errors, they can often be controlled by using aggregation and focusing on the large flows. Using the available large Netflow data sets, we illustrate in this section the kind of multi-scale and multi-protocol flow analysis that is intended to shed light on various persistency related aspects of Internet flows. In Section 6.1 we start out using both unsampled as well as sampled NetFlow traces and illustrate that there are qualitative differences in the results. Because of this observation and since the previous sections only justify the use of unsampled NetFlow traces, in the remaining part of this paper we focus exclusively on unsampled traces and leave the detailed exploration of the impact of sampling for future work. Our findings are largely qualitative, but they are also representative to the degree that we confirmed them using other traces. These latter data sets have also been used to validate on a case-by-case basis some of the findings presented below.

6.1 On the dynamics of flow rankings

Informally, Zipf's law and its variations are often interpreted as 80-20 or 90-10 rules, which state that some 80% (or 90%) of consequences stem from some 20% (or 10%) of causes. In the present context, this translates into "a significant portion of the total number of bytes in a bin is due to a relatively small percentage of the top-ranked flows (i.e., flows with the highest flow rates). Relying on the unsampled Netflow trace, F1, and the sampled Netflow trace, F2, Fig. 7 considers raw IP flows and 1-minute bins and shows how many of the top-ranked flows are needed for each bin to account for 20%, 50%, and 80% of the bin's total traffic volume. For example, we note that the top 100 or so flows account for some 20% of the total traffic per bin, the top 1,000 or so flows are responsible for about 50%, and to account for 80% of the total traffic, we need to consider way more than the top 1,000 flows (e.g., there are times that require more than the top 10,000 flows). We note that the relative bytes per top ranked flows for the unsampled Netflow trace, F1, is larger than those for the sampled Netflow trace, F2. Yet for the lower ranked flows they are less. Also note that the overall per bin volume of the unsampled trace is about a factor 2-3 smaller than for the sampled trace.

Fig. 7 leaves open the possibility that the cast of top-ranked flows can vary considerably from one bin to the next; that is, due to the arrivals of new and the departure of existing flows, there always exist opportunities for newly arriving flows to make it into the top ranks and for existing flows to fall out of the top ranks. Note that for raw flows, whose rates are by definition constant throughout their lifetimes, the arrival/departure dynamics of flows is the only ingredient that can cause instability among the top-ranked flows (i.e., significant rank changes from one bin to the next). In contrast, aggregate flows can also change ranks as a result of fluctuations in their rates from one bin to the next due to the arrival/departure dynamics of their constituent flows.

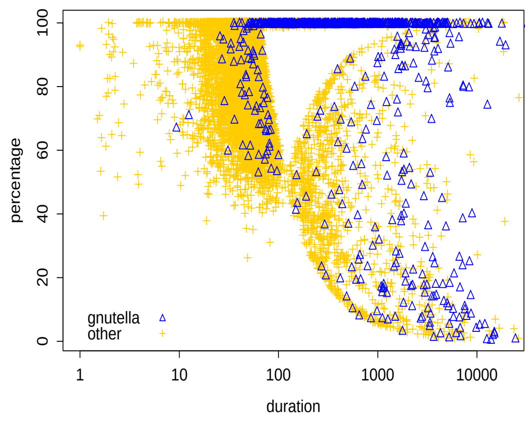
Fig. 7 also leaves open the possibility that the dynamics is only due to churn; that is, there is no persistency of flows across bins and the observations automatically follow from the well-known heavy-tailed distribution of flow sizes or lengths. However, a simple comparison of the flow length distributions shows that this is not the case for the top ranked flows. For example, for the trace F1 the median of the flow length distribution for 60 second bins increases from 4.2 seconds for the top 10000 flows to 44.7 for the hybrid flows and to 57.8 seconds for the elephant flows. In the case of 480 second bins, the larger bin length improves the ability of longer but lower average rate flows to be higher ranked. In fact, for the same trace, the median of the flow length distribution changes from 44.7 seconds to 136.0 seconds for hybrid and from 57.8 to 296.5 seconds for elephants. The maximum flow length coincides with the trace duration.

To illustrate the degree of (non-)persistency among the top-ranked flows over time, Fig. 8 considers raw IP flows and 1-minute bins again and shows the "churn rate" among the top 10, top 100, and top 1000 flows, respectively. Here, for each bin, the churn rate is defined to be the percentage of top 10 (top 100, top 1000) flows in that bin that were not among the top 10 (top 100, top 1000) flows in any of the previous bins. Thus, a high churn rate is an indication of significant non-persistency among the top-ranked flows, while a low churn rate reflects a considerable degree of persistency among the cast of top-ranked flows in time. In the upper and middle parts of Figures 8 a) and 8 b) we can see that while the different churn rates are roughly comparable, they show subtle but nevertheless important differences. Overall, the variability of the churn rate drops as one considers more ranks. For F1 (Figure 8 a)), the churn rate among the top 100 and top 200 ranks is lower than the churn rate for the top 10 ranks. During the early morning hours (bins 500-900 or so) this difference is reduced. The churn rate for the top 1000 ranks shows the opposite behavior. It is larger than the churn rate for the other ranks and it increases as the traffic volume decreases (not shown). This indicates that most flows that are in the top 1000 but not the top 200 are short lived and interchangeable. The byte volume differences between flows at rank 1000 are in the order of 10s of bytes. For F2, the churn rate among the top 100 and top 1000 ranks is slightly lower than the churn rate for the top 10 ranks during the early AM hours (bins 1–300 or so), it is slightly higher during the later AM hours/early PM hours (bins 300–900 or so), and appears to revert to the early AM behavior during the late PM hours (bins 900–1440). For this trace the churn rate for top 1000 ranks does not show the opposite behavior as for F1 since the byte volume differences between flows at rank 1000 are in the order of 100s of bytes.

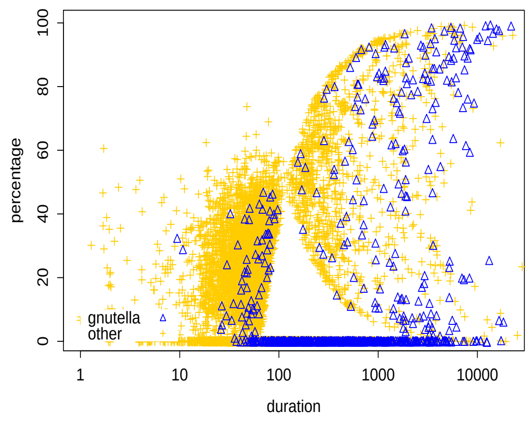
For both, F1 and F2, the churn rate appears to be correlated with the total number of flows, see Figure 7. However during the less busy periods in F1, a flow needs to contribute a smaller number of bytes to a bin in order to be top ranked than during the corresponding periods in F2. For a visual assessment of this observed behavior of the churn rates, we show in the lower parts of Fig. 8 the time series representing for each bin the rate (i.e., number of bytes per bin) at which a newly arriving flow would have to send data to move into the top 10 ranks (i.e., be classified as heavy hitter). The differences observed in the plots in Fig. 8 resulting from the use of unsampled vs. sampled Netflow traces clearly warrants further investigations. For the rest of this paper we focus on the unsampled Netflow trace F1.

Given the persistency behavior of Internet flows suggested by Fig. 8, we next focus on the heavy hitters, where we define a heavy hitter as in Section 5 and ask whether or not heavy hitters have a distinct persistency property. Put differently, we are interested in whether "once an elephant" implies "always an elephant", at least with high probability. Evidence of such persistency properties for the largest Internet flows is crucial for approaches to traffic engineering that rely on the persistence in time of flows to remain elephants. For the trace F1 with a bin size of 1 minute and when considering raw IP flows, we extracted a total of 5,666 heavy hitters and show in Fig. 9 scatterplots of the heavy hitters' lifetimes against the percentage of time they were ranked elephants (ranks 1-10; left plot) or "hybrids" (ranks 11-100; right plot); we use log-scale for their lifetimes on the x-axis and linear scale for percentages on the y-axis. To avoid certain artifacts due to binning when computing the percentage of time flows were ranked elephants/hybrids, for flows that start during some bin and subsequently cover one or more full bins, the time spent in the beginning partial bin is counted towards the flow's ranking in the first full bin; ending partial bins are handled similarly.

Fig. 9 reveals a number of interesting features as far as the heavy hitters are concerned. Ignoring for the time being the coding of the points, we first note that about $1/2$ of the heavy hitters are elephants



a) Heavy hitters as Elephants



b) Heavy hitters as Hybrids

Figure 9: Scatterplots of lifetimes of heavy hitters (log-scale on x-axis) against relative amount of time spent as elephants (left plot) or hybrids (right plot) for NetFlow trace F1.

during their entire lifetime (i.e., out of a total of 5,666 points, some 2,875 fall on the $y = 100\%$ line). Second, heavy hitters who are alive for 2 or more bins and are not elephants during their whole lifetime have about a 40% chance to be elephants for more than half their lifetime and a 60% chance to be elephants for less than half their lifetime (i.e., half-moon shaped cluster starting at $x = 120$ sec and $y = 50\%$). Finally, when comparing the left and right part of figure 9, the anti-symmetry between heavy hitters as elephants and heavy hitters as hybrids is not an accident. In fact, the right plot shows some 60% of the heavy hitters are never hybrids, and those heavy hitters that are alive for 2 or more bins and are hybrids for some time have about a 60% chance to be hybrids for less than half of their lifetime. Note that the remaining structure in the left upper (left lower) corner of the left (right) plot of Fig. 9 is relatively uninteresting since those points correspond to heavy hitters that are alive for less than 120 seconds (2 bins)[3]. In summary, Fig. 9 shows that more than 95% of the heavy hitters are elephants for more than half of their lifetime. A breakdown of all the heavy hitters by application is easily possible, but simply shows the usual suspects (e.g., web, nntp, p2p, ftp, and others) and individually, they produce plots similar to the ones shown in Fig. 9. To illustrate, we use in Fig. 9 the symbol "△" to denote heavy hitters associated with the well known p2p application Gnutella.

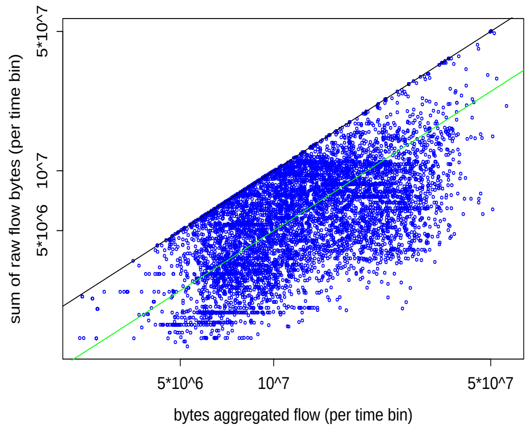
6.2 Heavy hitters and time aggregation

Part of the multi-scale aspect of our flow analysis involves considering different time scales and performing the same type of persistency study across a range of time scales. Our analysis (not shown here) suggests that the observations reported in Section 6.1 are largely invariant under different choices of bin sizes and hold in a genuinely multi-scale fashion. To explain this property, note that heavy hitters at the level of raw IP flows and at large time scale tend to remain heavy hitters at finer time scales. In fact, considering for example coarse scale to mean an 8-minute bin size and fine scale to mean a 4-minute bin size, a raw IP flow that is a heavy hitter at coarse scale will simply re-distribute the bytes in each large bin evenly among the two corresponding 4-minute bins at the finer time scale and is thus likely to cause the resulting flow to be a heavy hitter at the finer time scale. The situation is slightly more complicated for heavy hitters at the aggregate level (e.g., prefix flows), because their contributions to a large bin are generally no longer distributed evenly among the corresponding smaller bins at the finer time scale. Nevertheless, a preliminary analysis of the aggregate heavy hitters across a limited range of time scales (from a few seconds to hundreds of seconds) shows that heavy hitters tend to be invariant under time (dis)aggregation which explains why certain persistency properties associated with heavy hitters can already be gleaned from an analysis at coarse time scales which generally involves a substantially reduced data set and is therefore faster and more efficient.

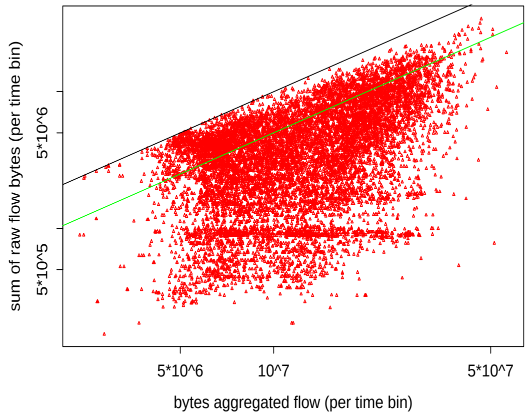
6.3 Heavy hitters and flow aggregation

Another aspect of our multi-scale analysis of Internet flows concerns aggregation in IP or flow abstraction. That is, Netflow data lends itself naturally to different levels of aggregation, from raw IP flows (defined by source and destination IP addresses and port numbers and protocol) to prefix flows (defined by source and destination prefix) to AS flows (defined by source and destination AS). To illustrate that flow aggregation is in many ways more intricate than time aggregation, we explore for trace F1 in Fig. 10 the question whether or not flows that are heavy hitters at some coarse scale of flow aggregation (e.g., prefix flows) are in general made up of constituents that are heavy hitters at a finer scale of flow abstraction (e.g., raw IP flows). That is, what is the observed behavior of heavy hitters under flow (dis)aggregation? To this end, for a 1-minute bin size, Fig. 10 shows scatterplots of the per-bin contributions of the heavy hitters at the aggregate level (destination-prefix, log-scale on x-axis) against the sum of the per-bin contributions of those flows that were elephants (top row), hybrids (middle row), and mice (bottom row), respectively, at the level of raw IP flows (log-scale on y-axis). We observe that while there are a number of points between the two lines $y = x$ and $y = x/2$ in the top and middle rows, the vast majority of points in the bottom row are concentrated in that area. That is, while many of the aggregate heavy hitters are largely (e.g., more than 50%) made up of raw IP flows

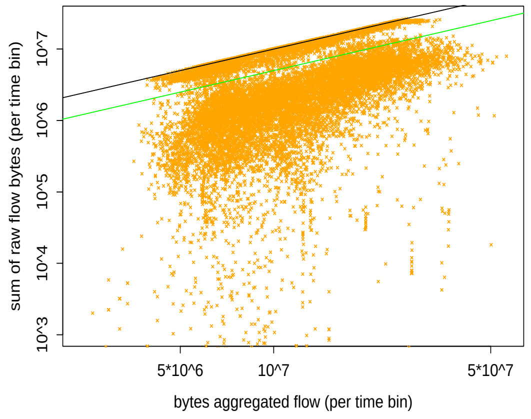
[3]Flows that last less than 120 seconds and are elephant for only part of their lifetime are split across two bins. Since they are more likely to be elephants in the bin in which they spend most of their time it is not surprising that most of the percentages are larger than 50%.



a) Elephant Contributers, non-sampled



b) Hybrid Contributers, non-sampled



c) Mice Contributers, non-sampled

Figure 10: Scatterplots of flow rates of aggregate heavy hitters (log-scale on x-axis) against aggregate contributions of the constituent raw IP flow elephants (top), hybrids (middle), and mice (bottom) for non-sampled Netflow trace F1 **(Bin size = 1 minute).**

that are elephants and hybrids, respectively, the bottom row shows a large number of instances where the aggregate heavy hitters are almost exclusively the result of raw IP flows that are mice. It would be interesting to explore this finding further and check for example whether or not the latter aggregates correspond to particular Web servers and whether or not the former can be associated with particular applications, but we leave such questions for future work.

7. SUMMARY

Using a combination of packet-level and Netflow traces, we examined the quality and accuracy of results obtained from relying on Netflow data (including the widely-assumed constant flow rate assumption) instead of packet-level data for studying various aspects of Internet flows. Subsequently, we focus on a largely unexplored facet of Internet traffic analysis related to Zipf's law for IP flow rates as a function of time and to the persistency properties of Internet flows, especially of the large flows or "elephants". Several applications motivate us ranging from reservation to traffic engineering. Our examination allows us to make a number of interesting observations. First, for all practical purposes, using the widely available but relatively coarse-grained Netflow traces vs. the scarcely recorded but very detailed packet-level traces for studying properties of Internet flows is justified. Errors and mismatches due to the constant bandwidth assumption underlying the use of Netflow traces are always a concern, but can in general be significantly reduced by aggregation (time and/or flow aggregation) or by focusing on the heavy hitters, and should be dealt with on a case-to-case basis. Second, at the level of raw IP flows, elephants tend to stay elephants for a very large portion of their lifetime and mice rarely move beyond their category; at the level of aggregate flows, the persistency properties tend to be more intricate due to a richer set of possible causes for variations under time or flow aggregation. Our software allows for examination of such phenomena in isolation (raw flows), as well as a variety of aggregations (prefixes) and de-aggregations (traffic partitioned into component protocols), and varying time scales.

The work presented in this paper identifies a number of issues that deserve further attention. For the purpose of reducing the volume of Netflow measurements, the more recently collected Netflow traces represent sampled data. As observed, sampling is yet another source of potential inaccuracy that needs to be dealt with when checking the quality and accuracy of findings that rely on sampled Netflow traces. Clearly, the validation approach presented in this paper can be extended to handle sampled Netflows. Another direction for future work is motivated by a number of the plots presented in this paper, each of which identifies its own set of "interesting" flows (i.e., clearly identifiable "outliers") that beg for a full-blown use of our proposed methodology, including a "drilling down" into the protocol and/or application-specific aspects that can be gleaned from the data. Such a detailed multi-scale and multi-protocol analysis (including a systematic study of prefix-based flows) is part of future work. Furthermore "joining" this kind of data with other relevant measurements such as appropriate BGP routing tables promises additional insights.

Acknowledgments

We would like to thank AT&T Labs Research for providing access to Netflow data and Leibnitz Rechenzentrum, München for their help in collecting packet traces.

8. REFERENCES

[1] Y. Zhang, L. Breslau, V. Paxson, and S. Shenker, "On the characteristics and origins of Internet flow rates," in *Proc. ACM SIGCOMM*, 2002.

[2] S. Sarvotham, R. Riedi, and R. Baraniuk, "Connection-level analysis and modeling of network traffic," in *Proc. ACM Internet Measurement Workshop*, 2001.

[3] S. Uhlig and O. Bonaventure, "Implications of interdomain traffic characteristics on traffic engineering," Tech. Rep. Infonet-TR-2001-08, University of Namur, Belgium, 2001.

[4] G. Zipf, *Human Behavior and the Principle of Least Effort*. Addison-Wesley, 1949.

[5] M. Mitzenmacher, "A brief history of generative models for power law and lognormal distributions.," *Internet Mathematics*, 2003.

[6] C. Estan and G. Varghese, "New directions in traffic measurement and accounting," in *Proc. ACM SIGCOMM*, 2002.

[7] J. Jung, B. Krishnamurthy, and M. Rabinovich, "Flash Crowds and Denial of Service Attacks: Characterization and Implications for CDNs and Web Sites," in *Proc. of the World Wide Web Conference*, 2002.

[8] W. Fang and L. Peterson, "Inter-AS traffic patterns and their implications," in *Proc. of the 4th Global Internet Symposium*, 1999.

[9] L. Guo and I. Matta, "The war between mice and elephants," in *Proceedings of ICNP*, 2001.

[10] A. Feldmann, A. Greenberg, C. Lund, N. Reingold, J. Rexford, and F. True, "Deriving Traffic Demands for Operational IP Networks: Methodology and Experience," in *Proc. ACM SIGCOMM*, 2000.

[11] K. C. Claffy and N. Brownlee, "Understanding Internet traffic streams: Dragonflies and Tortoises.," *IEEE Communications*, 2002.

[12] E. Kohler, J. Li, V. Paxson, and S. Shenker, "Observed structure of addresses in IP traffic.," in *Proc. ACM Internet Measurement Workshop*, 2002.

[13] K. Papagiannaki, N. Taft, S. Bhattacharyya, P. Thiran, K. Salamatian, and C. Diot, "A pragmatic definition of elephants in Internet backbone traffic.," in *Proc. ACM Internet Measurement Workshop*, 2002.

[14] K. C. Claffy, H.-W. Braun, and G. C. Polyzos, "A parameterizable methodology for Internet traffic flow profiling," *IEEE Journal on Selected Areas in Communications*, 1995.

[15] K. Thompson, G. J. Miller, and R. Wilder, "Wide-area Internet traffic patterns and characteristics," *IEEE Network Magazine*, 1997.

[16] S. Lin and N. McKeown, "A simulation study of IP switching," in *Proc. ACM SIGCOMM*, 1997.

[17] A. Feldmann, J. Rexford, and R. Caceres, "Efficient policies for carrying Web traffic over flow-switched networks," *IEEE/ACM Transactions on Networking*, 1998.

[18] P. Newman, G. Minshall, and T. Lyon, "IP switching: ATM under IP," *IEEE/ACM Transactions on Networking*, 1998.

[19] W. Stallings, *SNMP, SNMPv2, SNMPv3 and RMON 1 and 2*. Addison-Wesley, 1999.

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

[21] R. Sommer and A. Feldmann, "NetFlow: Information loss or win?," in *Proc. ACM Internet Measurement Workshop*, 2002.

[22] C. Barakat, P. Thiran, G. Iannaccone, C. Diot, and P. Owezarski, "A flow-based model for Internet backbone traffic," in *Proc. ACM Internet Measurement Workshop*, 2002.

APPENDIX

A. SOFTWARE COMPONENTS

Processing a very large number of flows into bins and aggregating and ranking them is a non-trivial exercise. Given the exploratory nature of the work, we decided to have an evolvable intermediate format that can help us answer questions quickly but retain enough information in order to look for alternate explanations. Note that different slices on the data require different parts of the flow records; e.g., per-application analysis requires port numbers and protocol information, while others require source and destination prefixes.

We divided the task into four components: restitching of the flows, binning, aggregation, and ranking. We used *unsampled* Netflow [20] records, ordering them temporally, and extracted the following fields: source and destination IP address/port number/AS number/prefix mask length, time fields, number of bytes/packets, protocol, and TCP flags. Netflow records usually come directly from a router. For the purpose of evaluation our software is capable of generating Netflow records as well as FragFlow records from detailed packet traces. Both constitute inputs for the binning step.

The base data structure used for the restitching of flows consists of a splay tree with orderings based on flow start time and a hash table using a flow key composed of source/destination IP address/port and protocol for fast flow information lookup. A flow can be accessed either by its key or the start time giving us the necessary flexibility for our analysis. Netflow records with the same key that overlap in time or follow each other within an inactivity timeout period (default 15 seconds) are merged into a single flow. In addition, associated information such as bytes and packets are summed, flags are `OR`ed, start time is set to be minimum of the two times etc. We process the flows using a sliding window data structure that covers all the active flow records while limiting the number of flows that need to be kept in memory. Really long flows hinder in moving the window forward. Thus, as a temporary backup procedure, we write to disk. In the future we will migrate to a merge-sort procedure to obviate the use of disk.

The binning phase uses the number of bytes contributed to the bin as a ranking key. The bins are ordered by bin start times. Raw flows are represented via a priority queue while aggregated flows are stored in a splay tree (since their ranks change) which contains all raw flows with the same aggregation key. Source (destination) prefix aggregation is done based on flows that share source (destination) prefixes with same source (destination) prefix mask lengths. If both source and destination prefixes match we call this prefix aggregation. We can also aggregate at a particular prefix mask length (e.g., 16, 24 etc.) or at source/destination AS level. We use a segment abstraction to partition aggregated flows using an inactivity timeout just as we do for raw flows. The segment abstraction aids in comparing duration of raw and aggregated flows.

Since all the bytes (packets) from a flow may not fall into complete bins we keep track of fractions of bytes (packets) that fall into incomplete bins and use average flow bandwidth to determine the fraction. The ranking of raw flows for each bin is done using standard priority queue insertion of flows with highest priority for ranks with the smallest number of bytes. This allows periodic culling of the flows with the least weight to limit the size of the priority queue. The aggregated flows are ranked by adding byte contributions to appropriate segments. Culling is harder since another flow belonging to an aggregated flow may arrive at some later time and with its contribution changes the priority of this aggregated flow. Their ranking is obtained by extracting flows in reverse order from the priority queue (ascending order of number of bytes). For aggregated flows, we remove excess flows while obtaining the ranking.

The resulting information is stored in a DBMS (PostgreSQL), which allows for highly flexible data handling. For example, we can derive more detailed information like the raw flows that are the contributors of an aggregated flow.

About 12,000 lines of C code forms the code base (5000 of which are shared as libraries for the entire system and the remaining ones are split roughly equally between restitching, binning, and flow collection). 1200 lines of shell, Perl and SQL scripts are used to load data into the DBMS and extract information from it. S-plus is sed to generate the plots. The entire process is automated with a variety of supplied parameters, such as flow time out, binning-size, rank count, aggregation level, prefix mask length etc. Restitching takes the bulk of the time and once that is complete different bin sets can be constructed in parallel. Filters are specified as dynamically loadable shared library modules. Output is separated into a file for ranking and a flow lookup table, with the latter containing per-segment information for aggregated flows. Extending the software to handle a new aggregation type requires addition of a simple comparison function for looking up a matching aggregated flow.

Measuring the Evolution of Transport Protocols in the Internet [*]

Alberto Medina
BBN Technologies
amedina@bbn.com

Mark Allman, Sally Floyd
ICSI Center for Internet Research
{mallman,floyd}@icir.org

ABSTRACT

In this paper we explore the evolution of both the Internet's most heavily used transport protocol, TCP, and the current network environment with respect to how the network's evolution ultimately impacts end-to-end protocols. The traditional end-to-end assumptions about the Internet are increasingly challenged by the introduction of intermediary network elements (middleboxes) that intentionally or unintentionally prevent or alter the behavior of end-to-end communications. This paper provides measurement results showing the impact of the current network environment on a number of traditional and proposed protocol mechanisms (e.g., Path MTU Discovery, Explicit Congestion Notification, etc.). In addition, we investigate the prevalence and correctness of implementations using proposed TCP algorithmic and protocol changes (e.g., selective acknowledgment-based loss recovery, congestion window growth based on byte counting, etc.). We present results of measurements taken using an active measurement framework to study web servers and a passive measurement survey of clients accessing information from our web server. We analyze our results to gain further understanding of the differences between the behavior of the Internet in theory versus the behavior we observed through measurements. In addition, these measurements can be used to guide the definition of more realistic Internet modeling scenarios. Finally, we present several lessons that will benefit others taking Internet measurements.

Categories and Subject Descriptors

C.2.2 [**Computer-Communication Networks**]: Network Protocols; C.2.3 [**Computer-Communication Networks**]: Network Operations

General Terms

Measurement, Design, Reliability, Standardization, Verification

Keywords

TCP, middleboxes, Internet, evolution

1. INTRODUCTION

In this paper, we investigate the evolution of TCP [45], the Internet's most heavily used transport protocol, on its own and in the context of ongoing changes to the Internet's basic architecture. As part of this work, we study the ways in which so-called "middleboxes" (firewalls, NATs, proxies, etc.) — which change the Internet's basic *end-to-end principle* [47] — impact TCP. We seek to elucidate unexpected interactions between layers and ways in which the Internet differs from its textbook description, including the difficulties various real-world "gotchas" impose on the evolution of TCP and end-to-end protocols in general. The measurements presented in this paper also serve as lessons for efforts that wish to further evolve end-to-end protocols and the Internet architecture.

While the Internet's architecture, protocols and applications are constantly evolving, there is often *competing evolution* between various network entities. This competing evolution can impact performance and robustness, and even halt communications in some cases. For instance, [41] shows that when setting up a TCP connection to a web server, attempting to negotiate the use of Explicit Congestion Notification (ECN) [46] interfered with connection establishment for over 8% of the web servers tested in 2000 (but down to less than 1% of the web servers tested for this paper in 2004). For such web servers, the client can only establish a TCP connection by re-attempting the connection without negotiating ECN usage. The connection failures in the presence of ECN negotiation were caused by firewalls configured to interpret the attempt to negotiate ECN as the signature of a port-scanning tool [25]. On the one hand, these firewalls can be seen as incorrectly associating new functionality with one of the first appearances of that new functionality in an undesirable application. On the other hand, the firewalls can also be seen as doing their job of blocking unwanted traffic. This example shows the fundamental problem of different evolution paths that can cross to the detriment of smooth traffic flow on the Internet.

Internet research is driven by simulations, experiments, analysis, and deployment studies designed to address particular problems in the Internet. However, the design of effective and accurate network models is challenging due to the intrinsic complexity of the Internet and the dynamic nature of the elements composing it. Researchers need better models of networks and protocols to ground their investigations, such that they can provide practical benefit on the evolving network [28]. Therefore, a second component of our work assesses the current deployment status of various proposed TCP algorithmic and protocol modifications and updates the literature with respect to the capabilities of a "modern" TCP stack. This will help us learn about TCP as it is actually deployed in the Internet, and aid researchers in accurately conducting future evaluations of the network and proposed changes.

In this paper, we bring both active and passive measurement techniques to bear to study web traffic in the context of the above stated issues. We use extensive active measurements to assess the

[*]This material is based in part upon work supported by the National Science Foundation under Grant Nos. 0205519 and 0230921. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

capabilities and algorithms used by web servers (the primary data senders in web transactions). Data senders are ultimately in control of TCP's congestion control and reliability algorithms. Therefore, our active measurements are focused on studying which congestion control algorithms, loss recovery schemes and options are implemented and how the interaction with today's evolving network environment influences the correctness and performance behavior of actual web servers.

These active measurements use and extend the TCP Behavior Inference Tool (TBIT) from [41], revising and rerunning the earlier tests on Selective Acknowledgment (SACK) and ECN capability, Reno vs. NewReno, initial congestion windows, and proper window halving after a loss.[1] The tests show that over the last four years, almost all web servers tested still appropriately halve their congestion window after a packet loss (Table 8); most web servers tested are still not ECN-capable (Table 3); the fraction of tested web servers that are SACK-capable increased from 41% in 2001 to 68% in 2004 (Table 7); and the fraction of successfully-tested web servers that use NewReno instead of Reno loss recovery with a non-SACK receiver has increased from 42% in 2001 to 76% in 2004 (Table 6).

We have also added a range of new active measurement tests exploring Path MTU Discovery, the effects of IP and TCP options on the TCP connection, the content of received SACK blocks, the congestion window increase during slow-start, the response to one or two duplicate acknowledgements, congestion window increases in the face of a receive window limitation, effective RTO values, and more. Tables 11 and 12 at the end of the paper give a summary of the results of these tests.

We also conducted passive measurements of the capabilities and limits imposed by web clients (the primary data receivers). Although data receivers do not directly control the data flow on a TCP connection, clients can optionally provide information to the data sender to effectively increase performance (e.g., selective acknowledgments). In addition, limits imposed by receivers (e.g., the advertised window size) can have a dramatic impact on connection performance [12].

The remainder of this paper is organized as follows. Section 2 describes related work on measurement studies of transport protocols. Section 3 describes the tools and methodology we use in our study. Section 4 explores interactions between middleboxes and transport protocols. Section 5 presents the results of our measurements of the deployment of various TCP mechanisms in web servers. Section 6 reports the results of our measurements about the deployment of TCP mechanisms in web clients. Section 7 discusses lessons learned in the study that challenged our assumptions and ultimately shaped our measurements and tools. Section 8 presents our conclusions, and discusses open questions and future work.

2. RELATED WORK

This paper uses and extends TBIT, which performs active measurements to characterize TCP on remote hosts [41]. For the measurements presented in this paper, TBIT's functionality was extended in two ways. New tests were implemented to assess different types of web server behavior, and the general design of the tool was extended to enable the implementation of tests that elicit path behavior by, for example, allowing the use of IP options and the generation of ICMP messages. This paper is an extension of [39].

[1] It was necessary to revise the old tests to add robustness to reordering, because minor reordering seems to have increased since we last ran these tests in 2000. As discussed in Section 7, it was also necessary to use larger packets, as many web servers wouldn't use the small MSS of 100 bytes specified in 2000. The current tests also test a much larger set of web servers.

Independent and parallel work on TBIT extensions detailed at [34, 33] includes tests for Limited Transmit, Early Retransmit, and support for the Window Scaling option in TCP. TBIT, the measurement tool used in our work, follows an earlier history of active probing of TCP. For instance, [22] treats TCP implementations as black boxes, observing how they react to external stimuli, and studying specific TCP implementations in order to assess the adherence to the specification.

There is also a considerable body of work on passive tests of TCP based on the analysis of packet traces. [43] outlines *tcpanaly*, a tool for analyzing a TCP implementation's behavior by inspecting sender and receiver packet traces of TCP connections run between pairs of hosts, while [44] outlines observed packet dynamics based on *tcpanaly*'s analysis. Finally, [49] and [12] each consider packet traces of TCP connections to a single web server, with [49] studying TCP dynamics (e.g., the response to loss, the relationship between ACK compression and subsequent loss, the use of parallel connections) and [12] assessing the properties of web clients.

In addition, there is some research in the literature on the effect of middleboxes on transport protocol performance (e.g., [13]). We do not discuss the body of research on general architectural evaluations of middleboxes, or on the effect of middleboxes on DNS, BGP, and the like. Rather, the study presented in this paper focuses on interactions between middleboxes and transport protocols.

Finally, there is a large body of literature on active and passive approaches for estimating end-to-end network path properties using TCP (e.g., [43, 14, 6]). In this paper we do not discuss TCP-based tests for estimating path properties such as loss rates, available or bottleneck bandwidth and durations of congestion episodes. Also prevalent in the literature, yet out of scope for the current effort, is the body of work based on passive measurements of traffic on a particular link to determine the breakdown of the traffic in terms of round-trip times, application layer protocols, transfer sizes, etc.

3. MEASUREMENTS: TOOLS AND DATA

As discussed above, we employ both active and passive measurements in our study into the characteristics of web clients and servers. Web servers act as data senders and web clients as data receivers in web transactions. Therefore, we use active measurements to probe web servers for congestion control and loss recovery capabilities, while using passive measurements to assess the options and resource limits enforced by web clients. Our motivation, approach and methodology is presented in the following two subsections.

3.1 Active Tests

We use TBIT [41] to conduct active measurements that probe web servers for their characteristics. A few of the active TBIT tests we present, such as the test that determines the size of the initial window, could just as easily be performed by passive packet trace analysis. However, many of the TBIT tests are not amenable to straightforward post-facto analysis of packet traces. For example, consider a test to determine if a TCP data sender is responding correctly to SACK information. To evaluate the data sender, a certain pattern of loss events is required (e.g., multiple packets lost per window of data). An active tool like TBIT can easily induce such a specific loss pattern and evaluate the behavior of the data sender in comparison to the expected behavior. Meanwhile, passive analysis would require a tool that possessed a very general understanding of a range of loss patterns and the expected responses — which would be quite tricky to get right. Inducing a specific loss pattern does run the risk of tripping pathological behavior that is not indicative of the overall behavior of the TCP implementation under study. We believe the risk for biasing our overall results in this way is small given our large sample of web servers (discussed below).

Another class of tests that involve actively attempting alternative schemes in connection initiation cannot be performed by passive trace analysis alone. For instance, consider a test for middleboxes that block TCP SYN segments when the SYNs carry advertisements for ECN. Packet traces can indicate whether connections attempting to use ECN succeed or fail. However, determining that the failure of a connection attempting to negotiate ECN is due to a middlebox blocking ECN-capable SYNs requires the active insertion of SYNs with and without ECN advertisements.

TBIT provides a set of tests, each of which is designed to examine a specific aspect of the behavior of the remote web servers, or of the path to and from the web server. Most of these tests examine the characteristics of the TCP implementations on the web servers. However, the tests are not restricted to TCP (e.g., the Path MTU Discovery [40] tests). TBIT establishes a TCP connection with the remote host at the user level. TBIT composes TCP segments (or segments from another protocol), and uses raw IP sockets to send them to the remote host. TBIT also sets up a host firewall to prevent incoming packets from reaching the kernel of the local machine; a BSD packet filter is used to deliver incoming packets to the TBIT process. TBIT's user-level connection is used to control the sending of carefully constructed packets (control, data, acknowledgment, etc.) as desired from the local host. Note that all the TBIT tests are susceptible to network conditions to some degree. For instance, if an ACK sent by TBIT is lost in transit to the web server the result of the test could be inconclusive or even wrongly reported. We have taken test-specific measures to make each of our tests as robust as possible. In addition, our large set of web servers (described below) helps to minimize any biases that bogus tests introduce into our results.

The original TBIT paper [41] repeated each test five times for each server, accepting a result as valid only if at least three of the five attempts returned results, and all of the results were the same. We did not follow that methodology in this paper; instead, we ran each test once for each server. This allowed us to process a larger set of tests.

The list of target web servers used in our study was gathered from IRcaches, the NLANR Web Caching project [2]. We used web cache logs gathered from nine different locations around the United States. Table 1 shows the cache logs used from February 2004, along with the log sizes, expressed as the number of unique IP server addresses from each cache. Since the caches are located within the continental US, most of the cached URLs correspond to domain names within the US. However, the cache logs also contain a sizable set of web servers located in the other continents. Of the 84,394 unique IP addresses[2] found in the cache logs: 82.6% are from North America, 10.2% are from Europe, 4.9% are from Asia, 1.1% are from Oceania, 1.0% are from South America and 0.2% are from Africa. A subset of the tests were also done on a list of 809 IP addresses corresponding to a list of 500 popular web sites [1].

All the TBIT tests outlined in this paper were conducted between February and May 2004. The TBIT client was always run from a machine on the local network at the International Computer Science Institute in Berkeley, CA, USA. There is no local firewall between the machine running TBIT and the Internet.

Given that data senders (web servers in our study) implement most of TCP's "smarts" (congestion control, loss recovery, etc.), most of the remainder of this paper outlines active TBIT tests to determine various characteristics of TCP implementations and networks and where the evolutionary paths collide.

[2]We note that the list of servers could be biased by a single machine having multiple unique IP addresses – which would tend to skew the results. However, due to the size of the server list, we believe that such artifacts, while surely present, do not highly skew the overall results.

Server name	Location	Cache size
pb.us.ircache.net	Pittsburgh, PA	12867
uc.us.ircache.net	Urbana-Champain, IL	18711
bo.us.ircache.net	Boulder, CO	42120
sv.us.ircache.net	Silicon Valley, CA	28800
sd.us.ircache.net	San Diego, CA	19429
pa.us.ircache.net	Palo Alto, CA	5511
sj.us.ircache.net	MAE-West, San Jose, CA	14447
rtp.us.ircache.net	Research Triangle, NC	33009
ny.us.ircache.net	New York, NY	22846

Table 1: IRCache servers and locations

3.2 Passive Tests

When characterizing web clients, passive packet trace analysis is more appropriate than active probing for two main reasons. First, initiating a connection to a web client to probe its capabilities is difficult because often web clients are user machines that do not run publicly available servers. In addition, data receivers (web clients) do not implement subtle algorithms whose impact is not readily observable in packet headers (as is the case with data senders). Rather, data receivers expose their state, limits and capabilities to the data sender in packet headers and options (e.g., SACK information, advertised window limits, etc.). Therefore, by tracing packets near a web server, client TCP implementations can be well characterized with respect to client impact on web traffic. Section 6 outlines our observations of web clients.

4. MIDDLEBOX INTERACTIONS

The increased prevalence of middleboxes calls into question the general applicability of the end-to-end principle. Middleboxes introduce dependencies and hidden points of failure, and can affect the performance of transport protocols and applications in the Internet in unexpected ways. Middleboxes that divert an IP packet from its intended destination, or modify its contents, are generally considered fundamentally different from those that correctly terminate a transport connection and carry out their manipulations at the application layer. Such diversions or modifications violate the basic architectural assumption that packets flow from source to destination essentially unchanged (except for TTL and QoS-related fields). The effects of such changes on transport and application protocols are unpredictable in the general case. In this section we explore the ways that middleboxes might interfere in unexpected ways with transport protocol performance.

4.1 Web Server SACK Generation

In Section 5 we evaluate the behavior of web servers in response to incoming SACK information from a web client. The use of SACK information by a web server is the primary performance enhancement SACK provides to web traffic. In this section, however, we focus on whether web servers generate accurate SACK information. In the normal course of web transactions this matters little because little data flows from the web client to the web server. However, while not highly applicable to web performance, this test serves to illustrate potential problems in passing SACK information over some networks. This test calls for the client to split an HTTP GET request into several segments. Some of these segments are not actually sent, to appear to the server as having been lost. These data losses seen by the server should trigger SACK blocks (with known sequence numbers) to be appended to the ACKs sent by the server.

Type of Server	Number	% of Total
Total Number of Servers	84394	100%
I. Not SACK-Capable	24361	28.8%
II. SACK Blocks OK	54650	64.7%
III. Shifted SACK Blocks	346	0.5%
IV. Errors	5037	6.0%
IV.A. No Connection	4493	5.3%
IV.B. Early Reset	376	0.4%
IV.C. Other	160	0.2%

Table 2: Generating SACK Information at Web Servers

Table 2 shows the results of the server SACK generation test. The row "Not SACK-Capable" shows the number of servers that did not agree to the SACK Permitted option during connection setup. The row listed "SACK OK" shows the number of web servers that generated SACK blocks correctly. As Table 2 shows, most of the servers show proper SACK behavior.

A relatively small number of servers, however, return improper SACK blocks. The row listed as "Shifted SACK Blocks" indicates cases where the SACK blocks received contained sequence numbers that did not correspond to the sequence space used by connection. Instead, the sequence space in the SACK blocks was *shifted*. This shifting could have been caused by a buggy TCP implementation, or by incorrect behavior from middleboxes on the path from the server to the client. We note that none of the web sites from the list of 500 popular web sites had shifted SACK blocks.

Plausible scenarios whereby middleboxes may cause incorrect SACK blocks to be returned to the web client include NATs and fingerprint scrubbers:

• NATs: Shifting of TCP sequence numbers can be done by a NAT box that modifies the URL in a request, and as a consequence has to shift the TCP sequence numbers in the subsequent data packets. In addition, the cumulative acknowledgment number and SACK blocks should be altered accordingly in the ACKs transmitted to the clients. However, due to ignorance or a bug, the SACK blocks may not be properly translated, which could explain the results of our tests.

• Fingerpring Scrubbers: The shifting of TCP sequence numbers also occurs with fingerprint scrubbers [50] designed to modify sequence numbers in order to make it hard for attackers to predict TCP sequence numbers during an attack. One way that TCP/IP fingerprint scrubbers modify sequence numbers is by choosing a random number for each connection, X_i. Then, the sequence number in each TCP segment for the connection traveling from the *untrusted* network is incremented by X_i. Likewise, each segment traveling in the opposite direction has its acknowledgment number decremented by X_i. However, if the sequence numbers in the SACK blocks are not modified as well, then the SACK blocks could be useless to the data sender.

In some cases these bogus SACK blocks will simply be thrown away as useless by the data sender. In cases when the SACK blocks are merely offset a little from the natural segment boundaries, but otherwise are within the connection's sequence space, these incorrect SACK blocks can cause performance problems by inducing TCP to retransmit data that does not need to be retransmitted and by forcing reliance on the (often lengthy) retransmission timeout to repair actual loss.

While the topic of web server SACK generation is not important in terms of the performance of web transactions, the interactions illustrated are germane to all TCP connections, and are possible explanations for some of the results in Section 5.2 when web servers negotiate SACK but do not use "Proper SACK" recovery.

Year:	2000		2004	
ECN Status	**Hosts**	**%**	**Hosts**	**%**
Number of Servers	24030	100%	84394	100%
I. Classified Servers	21879	91%	80498	95.4%
I.A. Not ECN-capable	21602	90%	78733	93%
I.B. ECN-Capable	277	1.1%	1765	2.1%
I.B.1. no ECN-Echo	255	1.1%	1302	1.5%
I.B.2. ECN-Echo	22	0.1%	463	0.5%
I.C. Bad SYN/ACK	0		183	0.2%
II. Errors	2151	9%	3896	4.6%
II.A. No Connection	2151	9%	3194	3.8%
II.A.1. only with ECN	2151	9%	814	1%
II.A.2. without ECN	0		2380	2.8%
II.B. HTTP Error	–		336	0.4%
II.C. No Data Received	–		54	0%
II.D. Others	–		312	0.4%

Table 3: ECN Test Results

4.2 ECN-capable Connections

Explicit Congestion Notification (ECN) [46] is a mechanism that allows routers to mark packets to indicate congestion, instead of dropping them. After the initial deployment of ECN-capable TCP implementations, there were reports of middleboxes (in particular, firewalls and load-balancers) that blocked TCP SYN packets attempting to negotiate ECN-capability, either by dropping the TCP SYN packet, or by responding with a TCP Reset [25]. [41] includes test results showing the fraction of web servers that were ECN-capable and the fraction of paths to web servers that included middleboxes blocking TCP SYN segments attempting to negotiate ECN-capability. The TBIT test for ECN is described in [41].

Table 3 shows the results of the ECN test for 84,394 web servers. Only a small fraction of servers are ECN-Capable – this percentage has increased from 1.1% of the web servers tested in 2000 to 2.1% in 2004. After a web server has successfully negotiated ECN we send a data segment marked "Congestion Experienced (CE)" and record whether the mark is reflected back to the TBIT client via the ECN-Echo in the ACK packet. The results are given on lines I.B.1 and I.B.2 of the table. In roughly three-quarters of cases when ECN is negotiated, a congestion indication is not returned to the client. This could be caused by a bug in the web server's TCP implementation or by a middlebox that is clearing the congestion mark as the data packet traverses the network; further investigation is needed to explore this behavior. Finally, we also observe a small number of web servers send a malformed SYN/ACK packet, with both the ECN-Echo and Congestion Window Reduced (CWR) bits set in the SYN/ACK packet (line I.C of the table).

For 3194 of the web servers, no TCP connection was established. For our TBIT test, if the initial SYN packet is dropped, TBIT resends the same SYN packet – TBIT does not follow the advice in RFC 3168 of sending a new SYN packet that does not attempt to negotiate ECN. Similarly, if TBIT receives a TCP Reset in response to a SYN packet, TBIT drops the connection, instead of sending a subsequent SYN packet that does not attempt to negotiate ECN-capability.

In order to assess how many of these connection failures are caused by the attempt of ECN negotiation, we run two back-to-back TBIT tests to each server. The first test does not attempt to negotiate ECN. After a two-second idle period, another connection is attempted using ECN. We observe that 814 connections (1% of the web servers, or 25% of the connection failures) are apparently refused because of trying to negotiate ECN, since the connec-

ECN fields in data packets	Number	% of total
ECN-capable servers	1765	100%
Received packets w/ ECT 00 (Not-ECT)	758	42%
Received packets w/ ECT 01 (ECT(1))	0	0%
Received packets w/ ECT 10 (ECT(0))	1167	66%
Received packets w/ ECT 11 (CE)	0	0%
Received packets w/ ECT 00 and ECT 10	174	10%

Table 4: Data-packet codepoints for ECN-Capable Servers

tion was established successfully when no ECN negotiation was attempted. A test limited to 500 popular web servers gives a similar result. Table 3 indicates that the fraction of web servers with ECN-blocking middleboxes on their path has decreased substantially since September 2000 – from 9% in 2000 to 1% in 2004.

We further explored the behavior of ECN-capable servers by recording the ECT codepoints in the data packets received by TBIT. Table 4 shows the number of servers from which the different codepoints were observed. TBIT received data packets with the ECT 00 codepoint from about 42% of the ECN-capable servers. The ECN specification defines two ECT code points that may be used by a sender to indicate its ECN capabilities in IP packets. The specification further indicates that protocols that require only one such a codepoint *should* use $ECT(1) = 10$. We observe that ECN-capable servers do use ECT(1) and found no server made use of the $ECT(0) = 01$ codepoint. We further observe that no router between our TBIT client and the ECN-capable servers reported Congestion Experienced (CE) in any segment. Finally, TBIT received both data segments with $ECT = 00$ and $ECT = 10$ in the same connection from about 10% of the ECN-capable servers. This behavior may indicate that the ECT code point is being erased by a network element (e.g. router or middlebox) along the path between the ECN-capable server and the client.

4.3 Path MTU Discovery

TCP throughput is generally proportional to the segment size employed [32]. In addition, [32] argues that packet fragmentation can cause poor performance. As a compromise, TCP can use Path MTU Discovery (PMTUD) [40, 38] to determine the largest segment that can be transmitted across a given network path without being fragmented. Initially, the data sender transmits a segment with the IP “Don’t Fragment” (DF) bit set and whose size is based on the MTU of the local network and the peer’s MSS advertisement. Routers along the path that cannot forward the segment without first fragmenting it (which is not allowed because DF is set) will return an ICMP message to the sender noting that the segment cannot be forwarded because it is too large. The sender then reduces its segment size and retransmits. Problems with PMTUD are documented in [35], which notes that many routers fail to send ICMP messages and many firewalls and other middleboxes are often configured to suppress all ICMP messages, resulting in PMTUD failure. If the data sender continues to retransmit large packets with the DF bit set, and fails to receive the ICMP messages indicating that the large packets are being dropped along the path, the packets are said to be disappearing into a PMTUD *black hole*. We implemented a PMTUD test in TBIT to assess the prevalence of web servers using PMTUD, and the success or failure of PMTUD for these web servers. The test is as follows:

1. TBIT is configured with a *virtual link MTU*, MTU_v. In our tests, we set MTU_v to 256 bytes.
2. TBIT opens a connection to the web server using a SYN segment containing an MSS Option of 1460 bytes (which is based on the actual MTU of the network to which the TBIT client is attached).
3. The TCP implementation at the server accepts the connection and sends MSS-sized segments, resulting in transmitted packets of MSS + 40 bytes. If the data packets from the server do not have the DF bit set, then TBIT classifies the server as not attempting to use PMTUD. If TBIT receives a packet with the DF bit set that is larger than MTU_v it rejects the packet, and generates an ICMP message to be sent back to the server.
4. If the server understands such ICMP packets, it will reduce the MSS to the value specified in the MTU field of the ICMP packet, minus 40 bytes for packet headers, and resume the TCP connection. In this case, TBIT accepts the proper-sized packets and the communication completes.
5. If the server is not capable of receiving and processing ICMP packets it will retransmit the lost data using the same packet size. Since TBIT rejects packets that are larger than MTU_v the communication will eventually time out and terminate and TBIT classifies the server/path as failing to properly employ PMTUD.

PMTUD Status	Number	% of total
Total Number of Servers	81776	100%
I. Classified Servers	71737	88%
I.A. PMTUD not-enabled	24196	30%
I.B. Proper PMTUD	33384	41%
I.C. PMTUD Failed	14157	17%
II. Errors	9956	12%
II.A. Early Reset	545	0.6%
II.B. No Connection	2101	2.5%
II.C. HTTP Errors	2843	3.4%
II.D. Others	4467	5.5%

Table 5: PMTUD Test Results

Checking for the robustness of this test involves verifying that TBIT is sending properly assembled ICMP messages back to the server upon receiving packets that are larger than the stipulated MTU size. We do such a check for this and other tests using a public domain network protocol analyzer called *ethereal* [7] which behaves in a tcpdump-like fashion but allows the user to observe easily the structure and composition of the captured packets. Using ethereal we analyze the communications between TBIT and different servers and observe the exchange of ICMP packets from TBIT to the servers, check if they are properly assembled (e.g. proper checksums), and observe the associated server response to these packets.

Table 5 shows that PMTUD is used successfully for slightly less than half of the servers on our list. For 31% of the servers on our list, the server did not attempt Path MTU Discovery. For 18% of the servers on our list, Path MTU Discovery failed, presumably because of middleboxes that block ICMP packets on the path to the web server. The results were even worse for the list of 500 popular web servers, with Path MTU Discovery failing for 35% of the sites.

Alternate methods for determining the path MTU are being considered in the Path MTU Discovery Working Group in the IETF, based on the sender starting with small packets and progressively increasing the segment size. If the sender does not receive an ACK packet for the larger packet, it changes back to smaller packets.

In a similar sender-based strategy called *black-hole detection*, if a packet with the DF bit set is retransmitted a number of times without being acknowledged, then the MSS will be set to 536 bytes

[3]. We performed a variant of the PMTUD test in which TBIT does not send the ICMP packets, to see if any server reduces the size of the packets sent simply because it didn't receive an ACK for the larger packet. We didn't find any servers performing black-hole detection.

Since a non-trivial number of network elements discard well-known ICMP packets, the results of our tests do not offer hope for protocol designers proposing to use new ICMP messages to signal various network path properties to end systems (e.g., for explicit corruption notification [23], handoff or outage notification, etc.).

4.4 IP Options

IP packets may contain options to encode additional information at the end of IP headers. A number of concerns have been raised regarding the use of IP options. One concern is that the use of IP options may significantly increase the overhead in routers, because in some cases packets with IP options are processed on the *slow path* of the forwarding engine. A second concern is that receiving IP packets with malformed IP options may trigger alignment problems on many architectures and OS versions. Solutions to this problem range from patching the OS, to blocking access to packets using unknown IP options or using IP options in general. A third concern is that of possible denial of service attacks that may be caused by packets with invalid IP options going to network routers. These concerns, together with the fact that the generation and processing of IP options is nonmandatory at both the routers and the end hosts, have led routers, hosts, and middleboxes to simply drop packets with unknown IP options, or even to drop packets with standard and properly formed options. This is of concern to designers of transport protocols because of proposals for new transport mechanisms that would involve using new IP options in transport protocols (e.g., [31, 23]).

TBIT's IP options test considers TCP connections with three types of IP options in the TCP SYN packet, the *IP Record Route Option*, the *IP Timestamp Option*, and a new option called *IP Option X*, which is an undefined option and represents any new IP option that might be standardized in the future. We experimented with two variants of Option X, both of size 4. The first variant uses a copy bit of zero, class bits set to zero and 25 as the option number. The second variant of IP Option X sets the class bits to a reserved value, and uses an option number of 31. The results for experiments with both Option X variants are similar.

Checking for the robustness of this test involves verifying that TBIT is sending properly assembled IP options in the messages sent to the servers. We also observe the server's response to options such as the *Record Route* option to verify that the server is properly understanding the options sent to it by TBIT.

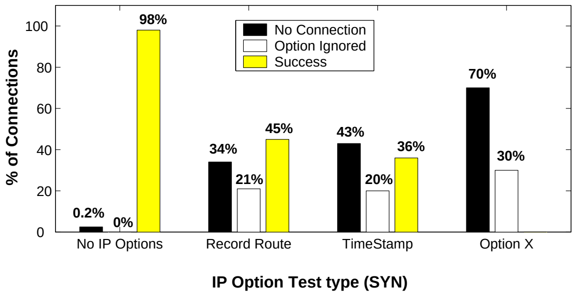


Figure 1: Handling IP Options in TCP SYN packets.

Figure 1 shows the TCP connection behavior with different IP options in the associated SYN packets. For each attempted connection there are three possible outcomes: no connection established, connection established with the IP option ignored, or IP option accepted. As Figure 1 shows, in many cases no connection was established when the Record Route Option or the Timestamp Option was included in the SYN packet. When IP Option X is included in the SYN segment, the connection was not established to over 70% of the web servers tested. The results were slightly worse when limited to the list of 500 popular web sites. This does not bode well for the deployment of new IP options in the Internet.

Most IP options are usually expressed in the first packet (e.g., the TCP SYN packet) in the communication between end hosts. We performed an additional test to assess the behavior when IP option X is placed in data packets in the middle of an established connection. For each established connection TBIT offers two classifications: "success" or "broken connection". The former indicates that the server successfully delivered its data regardless of the IP option insertion. The latter classification indicates that the insertion of the IP option forced the connection to be idle for at least 12 seconds (which we then define as "broken"). We performed two sets of tests, with and without insertion of option X. Across both sets of tests roughly 3% of the connection attempts failed. The tests without IP options show nearly 6% of the connections are "broken" for some reason. Meanwhile, when inserting IP option X into the middle of the transfer, 44% of the connections are broken, indicating a significant issue when attempting to utilize IP options in mid-connection.

4.5 TCP Options

Next we turn our attention to potential problems when TCP options are employed. TCP options are more routinely used than IP options. For instance, TCP uses the timestamp option [30] to (among other things) take round-trip time measurements more frequently than once per round-trip time, for the Protection Against Wrapped Sequences [30] algorithm and for detecting spurious timeouts [36].

However, middleboxes along a path can interfere with the use of TCP options, in an attempt to thwart attackers trying to fingerprint hosts. Network mapping tools such as NMAP (Network Mapper) use information from TCP options to gather information about hosts; this is called *fingerprinting*. Countermeasures to fingerprinting, sometimes called *fingerprint scrubbers* [50], attempt to block fingerprinting by inspecting and minimally manipulating the traffic stream. One of the strategies used by fingerprint scrubbers is to reorder TCP options in the TCP header; any unknown options may be included after all other options. The TBIT test for TCP options checks to see if sites reject connections negotiating specific or unknown TCP options, or drop packets encountered in the middle of the stream that contain those options.

The TCP options test first assesses the behavior of the web server when the TCP Timestamp option is included in the SYN packet. To test for performance with unknown TCP options, we also initiate connections using an unallocated option number, *TCP Option Y*, in the SYN packet.

Checking for the robustness of this test involves verifying that TBIT is sending properly assembled TCP options in the messages sent to the servers.

Our tests indicate a connection failure rate of about 0.2% in all scenarios. Option Y is ignored in the remainder of the connections. The timestamp option is ignored by roughly 15% of the servers (but the connection is otherwise fine). The reason the servers ignore the timestamp option is not visible to TBIT, but could be either a middlebox stripping or mangling the option or the web server not supporting timestamps. Next we assess the use of TCP options in the middle of a TCP connection, by establishing a connection without TCP options and then using the Timestamp option or Option Y on a data packet in the middle of the connection. The connection failure rate for both options is roughly 3% – indicating that sending unknown TCP options midstream is not problematic for most web servers.

5. DEPLOYMENT OF TRANSPORT MECHANISMS

This section describes TBIT tests to assess the deployment status of various TCP mechanisms in web servers. Such tests are useful from a number of angles. First, it is useful for protocol designers to understand the deployment cycle for proposed changes. In addition, as discussed previously, it is useful to test the actual behavior of proposed mechanisms in the Internet, keeping an eye out for unexpected behaviors and interactions. Another goal of this section is to guide researchers in constructing models for the design and evaluation of transport protocols. For example, if TCP deployments are dominated by NewReno and SACK TCP, then it is counter-productive for researchers to evaluate congestion control performance with simulations, experiments, or analysis based on Reno TCP.

5.1 Reno/NewReno Test

The Reno/NewReno test, adapted from the original TBIT [41], determines whether a web server uses Tahoe, Reno, or NewReno loss recovery for a TCP connection that is not SACK-capable. It is well-known that Reno's congestion control mechanisms perform poorly when multiple packets are dropped from a window of data [24]. Tracking the deployment of NewReno can guide researchers in their choices of models for simulations, experiments, or analysis of congestion control in the Internet; researchers that use Reno instead of NewReno or SACK TCP in their simulations or experiments could end up with significantly-skewed results that have little relevance for the current or future Internet. Another reason for these tests is to look for unanticipated behaviors; for example, the Reno/NewReno tests in [41] discovered a variant of TCP without Fast Retransmit that resulted from a vendor's buggy implementation.

The Reno/NewReno test determines the sender's congestion control mechanism by artificially creating packet drops that elicit the congestion control algorithm of the server. In order to enable the server to have enough packets to send, TBIT negotiates a small MSS (256 bytes in our tests). However, using a small MSS increases the chances of observing reordering packets (see Section 7), and this reordering can change the behavior elicited from the server. Therefore, the current test has evolved from the original TBIT test to make it more robust to packet reordering, and consequently to be able to classify behavior the original TBIT was not able to understand. The framework of the Reno/NewReno test is as described in [41], with the receiver dropping the 13th and 16th data packets.

Table 6 shows the results of the Reno/NewReno test. The Tahoe, Tahoe without Fast Retransmit (FR), Reno, and NewReno variants are shown in [41]. Reno with Aggressive Fast Retransmit, called RenoPlus in [41], is also shown in [41]; Reno with Aggressive Fast Retransmit has some response to a partial acknowledgment during Fast Recovery, but does not take the NewReno step of retransmitting a packet in response to such a partial acknowledgment. For each TCP variant, the table shows the number and percentage of web servers using that variant. We note that the results from May 2001 and February 2004 are not directly comparable; they use different lists of web servers, and the February 2004 list is considerably larger than the May 2001 list. However, Table 6 implies that the deployment of NewReno TCP has increased significantly in the last few years; NewReno is now deployed in 76% of the web servers on our list for which we could classify the loss recovery strategy. In addition, the deployment of TCP without Fast Retransmit has decreased significantly; this poorly-behaving variant was discovered in [41], where it was reported to be due to a vendor's failed attempt to optimize TCP performance for web pages that are small enough to fit in the socket buffer of the sender.

Date:	May 2001		Feb. 2004	
TCP Stack	Num.	% of total	Num.	% of total
Total Number of Servers	4550		84394	
I. Classified Servers	3728	72%	27914	33%
I.A. NewReno	1571	35%	21266	25%
I.B. Reno	667	15%	3925	5%
I.C. Reno, Aggressive-FR	279	6%	190	0.2%
I.D. Tahoe	201	4%	983	1.2%
I.E. Tahoe, No FR	1010	22%	1181	1.4%
I.F. Aggr. Tahoe-NoFR	0	0%	7	0%
I.G. Uncategorized			362	0.4%
II. Classified but ignored (due to unwanted drops)			11529	14%
III. Errors	822	18%	44950	53%
III.A. No Connection			2183	2.6%
III.B. Not Enough Packets			22767	27%
III.C. No Data Received			3352	4%
III.D. HTTP Error			13903	16%
III.E. Request Failed			839	1%
III.F. MSS Error			266	0.3%
III.G. Other			2035	2.4%

Table 6: Reno/NewReno Deployment in Web Servers.

5.2 Web Server SACK Usage

The SACK Behavior test reports the fraction of servers that are SACK-capable, and categorizes the variant of SACK congestion control behavior for a TCP connection with a SACK-capable client. TCP's Selective Acknowledgment (SACK) option [37] enables the transmission of extended acknowledgment information to augment TCP's standard cumulative acknowledgment. SACK blocks are sent by the data receiver to inform the data transmitter of non-contiguous blocks of data that have been received and queued. The SACK information can be used by the sender to retransmit only the data needed by the receiver. SACK TCP gives better performance than either Reno or NewReno TCP when multiple packets are dropped from a window of data [24].

The SACK Behavior test builds on the original TBIT test, with added robustness against packet reordering. TBIT first determines if the server is SACK-capable by attempting the negotiation of the SACK Permitted option during the connection establishment phase. For a SACK-capable server, the test determines if the server uses the information in the SACK blocks sent by the receiver. TBIT achieves this by dropping incoming data packets 15, 17 and 19, and sending appropriate SACK blocks indicating the blocks of received data. Once the SACK blocks are sent, TBIT observes the retransmission behavior of the server.

Table 7 shows the results for the SACK test. The servers reported as "Not SACK-Capable" are those that did not agree to the SACK Permitted option negotiated by TBIT. The servers listed as "Proper SACK" are those that responded properly by re-sending only the data not acknowledged in the received SACK blocks. The servers listed as "Semi-SACK" make some use of the information

Date:	**May 2001**		**Feb. 2004**	
SACK Type	**Num.**	**% of total**	**Num.**	**% of total**
Total Number of Servers	4550	100%	84394	100%
I. Not SACK-Capable	2696	59%	24607	29%
II. SACK-Capable	1854	41%	57216	68%
II.A. Uses SACK Info:	550	12%	23124	27%
II.A.1. Proper SACK	–		15172	18%
II.A.2. Semi-Sack	–		7952	9%
II.B. Doesn't use SACK Info:	759	17%	2722	3%
II.B.1. NewReno	–		1920	2%
II.B.2. TahoeNoFR	–		802	1%
II.C. Inconsistent Results	545	12%	173	0.2%
II.D. Not enough Packets			20740	24.5%
II.E. No Data Received			549	0.5%
II.F. HTTP Errors			9853	12%
II.G. Request Failed			2	0%
II.H. MSS Error			55	0%
III. Errors			2569	3%
III.A. No Connection			1770	2%
III.B. Other			799	1%

Table 7: SACK Deployment in Web Servers

in the SACK blocks[3]. In contrast, the servers listed as "NewReno" and "Tahoe-NO-FR" make no use of the information in the SACK blocks, even though they claim to be SACK-capable. The four types of SACK behaviors are shown in Figure 4 in [41].

While the 2001 and 2004 results are not directly comparable, the results in Table 7 indicate that the fraction of web-servers that report themselves as SACK-capable has increased since 2001, and that most (90%) of the successfully-classified SACK-capable web servers now make use of the information in SACK blocks.

As suggested by the results in Section 4.1, some of the results in Table 7 that are not "Proper SACK" could be influenced by middleboxes that translate the TCP sequence space, but do not properly translate SACK blocks.[4]

An additional D-SACK test measures the deployment of D-SACK (duplicate-SACK), an extension to the TCP SACK option for acknowledging duplicate packets [26]. When deployed at TCP receivers, D-SACK can help TCP servers detect packet replication by the network, false retransmits due to reordering, retransmit timeouts due to ACK loss, and early retransmit timeouts [20]. Our tests show that roughly half of the SACK-capable web servers implement D-SACK. The more relevant question is whether D-SACK is also deployed in web clients; we comment on this aspect further in Section 6.

5.3 Initial Congestion Window

The Initial Congestion Window (ICW) test from [41] determines the initial congestion windows used by web servers. Traditionally, TCP started data transmission with a single segment using slow start to increase the congestion window [17]. However, [16] allows an initial window of two segments, and [11] allows an initial window of three or four segments, depending on the segment size. In particular, an initial window of two or more segments can reduce the number of round-trip times needed for the transfer of a small object, and can shorten the recovery time when a packet is dropped from the initial window of data (by stimulating duplicate ACKs that potentially can trigger fast retransmit rather than waiting on the retransmission timeout).

The test starts with TBIT establishing a TCP connection to a given web server using a 256 byte MSS. The small MSS increases the chances that the server will have enough packets to exercise its ICW. TBIT then requests the corresponding web page, and receives all packets initially sent by the server, without ACKing any of the incoming segments. The lack of ACKs forces the server to retransmit the first segment in the ICW. TBIT then counts the number of segments received, reports the ICW value computed and terminates the test.

Despite the small MSS, there still may be some servers without enough data to fill their ICW. TBIT detects such cases by watching for the FIN bit set in one of the data segments. Such tests are inconclusive; the corresponding servers have an ICW equal to or larger than the number of packets received. We report only those servers that had enough data to send their entire ICW without setting the FIN bit.

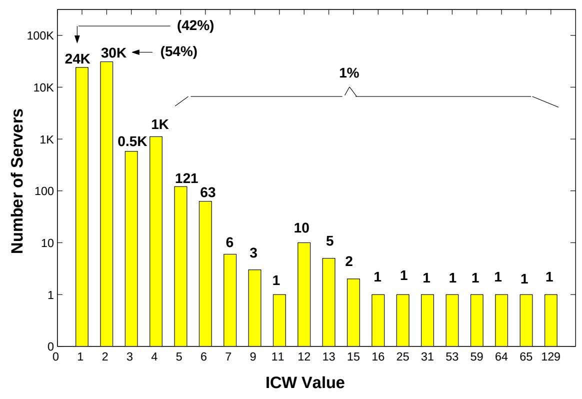


Figure 2: Initial Window Test, for an MSS of 256 bytes.

Figure 2 shows the distribution of ICWs used by the measured web servers. The figure shows that most web servers use an initial window of one or two segments, and a smaller number of servers use an initial window of three or four segments. In addition, there are a few servers using ICW values of more than four segments – including some servers using ICWs larger than 10 segments. These results are similar to those from 2001 [41], which show 2% of the web servers had an initial window of three or four segments, and 3% had initial windows larger than four segments. Thus, TCP initial windows of three or four segments are seeing very slow deployment in web servers.

We note that the ICWs shown in Figure 2 could change with different values for the MSS. For example, www.spaceimaging.com uses an ICW of 64 segments when the MSS is restricted to 256 bytes, but an ICW of *only* 14 segments with an MSS of 1460 bytes.

Figure 3 shows the fraction of connections with dropped or reordered packets, as a function of the ICW used by the server. The web servers with larger initial windows of three or four packets do not have a higher percentage of connections with packet drops. Even the occasional TCP connections with ICWs greater than four segments are not more likely to see packet drops. In addition, reordering rates are similar for ICWs of 1–3 segments and then the percentage of connections experiencing reordering drops off.

[3]There is a chance that the Semi-SACK servers actually perform Proper SACK, but have fallen prey to ACK loss. However, since SACKs are sent a number of times, the ACK loss would have to be quite bad before the server missed a block entirely. Therefore, while possible, we do not believe that ACK loss biases our aggregate conclusions in a large way.

[4]We note that the results in Section 4.1 are from a different run from those in Table 7, and have slightly different numbers for the prevalence of not-SACK-capable servers.

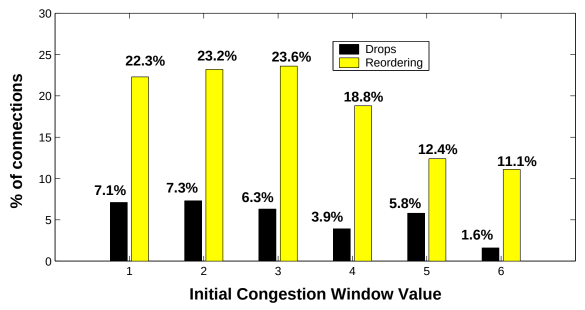


Figure 3: Percent of connections with dropped/reordered packets vs. ICW

Date:	May 2001		April 2004	
Window Halving	**Num.**	**% of total**	**Num.**	**% of total**
Total Number of Servers	4550	100%	84394	100%
I. Classified Servers	3461	76%	30690	36%
I.A. Window Halved	3330	73%	29063	34%
I.B. Window Not Halved	131	2.8%	1627	2%
II. Errors	1089	24%	53704	64%
II.A. No Connection			5097	6%
II.B. Not Enough Packets			22362	26%
II.C. No Data Received			4966	6%
II.D. HTTP Error			13478	16%
II.E. Request Failed			976	1.7%
II.G. Unwanted Reordering			4622	5.5%
II.H. Unwanted drops			732	0.9%
II.I. Other			1117	1.3%

Table 8: Window Halving Test Results

5.4 Congestion Window Halving

A conformant TCP implementation is expected to halve its congestion window after a packet loss [16]. This congestion control behavior is critical for avoiding congestion collapse in the network [27]. The Congestion Window Halving test in May 2001, from the original TBIT, verified that servers effectively halve their congestion window upon a loss event; in this section we run the test again on a much larger set of web servers, and show that the early result still holds. Because much of the traffic in the Internet consists of TCP traffic from web servers to clients, this result implies that much of the traffic in the Internet is using conformant end-to-end congestion control. This is consistent with the view that, unlike clients, busy web servers have a stake in the use of end-to-end congestion control in the Internet [27].

The Congestion Window Halving test works by initiating a transfer from the web server, waiting until the server has built up to a congestion window of eight segments, and then dropping a packet. After the loss, the server should reduce the congestion window to four segments. We classify the result as “Window Halved” if the congestion window is reduced to at most five packets after the loss, and we classify the result as “Window Not Halved” otherwise. TBIT is only able to determine a result for those servers that have enough data to send to build up a congestion window of eight segments. A detailed description of the test is available in [41]. TBIT maintains a receive window of eight segments, to limit the congestion window used by the sender.

Table 8 shows the results for the Congestion Window Halving test. Table 8 shows that, as in 2001, most of the servers exhibited correct window halving behavior. For the servers that did not halve the congestion window, a look at the packet traces suggests that these are servers limited by the receive window, whose congestion windows at the time of loss would otherwise have been greater than eight segments. One possibility is that these servers maintain the congestion window independently from the receive window, and do not properly halve the effective window when the congestion window is greater than the receive window. We note that RFC 2581 specifies that after a loss, the sender should determine the amount of outstanding data in the network, and set the congestion window to half that value in response to a loss.

5.5 Byte Counting

As described in RFC 2581 [16], TCP increases the congestion window (*cwnd*) by one MSS for each ACK that arrives during slow start (so-called “packet counting”, or “PC”). Delayed ACKs, described in [17, 16], allow a TCP receiver to ACK up to two segments in a single ACK. This reduction in the number of ACKs transmitted effectively leads to a reduction in the rate with which the congestion window opens, when compared to a receiver that ACKs each incoming segment. In order to compensate for this retarded growth, [8, 9] propose increasing *cwnd* based on the number of bytes acknowledged by each incoming ACK, instead of basing the increase on the number of ACKs received. [9] argues that such an *Appropriate Byte Counting (ABC)* algorithm should only be used in the initial slow start period, not during slow start-based loss recovery. In addition to improving slow-start behavior, ABC closes a security hole by which receivers may induce senders to increase the sending rate inappropriately by sending ACK packets that each ACK a fraction of the sequence space in a data packet [48].

The Byte Counting test is sensitive to the specific slow start behavior exhibited by the server. We have observed a large number of possible slow start congestion window growth patterns in servers which do not correspond to standard behavior. For this reason, we were forced to implement an elaborate test for an algorithm as simple as Byte Counting. The test works as follows, for an initial congestion window of one segment:

1. Receive and acknowledge the first data packet. After this ACK is received by the server, the congestion window should be incremented to two packets (using either PC or ABC).
2. ACK the second and third data packets with separate ACK packets. After these two ACKs are received, the server should increment its congestion window by two packets (using either PC or ABC).
3. ACK the next four packets with a single cumulative ACK (e.g., with an acknowledgment of the seventh data packet).
4. Continue receiving packets without ACKing any of them until the server times out and retransmits a packet.
5. Count the number of new packets, N, that arrived at least three quarters of a round-trip time after sending the last ACK.
6. Count the number of earlier ACKs, R, (out of the three earlier ACKs) which were sent within an RTT of the first of the N packets above. These are ACKs that were sent shortly before the last ACK. For servers with the standard expected behavior, R should be 0.
7. Compute the increase, L, in the server congestion window triggered by the last ACK as follows:

$$L = N - 4 - 2 * R \quad (1)$$

- If $L = 1$, then PC was used.
- If $L > 1$, then the server increased its congestion window by L segments in response to this ACK. We classify this as the server performing Byte Counting with a limit of at least L.

Slow-Start Behavior	Number	% of total
Total Number of Servers:	44579	100%
I. Classified Servers	23170	52%
I.A. Packet Counting	15331	51.9%
I.B. Appropriate Byte Counting	65	0.1%
II. Unknown Behvaior	288	0.6%
III. Errors	21121	47.4%
III.A. No Connection	528	1.2%
III.B. Not enough packets	13112	29.4%
III.C. No data received	386	0.9%
III.D. HTTP Error	215	0.5%
III.E. Request Failed	181	0.4%
III.F. Packet Size Changed	5762	13%
III.G. Unwanted Reordering	827	2%
III.H. Other	7	0%

Table 9: Byte Counting Test Results

The observation behind the design of this test is that N is the number of packets that the server sent after receiving the ACK packets in the preceding RTT. These N packets are assumed to include two packets for each ACK received that ACKed only one packet. These N packets are also assumed to include four packets due to the advance in the cumulative acknowledgment field when the last ACK was received. Any extra packets sent should be due to the increase in the congestion window due to the receipt of the last ACK. We note that the complexity of this test is an example in which the difference between theory and practice in protocol behavior significantly complicates the scenarios that need to be considered. Table 9 shows the results of the Byte Counting test, showing that Byte Counting had minimal deployment when these tests were performed.

We note that our Byte Counting test is not sufficient to distinguish between Packet Counting, and ABC with $L = 1$. The Appropriate Byte Counting test in [34] returns two split acknowledgements for a single packet, and can distinguish between Packet Counting and ABC with $L = 1$. [34] reports that 80 of the 200 servers tested used ABC with $L = 1$, and none of the servers used ABC with $L = 2$.

Our Byte Counting test uses the estimated RTT in inferring which data packets were sent by the server after the server received the final ACK packet, and this use of the estimated RTT is a possible source of error. From looking at packet traces, we observed one or two tests that were labeled by TBIT as Byte Counting, where the actual RTTs in the connection were unclear, and the packet trace was consistent with either Byte Counting or Packet Counting. However, from the traces that we looked at, we don't think that this possible source of error is a significant factor in our overall results.

5.6 Limited Transmit

TCP's Limited Transmit algorithm, standardized in [10], allows a TCP sender to transmit a previously unsent data segment upon the receipt of each of the first two duplicate ACKs, without inferring a loss or entering a loss recovery phase. The goal of Limited Transmit is to increase the chances of connections with small windows to receive the three duplicate ACKs required to trigger a fast retransmission, thus avoiding a costly retransmission timeout. Limited Transmit potentially improves the performance of TCP connections with small windows.

The Limited Transmit test assesses deployment in web servers. Like the Byte Counting test, this test is sensitive to the size of the initial window employed by the server. The strategy of the test in all cases is the same but the presence or absence of Limited Transmit must be determined in the context of a specific ICW. For an ICW of four packets, the test works as follows:

Limited Transmit (LT) Behavior	Number	% of total
Total Number of Servers	38652	100%
I. Classified Servers	29023	75%
I.A. LT Implemented	8924	23%
I.B. LT Not Implemented	20099	52%
II. Errors	9629	25%
II.A. No Connection	420	1.1%
II.B. Not enough packets	3564	9.2%
II.C. No Data Received	257	0.7%
II.D. HTTP Errors	224	0.6%
II.E. Request Failed	163	0.4%
II.F. Packet Size Changed	4900	12.7%
II.G. Other	101	0.3%

Table 10: Deployment of Limited Transmit

1. Acknowledge the first data segment in the initial window of four segments. Upon receiving this ACK, the server should open its window from four to five segments, and send two more packets, the 5th and 6th segments.
2. Drop the second segment.
3. TBIT sends two duplicate ACKs triggered by the reception of segments 5 and 6. TBIT does not send ACKs when segments 3 and 4 arrive, to provide for increased robustness against unexpected server congestion window growth. Only one duplicate ACK would suffice to trigger the Limited Transmit mechanism at the server but TBIT sends two to account for the possibility of ACK losses.
4. If the server does not implement Limited Transmit, then it will do nothing when it receives the duplicate ACKs. If the server does implement Limited Transmit, then it will send another segment when it receives each duplicate ACK.

We note that if the duplicate ACKs sent by TBIT are dropped in the network, then TBIT will see no response from the web server, and will interpret this as a case where Limited Transmit is not deployed. Greater accuracy could be gained by running the test several times for each web server, as done with the TBIT tests in [41].

Table 10 shows the results from our tests. The table shows that Limited Transmit is deployed in at least a fifth of the web servers in our dataset. The Limited Transmit test is sensitive to the size of the initial window and therefore care needs to be exercised with respect to the size of packets being received from the server. Note that if there is a change in the packet size for packets in the middle of the connection, TBIT flags the result "Packet Size Changed", and does not classify that server. As shown in the table, this happened with some frequency and renders that test inconclusive. Furthermore, a certain minimum number of packets need to be transferred for TBIT to be able to classify a server, therefore servers with small web pages are classified as not having enough packets.

5.7 Congestion Window Appropriateness

When the TCP sender does not have data to send from the application, or is unable to send more data because of limitations of the TCP receive window, its congestion window should reflect the data that the sender has actually been able to send. A congestion window that doesn't reflect current information about the state of the network is considered invalid [29]. TBIT's Congestion Window Appropriateness test examines the congestion window used by web servers following a period of restrictions imposed by the receive window.

In this test, TBIT uses a TCP receive window of one segment to limit the web server's sending rate to one packet per RTT. After five RTTs, TBIT increases the receive window significantly, and waits to see how many packets the web server sends in response. Consider a web server using standard slow-start from an initial window of K segments, increasing its congestion window without regard to whether that window has actually been used. Such a web server will have built up a congestion window of $K + 5$ segments after five round-trip times of sending one packet per round-trip time, because each ACK increases the congestion window by one segment. The web server could suddenly send $K + 5$ packets back-to-back when the receive window limitation is removed. In contrast, a web server using the Congestion Window Validation procedure from [29] will have a congestion window of either two segments or the ICW, whichever is larger.[5]

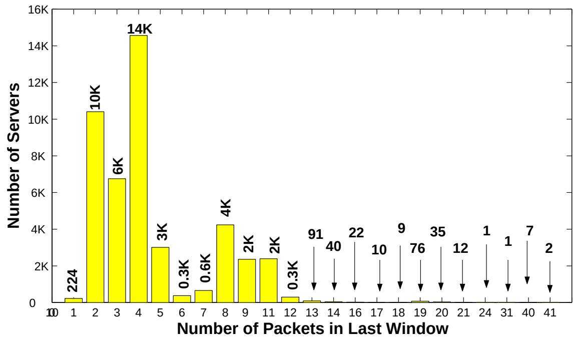


Figure 4: The congestion window after a receive-window-limited period

Figure 4 shows the number of segments that each server sends in response to the increased receive window at the end of the Congestion Window Appropriateness test. The majority of servers respond with a window of two to four packets, showing moderate behavior consistent with Congestion Window Validation. A smaller fraction of the servers respond with a large window of eight or nine packets, suggesting that the server increases its congestion window without regard for the actual number of segments sent.

In some cases the number of segments transmitted shows that the server is violating the standard rules for opening the congestion window during slow-start, even aside from the issue of the appropriateness of a congestion window that has never been used. Because a conformant web server can have an initial window of at most four segments, a conformant web server can have a congestion window of at most nine segments after five single-packet acknowledgments have been received.

It would also be possible to use TBIT to explore the congestion window used by web servers after an application-limited period. TBIT can create an application-limited period by using repeated HTTP requests, once per round-trip time, each requesting only a range of bytes from the web page. After this enforced application-limited period, TBIT would follow by requesting the full web page.

[5]RFC 2861 [29] was written when the ICW was still only one packet, so RFC 2861 doesn't explicitly say that the ICW should be taken as a lower bound for the reduced congestion window. However, RFC 3390 says that the sender MAY use the initial window as a lower bound for the restart window after an idle period, and it makes sense that the sender would use the initial window as a lower bound in this case as well.

5.8 Minimum RTO

TCP uses a retransmit timer to guarantee the delivery of data in the absence of feedback from the receiver. The duration of this timer is referred to as the *Retransmit TimeOut* (RTO). A detailed description of the algorithm for computing the RTO can be found in [17, 42]. [42] recommends a minimum RTO of one second, though it is well-known that many TCP implementations use a smaller value for the minimum RTO. A small minimum RTO gives better TCP performance in high-congestion environments, while a larger minimum RTO is more robust to reordering and variable delays [15].

The TBIT test to explore minimum RTO values initiates a connection with a given server, and receives and acknowledges packets as usual until packet 20 has been received. By this time, the TCP sender has taken a number of measurements of the round-trip time, and has estimated the average and mean deviation of the round-trip time for computing the RTO. Upon packet 20's reception, TBIT stops ACKing packets and measures the time until the retransmission for the last packet; this is used as an estimate of the RTO used by the server.

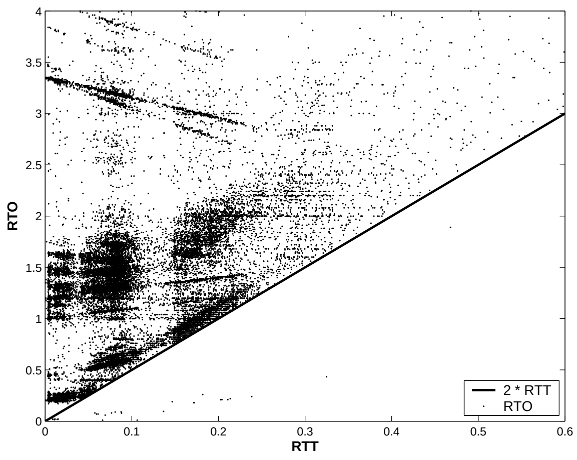


Figure 5: RTO vs. Initial RTT

Figure 5 shows the RTO values used by servers for retransmitting the given packet. The x-axis shows the initial round-trip time, and the y-axis shows the measured RTO for the server. The RTO used by a server will often be larger than the minimum RTO enforced by that server. However, of the 37,000 servers shown in Figure 5, 40% responded with an RTO of less than a second.[6]

6. PASSIVE CLIENT MEASUREMENTS

The previous sections discuss results from active measurements from a TBIT client machine to a target set of web server destinations. Such analysis sheds light on the correctness and performance characteristics of a significant population of in-the-field web servers, and also provides insights into the characteristics of the intermediate nodes on the paths that carry packets between the TBIT client and the servers. However, this is only one part of the story. We are also interested in observing the Internet from the perspective of web clients. To achieve this perspective we collect full packet traces of traffic to and from the web server of our research laboratory. In this section we present the result from the analysis of those traces.

[6]The minimum RTO test requires a transfer of at least 20 packets and therefore we could not assess the minimum RTO to over half the web servers in our list.

We collected packet traces of full TCP packets to and from port 80 on our lab's web server (*www.icsi.berkeley.edu*) for roughly two weeks (from February 24, 2004 to March 10, 2004). Such a dataset provides a wealth of information about a disparate set of web clients. However, given the heterogeneity of the Internet we do not claim this dataset is *representative*. Rather we present it as a data point. Capturing entire packets allowed us to verify the TCP checksum and discard packets that did not pass. In the dataset we observed 206,236 connections from 28,364 clients (where a "client" is defined as an IP address). Of these, 613 (or, 0.3%) connections were not analyzed due to the packet trace missing the initial SYN sent by the client and therefore throwing off our analysis.[7] We do not believe that deleting these connections biased our results.

The first set of items we measure are the capabilities the client TCPs advertise during connection startup. Of all the clients, 205 (or 0.7%) show inconsistent capabilities across connections from the same IP address. An example inconsistency would be one connection from a particular IP address advertising support for SACK, while a subsequent connection does not. Our inconsistency check includes the SACK permitted option, the timestamp option, the window scale option (and the advertised value), the MSS option (and the MSS value) and whether the connection advertises support for ECN. Options may be inconsistent due to a NAT between the client and our server that effectively hides multiple clients behind a single IP address. Alternatively, system upgrades and configuration changes may also account for inconsistency over the course of our dataset.

We next study TCP's cumulative acknowledgment and the selective acknowledgment (SACK) option [37]. In our dataset, 24,906 clients (or 87.8%) advertised "SACK permitted" in the initial SYN. Across the entire dataset 236,192 SACK blocks were returned from the clients to our web server. We observe loss (retransmissions from the server) without receiving any SACK blocks with only two clients that advertised SACK capability. This could be due to a bug in client implementations, middlebox interference or simple network dynamics (e.g., ACK loss). Therefore, we conclude that clients advertising "SACK permitted" nearly always followup with SACK blocks, as necessary.

As outlined in Section 4.1, the TBIT SACK tests yield some transfers where the sequence numbers in the SACK blocks from the clients are "shifted" from the sequence numbers in the lost packets. Inaccurate SACK blocks can lead to the sender spuriously retransmitting data that successfully arrived at the receiver, and waiting on a timeout to resend data that was advertised as arriving but which was never cumulatively acknowledged. To look for such a phenomenon in web clients or middleboxes close to clients we analyzed the SACK blocks received from the clients and determined whether they fall along the segment boundaries of the web server's transmitted data segments. We found 1,242 SACK blocks (or 0.5%) that do not fall along data segment boundaries. These SACK blocks were generated by 49 clients (or 0.2%). The discrepancy between the rate of receiving strange SACK blocks and the percentage of hosts responsible for these SACK blocks suggests a client-side or middlebox bug. These results roughly agree with the results in Section 4.1. Of the bogus SACK blocks received, 397 were offset – i.e., the sequence numbers in the SACK block were within the sequence space used by the connection, but did not fall along data segment boundaries. Meanwhile, the remaining 845 bogus SACK blocks were for sequence space never used by the connection. Note: a possible explanation for some of the strange SACK blocks is that our packet tracing infrastructure missed a data segment and therefore when a SACK arrives we have no record of the given packet boundaries. However, given that (i) the discrepancy between the overall rate of observing these SACKs when compared to the percentage of clients involved and (ii) many of the bogus SACK blocks were completely outside the sequence space used by the connection, we believe that packet capturing glitches are not the predominant cause of these bogus SACK blocks.

[7]The dataset is really composed from separate 24-hour packet traces, and so connections which continue across two of these traces are lost mid-connection.

Next we outline the prevalence of Duplicate SACK (D-SACK) [26] blocks in our dataset. D-SACK blocks are used by data receivers to report data that has arrived more than once and can be used for various tasks, such as attempting to set a proper duplicate ACK threshold and reversing needless changes to TCP's congestion control state caused by spurious retransmissions [20]. In our dataset we observed 809 hosts (or, 3% of all hosts) sending D-SACK blocks. Note that more than 3% of the hosts may support D-SACK, but were not faced with a situation whereby transmission of a D-SACK was warranted.

We also investigated whether there were cases when the cumulative acknowledgment in incoming ACKs did not fall on a segment boundary. Of the roughly 4.7 million ACKs received by our web server, 18,387 ACKs contained cumulative ACK numbers that did not agree with the segments sent. These ACKs were originated by 36 clients. The rate of receiving these strange ACKs is 0.4% in the entire dataset, meanwhile the number of clients responsible for these ACKs represents 0.1% of the dataset, indicating that buggy clients or middleboxes may be the cause of these ACKs.

In our dataset, the timestamp option is advertised by 6,106 clients (or 21.5%). Clients that do not accurately echo timestamp values to the server or middleboxes that alter the timestamp of a passing packet may cause performance degradation to the connection by increasing or reducing the retransmission timeout (RTO) estimate of the server. If the RTO is too small the data sender will timeout prematurely, needlessly resending data and reducing the congestion window. If the RTO is too large performance will suffer due to needless waiting before retransmitting a segment. In our dataset, 20 clients returned at least one timestamp that the server never sent (some of the timestamps returned by these clients were valid). This result suggests that the network and the endpoints are faithfully carrying timestamps in the vast majority of cases.

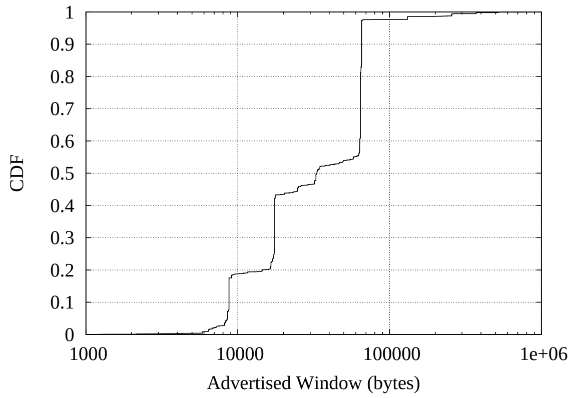


Figure 6: Distribution of advertised windows use by web clients.

We next examine the advertised windows used by web clients. [12] shows how the client's advertised window often dictates the ultimate performance of the connection. Figure 6 shows the distribution of the maximum window advertisement observed for each client in our dataset. Roughly, the distribution shows modes at

8 KB, 16 KB and 64 KB. These results show an increase in advertised window sizes over those reported in [12] (in 2000). In our dataset the median advertised window observed is just over 32 KB and the mean is almost 44 KB, whereas [12] reports the median advertised window as 8 KB and a mean of 18 KB. Additionally, 7,540 clients (or 26.6% of our dataset) advertised support for TCP's window scaling option [30], which calls for the advertised window to be scaled by a given factor to allow for larger windows than can naturally be advertised in the given 16 bits in the TCP header. Just over 97% of the clients that indicate support for window scaling advertise a window scale factor of zero — indicating that the client is not scaling its advertised window (but understands window scaling if the server wishes to scale its window). Just over 1% of the clients in our dataset use a scale factor of 1, indicating that the advertised window in the client's segments should be doubled before using. We observed larger window scale factors (as high as 9) in small numbers in our dataset.

We next look at the MSS advertised by web clients in the initial three-way handshake. Two-thirds of the clients used an MSS of 1460 bytes (Ethernet-sized packets). Over 94% of the clients used an MSS of between 1300 bytes and 1460 bytes. The deviation from Ethernet-sized packets may be caused by tunnels. Roughly 4% of the clients in our dataset advertised an MSS of roughly 536 bytes. We observed advertisements as small as 128 bytes and as large as 9138 bytes. This analysis roughly agrees with [12].

Finally, we note that we observed 48 clients (or 0.2% of the clients in our dataset) advertising the capability to use Explicit Congestion Notification (ECN) [46]. That is, only 48 clients sent SYNs with both the ECN-Echo and Congestion Window Reduced bits in the TCP header set to one.

7. MEASUREMENT LESSONS

In conducting the measurements presented in this paper we observed a number of properties of the network and the end systems that challenged our assumptions and ultimately shaped our tools. In this section, we distill several lessons learned that others conducting similar measurements should keep in mind.

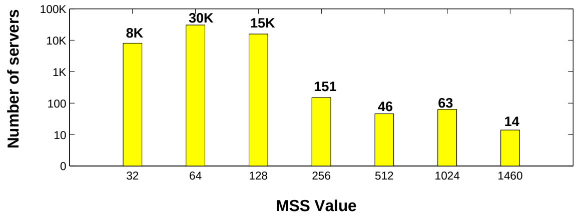


Figure 7: Minimum MSS Test

The TBIT tests presented in this paper attempt to use a small MSS so that the web server splits the data transfer into more segments than it naturally would. In turn, this provides TBIT with additional ways to manipulate the data stream. For instance, if a server transmits one segment of 1280 bytes then TBIT cannot easily conduct certain tests, such as assessing the Initial Window. However, if the server is coaxed into sending 10 segments of 128 bytes more tests become possible (due to the increased variety of scenarios TBIT can present to the server). The set of TBIT tests presented in [41] employed a 100 byte MSS. When we initiated the present study we found this MSS to be too small for a significant number of web servers. Therefore, determining the smallest allowable MSS is important for TBIT-like measurements. Figure 7 shows the distribution of minimum MSS sizes we measured across the set of web servers used in our study. As shown, nearly all servers will accept an MSS as small as 128 bytes, with many servers supporting MSS sizes of 32 and 64 bytes. Another aspect of the segment size that surprised us is that segment sizes sometimes change during the course of a connection (e.g., as reported in the tests of ABC in Section 5) . Therefore, we encourage researchers to design tests that are robust to changing packet sizes (or, at the least warn the user of a test when such an event is observed).

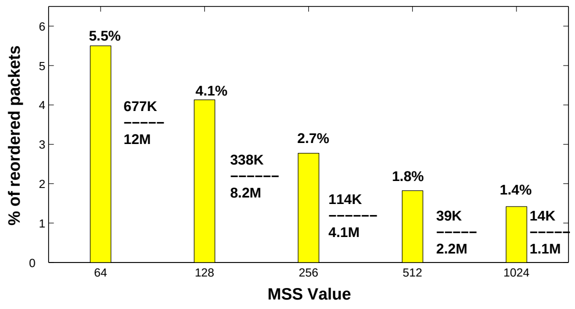


Figure 8: Reordering vs MSS

Choosing a small MSS to maximize the number of segments the web server transmits is a worthy goal. However, we also find that as the MSS is reduced the instances of packet reordering increase. Figure 8 shows the percentage of reordered segments as a function of the MSS size.

One explanation of this phenomenon is that using a smaller MSS yields transfers that consist of more segments and therefore have more opportunities for reordering. Alternatively, small packets may be treated differently in the switch fabric — which has been shown to be a cause of reordering in networks [18]. Whatever the cause, researchers should keep this result in mind when designing experiments that utilize small segments. Additionally, the result suggests that performance comparisons done using small segments may not be directly extrapolated to real-world scenarios where larger segments are the rule (as shown in Section 6) since reordering impacts performance [18, 19].

As outlined in Section 5, we find web servers' slow start behaviors to be somewhat erratic at times. For instance, Section 5.5 finds some web servers using "weak slow start" where the web server does not increase the congestion window as quickly as allowed by the standards[8]. In addition, we also found cases where the congestion window is opened more aggressively than allowed. These differences in behavior make designing TBIT-like tests difficult since the tests cannot be staked around a single expected behavior.

Also, we found that some of our TBIT measurements could not be as *self contained* as were all the tests from the original TBIT work [41]. Some of the tests we constructed depended on peculiarities of each web server. For instance, the Limited Transmit test outlined in Section 5.6 requires apriori knowledge of the web server's initial window. This sort of test complicates measurement because multiple passes are required to assess some of the capabilities of the web servers.

Finally, we note that in our passive analysis of web client characteristics *verifying the TCP checksum is key* to some of our observations. In our dataset, we received at least one segment with a bad TCP checksum from 120 clients (or 0.4% of the clients in the dataset). This prevalence of bogus checksums is larger than the prevalence of some of the identified characteristics of the web client (or network). For instance, we identified only 49 clients that advertise support for ECN and report receiving bogus SACK blocks

[8]Such non-aggressive behavior is explicitly allowed under the standard congestion control specification [16], but we found it surprising that a web server would be more conservative than necessary.

TCP Mechanism	Section	Deployment Status
Loss Recovery	6, 5.2 5.1	SACK is prevalent (in two-thirds of servers and nine-tenths of clients). NewReno is the predominant non-SACK loss recovery strategy.
D-SACK	6, 5.2	D-SACK is gaining prevalence (supported by 40% of servers and at least 3% of clients).
Congestion Response	5.4	Most servers halve their congestion window correctly after a loss.
Byte Counting	5.5	Most web servers use packet counting to increase the congestion window.
Initial Cong. Window	5.3	Most web servers use an ICW of 1 or 2 segments.
ECN	4.2	ECN is not prevalent.
Advertised Window	6	The most widely used advertised window among clients is 64 KB with many clients using 8 KB and 16 KB, as well.
MSS	6	Most of the clients in our survey use an MSS of around 1460 bytes.

Table 11: Information for modeling TCP behavior in the Internet.

Behavior	Section	Possible Interactions with Routers or Middleboxes
SACK	5.2,6	In small numbers of cases, web clients and servers receive SACK blocks with incorrect sequence numbers.
ECN	4.2	Advertising ECN prevents connection setup for a small (and diminishing) set of hosts.
PMTUD	4.3	Less than half of the web servers successfully complete Path MTU Discovery. PMTUD is attempted but fails for one-sixth of the web servers.
IP Options	4.4	For roughly one-third of the web servers, no connection is established when the client includes an IP Record Route or Timestamp option in the TCP SYN packet. For most servers, no connection is established when the client includes an unknown IP Option.
TCP Options	4.5	The use of TCP options does not interfere with connection establishment. Few problems were detected with unknown TCP options, and options included in data packets in mid-stream.

Table 12: Information on interactions between transport protocols and routers or middleboxes.

from 36 clients. If we had not verified the TCP checksum these two characteristics could have easily been skewed by mangled packets and we'd have been none-the-wiser. In our experiments, we used *tcpdump* [4] to capture full packets and then *tcpurify* [5] to verify the checksums and then store only the packet headers in the trace files we further analyzed.[9]

8. CONCLUSIONS AND FUTURE WORK

The measurement study reported in this paper has explored the deployment of TCP mechanisms in web servers and clients, and has considered the interactions between TCP performance and the behavior of middleboxes along the network path (e.g., SACK information generation, ECN, Path MTU Discovery, packets with IP or TCP options). Our concerns have been to track the deployment (or lack of deployment) of transport-related mechanisms in transport protocols; to look out for the ways that the performance of mechanisms in the Internet differs from theory; to consider how middleboxes interfere with transport protocol operation; and to consider how researchers should update their models of transport protocols in the Internet to take into account current practice and a more realistic network environment (Table 11). The main contribution of this work is to illustrate the ways that the performance of protocol mechanisms in the Internet differ from theory. The insights gathered from our measurements involving the interactions between TCP and middleboxes along the network path are summarized in Table 12.

There exist significant avenues for future work in the light of the results presented in this paper. There are a wealth of important TCP behaviors that have not been examined, and new TCP mechanisms are continually being proposed, standardized and deployed. Assessing their deployment, characteristics and behaviors in the context of the evolving Internet architecture are useful avenues of future work.

[9]Before truncating a captured packet to store on the headers for later processing, *tcpurify* stores a code in the TCP checksum field indicating whether the checksum in the original packet was right, wrong or whether *tcpurify* did not have enough of the packet to make a determination.

Another class of extensions to this work is exploring the behavior of TCP in additional applications (e.g., peer-to-peer systems, email, web caching, etc.). Also, we performed all our tests having the measurement client machine in our research laboratory. Further network and host dynamics may be elicited by performing TBIT-like tests in different environments such as having the TBIT client behind different types of middleboxes (e.g. firewalls, NATs, etc.) at different security levels.

An additional interesting area for future investigation is using TBIT-like tools for *performance* evaluation. For instance, a performance comparison of servers using various initial congestion window values or servers with and without SACK-based loss recovery may prove useful. Developing techniques for conducting this kind of performance comparison in a solid and meaningful way (and detecting when such a comparison is not meaningful) is a rich area for future investigation. Furthermore, performing tests from multiple vantage points to assess middlebox prevalence and behavior on a wider scale would be useful.

As new transport protocols such as SCTP and DCCP begin to be deployed, another area for future work will be to construct tools to monitor the behavior, deployment and characteristics of these protocols in the Internet.

While we examined some ways that middleboxes interfere with TCP communications, a key open question is that of assessing ways that middleboxes affect the *performance* of transport protocols or of applications. One middlebox that clearly affects TCP performance is that of Performance Enhancing Proxies (PEPs) [21] that break single TCP connections into two connections potentially changing end-to-end behavior. While [13] presents some results in this general area, additional active tests would be useful to investigate this area further.

Finally, a completely different kind of test that may benefit from the active probing approach outlined in this paper would be one to detect the presence or absence of Active Queue Management mechanisms at the congested link along a path. To some extent, this can be done with passive tests, by looking at the pattern of round-trip times before and after a packet drop. However, active tests may be more powerful, by allowing the researcher to send short runs of back-to-back packets, as well as potentially problematic, in that the tool might need to induce transient congestion in the network to assess the queueing strategy.

Acknowledgments

Orion Hodson assisted with our TBIT measurements. Thanks also to Gorry Fairhurst, Sourabh Ladha, and the anonymous reviewers for their helpful feedback.

9. REFERENCES

[1] Alexa web search - top 500 web sites. URL http://www.alexa.com/site/ds/top_sites.

[2] NLANR Web Caching project. http://www.ircache.net/.

[3] PMTU Black Hole Detection Algorithm Change for Windows NT 3.51. Microsoft Knowledge Base Artible - 136970.

[4] tcpdump. URL http://www.tcpdump.org.

[5] tcpurify. URL http://irg.cs.ohiou.edu/~ eblanton/tcpurify/.

[6] Tools for Bandwidth Estimation. Web page, URL 'http://www.icir.org/models/tools.html'.

[7] Ethereal: Network Protocol Analyzer, 2004.

[8] M. Allman. On the Generation and Use of TCP Acknowledgements. *Computer Communication Review*, 28(5), October 1998.

[9] M. Allman. TCP Byte Counting Refinements. *Computer Communication Review*, 29(3), July 1999.

[10] M. Allman, H. Balakrishnan, and S. Floyd. Enhancing TCP's Loss Recovery Using Limited Transmit, January 2001. RFC 3042.

[11] M. Allman, S. Floyd, and C. Partridge. Increasing TCP's Initial Window, 2002. RFC 3390.

[12] Mark Allman. A Web Server's View of the Transport Layer. *Computer Communications Review*, 30(5):10–20, October 2000.

[13] Mark Allman. On the Performance of Middleboxes. In *ACM SIGCOMM/USENIX Internet Measurement Conference*, pages 307–312, October 2003.

[14] Mark Allman and Vern Paxson. On Estimating End-to-End Network Path Properties. In *ACM SIGCOMM*, pages 229–240, 1999.

[15] Mark Allman and Vern Paxson. On Estimating End-to-End Network Path Properties. In *ACM SIGCOMM*, September 1999.

[16] Mark Allman, Vern Paxson, and W. Richard Stevens. TCP Congestion Control, April 1999. RFC 2581.

[17] R. Barden. Requirements for Internet Hosts – Communication Layers, October 1989. RFC 1122.

[18] Jon C.R. Bennet, Craig Patridge, and Nicholas Schetman. Packet Reordering is not Pathological Network Behavior. *IEEE/ACM Transactions on Networking*, 7(6), August 1999.

[19] Ethan Blanton and Mark Allman. On Making TCP More Robust to Packet Reordering. *ACM Computer Communication Review*, 32(1):20–30, January 2002.

[20] Ethan Blanton and Mark Allman. Using TCP DSACKs and SCTP Duplicate TSNs to Detect Spurious Retransmissions, 2004. RFC 3708.

[21] John Border, Markku Kojo, Jim Griner, Gabriel Montenegro, and Zach Shelby. Performance Enhancing Proxies Intended to Mitigate Link-Related Degradations, June 2001. RFC 3135.

[22] Douglas E. Comer and John C. Lin. Probing TCP Implementations. In *USENIX Summer 1994 Conference*, 1994.

[23] Wesley Eddy, Shawn Ostermann, and Mark Allman. New Techniques for Making Transport Protocols Robust to Corruption-Based Loss. *ACM Computer Communication Review*, 34(5), October 2004.

[24] Kevin Fall and Sally Floyd. Simulation-based Comparisons of Tahoe, Reno, and SACK TCP. *Computer Communications Review*, 26(3), July 1996.

[25] S. Floyd. Inappropriate TCP Resets Considered Harmful, 2002. RFC 3360.

[26] S. Floyd, J. Mahdavi, M. Mathis, and M. Podolsky. An Extension to the Selective Acknowledgement (SACK) Option for TCP, July 2000. RFC 2883.

[27] Sally Floyd and Kevin Fall. Promoting the Use of End-to-End Congestion Control in the Internet. *IEEE/ACM Transactions on Networking*, 7(6), August 1999.

[28] Sally Floyd and Eddie Kohler. Internet Research Needs Better Models. In *Proceedings of the First Workshop on Hot Topics in Networks (HotNets-I)*, October 2002.

[29] Mark Handley, Jitendra Padhye, and Sally Floyd. TCP Congestion Window Validation, June 2000. RFC 2861.

[30] V. Jacobson, R. Barden, and D. Borman. TCP Extensions for High Performance, May 1992. RFC 1323.

[31] Amit Jain, Sally Floyd, Mark Allman, and Pasi Sarolahti. Quick-Start for TCP and IP, February 2005. Internet-Draft draft-amit-quick-start-04.txt (work in progress).

[32] Christopher Kent and Jeffrey Mogul. Fragmentation Considered Harmful. In *ACM SIGCOMM*, October 1987.

[33] Sourabh Ladha. The TCP Behavior Inference Tool (TBIT) Extensions, 2004. URL http://www.cis.udel.edu/ ladha/tbit-ext.html.

[34] Sourabh Ladha, Paul D. Amer, Armando L. Caro, and Janardhan Iyengar. On the Prevalence and Evaluation of Recent TCP Enhancements. *Globecom 2004*, November 2004.

[35] Kevin Lahey. TCP Problems with Path MTU Discovery, September 2000. RFC 2923.

[36] R. Ludwig and M. Meyer. The Eifel Detection Algorithm for TCP, 2003. RFC 3522.

[37] Matt Mathis, Jamshid Mahdavi, Sally Floyd, and Allyn Romanow. TCP Selective Acknowledgement Options, October 1996. RFC 2018.

[38] Jack McCann, Steve Deering, and Jeffrey C. Mogul. Path MTU Discovery for IP Version 6, August 1996. RFC 1981.

[39] Alberto Medina, Mark Allman, and Sally Floyd. Measuring Interactions Between Transport Protocols and Middleboxes. In *ACM SIGCOMM/USENIX Internet Measurement Conference*, October 2004.

[40] Jeffrey C. Mogul and Steve Deering. Path MTU Discovery, November 1990. RFC 1191.

[41] Jitendra Padhye and Sally Floyd. Identifying the TCP Behavior of Web Servers. In *ACM SIGCOMM*, August 2001.

[42] V. Paxson and M. Allman. Computing TCP's Retransmission Timer, November 2000. RFC 2988.

[43] Vern Paxson. Automated Packet Trace Analysis of TCP Implementations. In *ACM SIGCOMM*, September 1997.

[44] Vern Paxson. End-to-End Internet Packet Dynamics. In *ACM SIGCOMM*, September 1997.

[45] Jon Postel. Transmission Control Protocol, September 1981. RFC 793.

[46] K.K. Ramakrishnan, Sally Floyd, and David Black. The Addition of Explicit Congestion Notification (ECN) to IP, September 2001. RFC 3168.

[47] J.H. Saltzer, D.P. Reed, and David Clark. End-to-End Arguments in System Design. In *Proceedings of the Second International Conference on Distributed Computing Systems*, pages 509–512, August 1981.

[48] S. Savage, N. Cardwell, D. Wetherall, and T. Anderson. TCP Congestion Control with a Misbehaving Receiver. *ACM Computer Communication Review*, 29(5), October 1999.

[49] Srinivasan Seshan, Hari Balakrishnan, Venkata N. Padmanabhan, Mark Stemm, and Randy Katz. TCP Behavior of a Busy Internet Server: Analysis and Improvements. San Francisco, CA, March 1998.

[50] Matthew Smart, G. Robert Malan, and Farnam Jahanian. Defeating TCP/IP Stack Fingerprinting. In *9th USENIX Security Symposium*, pages 229–240, 2000.

Notes on Burst Mitigation for Transport Protocols

Mark Allman
ICSI / ICIR
mallman@icir.org

Ethan Blanton
Purdue University
eblanton@cs.purdue.edu

ABSTRACT

In this note we explore the various causes of micro-bursting in TCP connections and also the behavior of several mitigations that have been suggested in the literature along with extensions we develop herein. This note methodically sketches the behavior of the mitigations and presents the tradeoffs of various schemes as a data point in the ongoing discussion about preventing bursting in transport protocols.

Categories and Subject Descriptors

C.2.2 [**Computer-Communication Networks**]: Network Protocols; C.2.6 [**Computer-Communication Networks**]: Internetworking

General Terms

Algorithms, Performance

Keywords

Bursting, TCP

1. INTRODUCTION

In this note we investigate the Transmission Control Protocol's (TCP) [20] bursting behavior.[1] TCP's bursting behavior can be problematic in several ways (as sketched in more detail in the next section). First, bursting can stress queues in the network and lead to packet loss, which can in turn negatively impact both the connection doing the bursting and traffic sharing the stressed router. Second, bursting can cause scaling on short timescales as well as increase queuing delays in routers. Over the years, several researchers have suggested mitigations to TCP's burstiness. We investigate and show the behavior of both these previously proposed mitigations and newly extended techniques.

TCP naturally sends two distinct kinds of segment bursts into the network. First, each TCP acknowledgment (ACK) covering new data that arrives at the TCP sender slides a sending window and liberates a certain number of segments which are transmitted at the line-rate of the connected network. We will call bursts of segments sent in response to a single incoming ACK *micro-bursts*. TCP's congestion control algorithms cause the second kind of bursting behavior. While using the slow start algorithm, a TCP sender increases the amount of data transmitted into the network by a factor of 1.5–2 during each subsequent round-trip time (RTT) (the exact factor depends on whether the receiver generates delayed ACKs [7, 4], whether the sender uses byte counting [2], and the network dynamics of the path the ACKs traverse). An additional cause of macro-bursting is ACK compression [23]. This phenomenon occurs when ACK packets get "bunched up" behind larger data packets in router queues and end up arriving at the end host more rapidly than they were transmitted by the peer. This "bunching up" can cause bursting. We term the bursts caused by slow start and ACK compression *macro-bursts* since they occur over longer time periods than the micro-bursts sketched above.

TCP's macro-burstiness has been the topic of several papers in the literature. [16] analyzes the impact of TCP's macro-burstiness on queueing requirements. [2] proposes an increase in the macro-burstiness of TCP in an attempt to mitigate some of the performance hit caused by delayed ACKs during slow start. [17] proposes using rate-based pacing during slow start to reduce the queueing requirement TCP's macro-bursts place on routers in a network path. Additionally, [1] investigates a general pacing scheme for TCP.

We do not consider macro-bursts further here. Rather, we outline the behavior of schemes to reduce micro-burstiness. TCP's "normal" level of micro-burstiness is to transmit 1–3 segments in response to each ACK (assuming no ACK loss and nothing else anomalous on the connection) [4]. Each ACK that acknowledges new data causes TCP's transmission window to slide. In the normal case, with delayed ACKs, the window slides by 2 segments for each ACK. When TCP is increasing the size of the window, an additional third segment may be sent due to this increase (which happens during slow start and congestion avoidance, at different intervals). As outlined in § 4 bursts of more than 3 segments can happen naturally in TCP connections. We consider bursts of 3 segments or less to be acceptable (per the standards) and bursts of over 3 segments in length to be anomalous[2]. That is not to say that the bursts are caused by problematic TCP implementations[3]. Here we concentrate on bursts caused by an interaction between TCP's congestion control algorithms and specific network dynamics.

[1]This note is cased in terms of TCP, but should naturally apply to transport protocols that use similar window-based congestion control algorithms (e.g., SCTP [21], DCCP [15] using CCID2 [9]).

[2]TCP specifically allows bursts of 4 segments at the beginning of a connection or after a lengthy idle period [3]. However, we do not consider this one-time allowance to indicate that the TCP congestion control specification tolerates such micro-bursts. Also, [2] allows for micro-bursts up to 4 segments in length as a regular occurrence during TCP slow start. However, [2] is an experimental document and not standard.

[3]Bursts *are* caused by buggy TCP implementations, as well, of course. For instance, stretch ACKs outlined in [18, 19] cause micro-bursts. [6] shows that such stretch ACKs are not uncommon in today's Internet.

The goal of this note is to illustrate the causes of bursting and the behavior of several mitigations that have been proposed, as well as extensions developed within. The purpose is not to make hard conclusions, but rather to offer a data point in the ongoing discussion about micro-bursting within transport protocols[4]. This note is organized as follows. § 2 details related work. § 3 discusses a number of mechanisms that reduce the size of bursts. § 4 presents simulations illustrating TCP's bursting behavior and the behavior of the burst mitigation strategies. We conclude and sketch future work in § 5.

2. RELATED WORK

The literature contains several studies dealing with the impact of micro-bursts and potential methods for mitigation of large burst sizes. [22] considers bursts caused by re-using HTTP connections after an idle period, and shows that rate-based pacing is useful in reducing burstiness and increasing performance. We discuss the topic of using rate-based pacing as a general micro-burst mitigation strategy in § 3 and § 4.

[12] discusses micro-bursts in a more general way, considering techniques for both detecting and reducing micro-bursts. It introduces the Use It or Lose It algorithm discussed in § 3.

[11] discusses the behavior and performance impact of micro-bursts that occur after loss recovery in satellite networks across a range of TCP variants. [8] also illustrates micro-bursts in the context of loss recovery, and introduces the MaxBurst algorithm discussed in § 3.

[14] investigates the causes of bursts (both micro- and macro-bursts) in the network and their impact on aggregate network traffic. It finds that bursts at sources create scaling on shortb timescales and can cause increased queuing delays in intermediate nodes along a network path.

[6] attempts to quantify micro-bursting in the Internet, and correlate micro-bursts with performance impact on individual TCP connections, and suggests that while micro-bursts of moderate size are well-tolerated (in the context of individual TCP connections, in contrast to the findings in [14] about aggregate traffic), larger bursts greatly increase the probability of packet drops.

We incorporate the techniques previously defined in the literature and make several extensions to them in our analysis.

3. BURST MITIGATION ALGORITHMS

Several mitigation strategies have been proposed by various researchers to control micro-bursts. The two fundamental methods that have been proposed to deal with micro-bursts are to (i) limit the number of segments sent in response to a given ACK or to (ii) spread the transmission of the burst of segments out using rate-based pacing. With regards to the former, the two basic ways that have been proposed for limiting the size of the bursts are: (a) placing a simple limit, called *maxburst*, on the transmission of new segments in response to any given ACK, and (b), scaling TCP's congestion window (*cwnd*) back to prevent a line-rate burst from being transmitted. Both of these controls are enforced on a per ACK basis. In this note we explore two variants of each basic strategy. In addition, we discuss the application of a rate-based pacing scheme to the bursting scenarios studied.

MaxBurst (MB). This mechanism, introduced in [8], is a simple limit on the number of data segments that can be transmitted in response to each incoming ACK. The `MaxBurst()` function in figure 1 provides the pseudo-code for the MB strategy. The code includes a check to ensure that the most recent ACK that arrived is valid (i.e., the cumulative acknowledgment number in the arriving segment is not below the current cumulative ACK point). After passing this check the sender's transmission of data is constrained by (i), ensuring that the amount of outstanding data (*ownd*) does not exceed *cwnd*[5], and (ii), by ensuring that at most a constant number of segments (`MB_SIZE`) are transmitted. This method lends itself to controlling the acceptable bursting by offering a direct control (`MB_SIZE`) of the allowable burst size on each ACK.

```
def MaxBurst ():
    if ackno > highack:
        count = 0
        while (ownd < cwnd) && \
                  (count < MB_SIZE):
            send_packet ()
            count += 1
            ownd += 1


def AggressiveMaxBurst ():
    count = 0
    while (ownd < cwnd) && \
              (count < MB_SIZE):
        send_packet ()
        count += 1
        ownd += 1


def UseItOrLoseIt ():
    if cwnd > (ownd + MB_SIZE):
        cwnd = ownd + MB_SIZE
    while ownd < cwnd:
        send_packet ()
        ownd += 1


def CongestionWindowLimiting ():
    if cwnd > (ownd + MB_SIZE):
        if ssthresh < cwnd:
            ssthresh = cwnd
        cwnd = ownd + MB_SIZE
    while ownd < cwnd:
        send_packet ()
        ownd += 1
```

Figure 1: Burst mitigation pseudo-code.

Aggressive Maxburst (AMB). We introduce this mechanism, given in the `AggressiveMaxBurst()` function in figure 1 and is similar to MB. The AMB scheme calls for the removal of the validity check on the incoming ACK used in the MB scheme. This may seem odd as [20] declares ACKs with acknowledgment numbers less than the connection's current cumulative ACK point to be "invalid". However, as shown in the next section these ACKs can be useful (at least in some cases) for clocking out new segments during a burst mitigation phase (which was not considered in [20]).

[4]E.g., as discussed on the IETF's transport area working group mailing list in June, 2003. Archive at: http://www1.ietf.org/mail-archive/web/tsvwg/current/.

[5]We ignore the limit imposed by the receiver's advertised window from our discussions (and code) in this section for simplicity since the advertised window limit applies to all of the burst mitigation strategies.

Use It or Lose It (UI/LI). This mechanism, introduced in [12], calls for the TCP sender to monitor the size of the burst that will be transmitted in the response to an ACK arrival and reduce *cwnd* accordingly if a large line-rate burst will be transmitted. The pseudo-code for the UI/LI scheme is given in the `UseItOrLoseIt()` function of figure 1. This function first compares the *ownd* and the *cwnd* to gauge whether a burst of more than a certain size (`MB_SIZE`) will be transmitted. If so, the *cwnd* is scaled back to limit the burst to no more than `MB_SIZE` segments. Under this scheme the actual sending of data is only constrained by the *cwnd*, in contrast with the two controls used in MB and AMB (in which transmission is controlled by both the `MB_SIZE` and the *cwnd*).

Congestion Window & Slow Start Threshold Limiting (CWL). We extend UI/LI with this mechanism in order to mitigate the performance penalty imposed by a potentially large decrease in the *cwnd* when using UI/LI. The `CongestionWindowLimiting()` function in figure 1 shows the pseudo-code for CWL. Like UI/LI the CWL technique compares *cwnd* to the *ownd* to detect bursts. If a burst would otherwise be sent and the value of the slow start threshold (*ssthresh*) is less than the current *cwnd* then *ssthresh* is set to the *cwnd* before the *cwnd* is reduced to mitigate the burst. This causes TCP to use slow start (exponential *cwnd* increase), as opposed to congestion avoidance (linear *cwnd* increase) to build the *cwnd* back to the point it was at when the burst was detected. In effect, CWL uses *ssthresh* as a history mechanism. This contrasts with the UI/LI scheme which leaves the method for *cwnd* increase to chance by leaving *ssthresh* untouched.

Rate-Based Pacing (RBP). While the above schemes take efforts to limit the number of segments transmitted in response to an ACK, using RBP limits the rate the segments are emitted from the sender. The benefit of RBP is that — unlike the above schemes — there is no reduction in the amount of data sent. However, as will be observed in the next section, sometimes there are natural gaps in the connection after a burst that a TCP could fill with a rate-based smoothing of a burst. However, at other times there is not a natural gap in which to send a rate-based volley of segments. If no natural pause in the transmission occurs then TCP either has to use something different from RBP or discontinue the use of traditional ACK clocking (even if temporarily) or RBP will not offer any burst mitigation.

4. SIMULATIONS

In this section we use the *ns* simulator to illustrate four different situations that cause bursts to naturally occur in TCP connections. We then illustrate how the four mitigations outlined in figure 1 cope with the burstiness. Our simulations involve a simple 4 node network with two endpoints connected by two intermediate routers. The endpoints connect to the routers using 10 Mbps links with a one-way delays of 1 ms. The routers are connected to each other using a 1.5 Mbps link with a one-way delay of 25 ms (except for the simulations involving ACK reordering given in § 4.4 which use a one-way delay of 75 ms on the link between the routers). The router employs drop-tail queueing with a maximum queue depth of 20 packets. Each endpoint uses the *sack1* variant [8] of TCP included in *ns* and delayed acknowledgments [7, 4]. Unless otherwise noted the advertised window used in these simulations is 500 segments (enough to never be a limit on sending). All simulations involve a single TCP connection.

In addition, each simulation involves some manipulation to ensure that bursting occurs (as will be described in the subsequent subsections). We note that the exact setup of the simulations and the manipulations performed are not terribly important to the results presented in this note. As will be shown, all the situations discussed in this section can occur naturally. The simulations are presented to show the stock TCP *behavior* and that of the mitigations in theoretical terms and are not a complete study of how well the mitigations work (which will be highly dependent on specific network dynamics and their prevalence).

4.1 ACK Loss

First, we explore bursts caused by ACK loss. During these simulations, all ACKs between 3.3 and 3.4 seconds of simulated time are dropped. Figure 2(a) shows a time-sequence plot of the behavior of stock TCP in the face of ACK loss. As shown in the figure, each of the missed ACKs represents a missed opportunity for the sender to transmit new data. When an ACK (finally) arrives at nearly 3.45 seconds the sender transmits a burst of roughly 20 segments.

The second two plots in figure 2 ((b) and (c)) show the behavior of MB and AMB for allowable burst sizes (`MB_SIZE`) of 3 and 5 segments respectively. In this simulation there is no difference in the behavior of MB and AMB since no ACKs arrive out-of-order. As shown the *maxburst* limit on sending reduces the size of the burst just before 3.45 seconds to 3 (or 5) segments and continues to limit the sending of segments to no more than 3 (or 5) segments per ACK until the *ownd* reaches the size of *cwnd*. The time required to build *ownd* to *cwnd* is directly related to the choice of the `MB_SIZE` parameter used. When using an `MB_SIZE` of 3 segments the *ownd* increase takes roughly 150 ms longer than when `MB_SIZE` is 5 segments in the sample case.

Figure 2(d) shows the behavior of UI/LI (with an `MB_SIZE` of 3) in the face of ACK loss. When the burst is detected just before 3.45 seconds *cwnd* is reduced, followed by roughly 1 second of slow start. The amount of slow start (if any) used after the burst mitigation under UI/LI is arbitrary and depends on the value of *ssthresh* before the burst is detected. If the *cwnd* is less than *ssthresh* after UI/LI, then slow start will be used until *cwnd* reaches *ssthresh*. However, if *cwnd* is not reduced to below *ssthresh*, then the linear increase of congestion avoidance will be used.

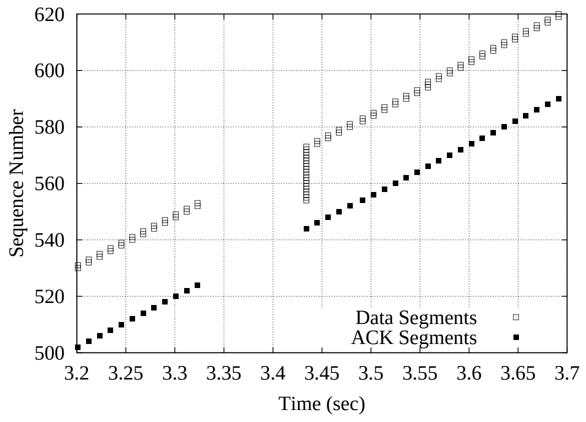
Figure 2(e) shows the behavior of CWL when faced with ACK loss. As with UI/LI, *cwnd* is reduced just before 3.45 seconds. However, as discussed in § 3, CWL sets *ssthresh* to *cwnd* (assuming the connection is using congestion avoidance) before scaling back *cwnd* to prevent the burst. This provides a sort-of history that helps the connection return to its pre-burst state and provides more determinism in the *cwnd* growth after burst mitigation than UI/LI. As shown, CWL utilizes slow start to increase the *cwnd* for almost 2 seconds yielding a larger *cwnd* when compared to UI/LI.

We also note that subfigures (b) and (e) are exactly the same in this situation. That is, MB, AMB and CWL provide the same effective response to the burst in terms of data sent into the network when `MB_SIZE` is 3 segments, even though the methodology is different.

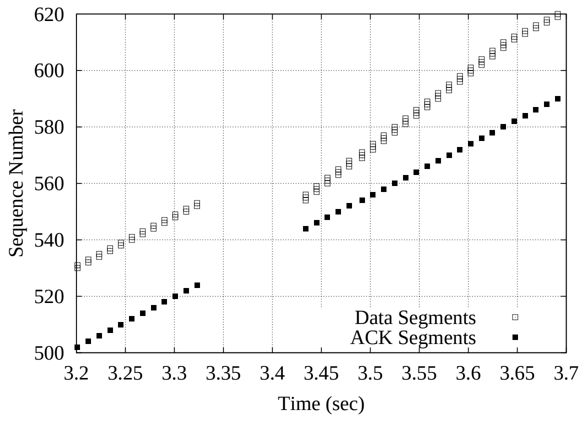
Finally, we note that in the case of the burst depicted in this situation (and sent in figure 2(a)), a rate-based pacing scheme would have no natural lull over which to spread the burst of segments.

4.2 Limited Advertised Window

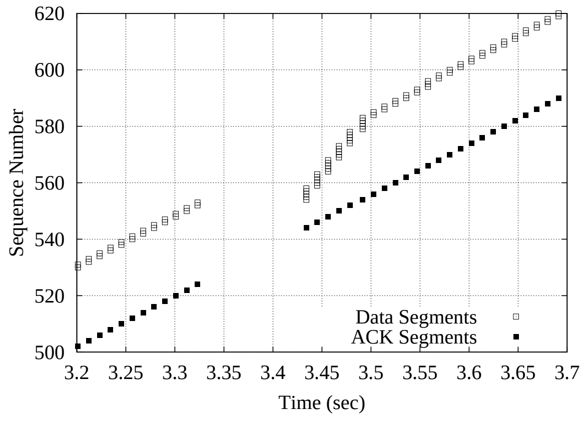
In the next scenario we explore bursting as caused by the advertised window during loss recovery. For this set of simulations we set the advertised window to 32 segments. Figure 3(a) shows the time-sequence plot of standard TCP's behavior. The TCP sender is able to fill the advertised window and then takes a single loss. Fast retransmit [13, 4] is used to retransmit the segment. Fast recovery should then take over and clock out new segments during the sec-



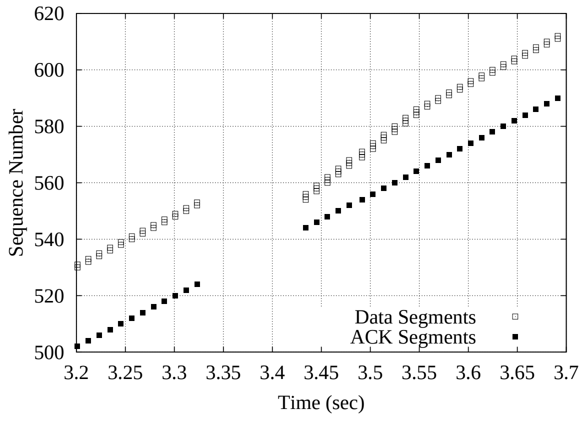
(a) Stock TCP



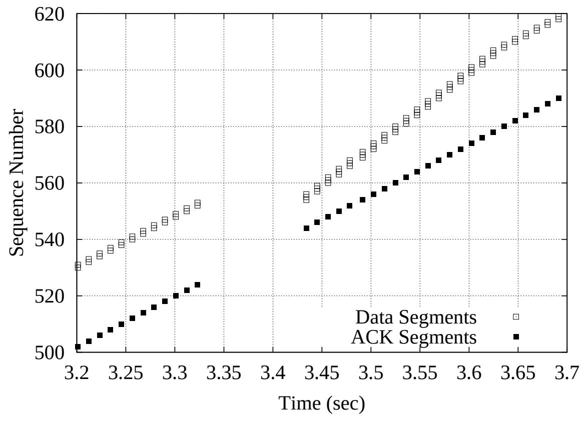
(b) MB 3 (and AMB 3)



(c) MB 5 (and AMB 5)

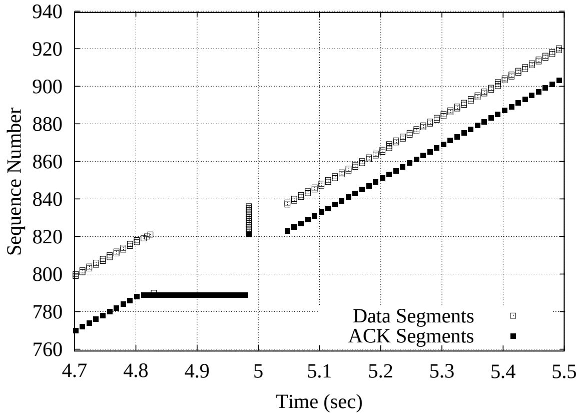


(d) UI/LI 3

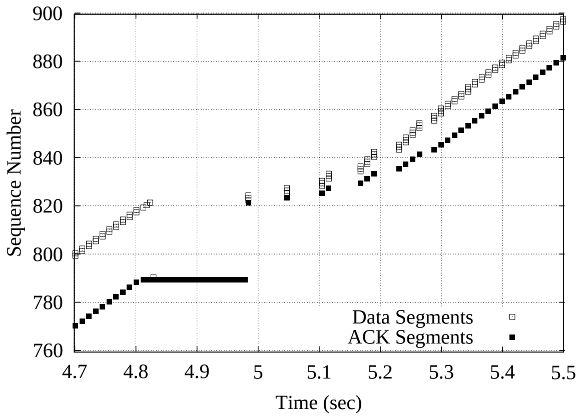


(e) CWL 3

Figure 2: ACK loss induced bursting behavior.



(a) Stock TCP



(b) MB 3 and AMB 3 and UI/LI 3 and CWL 3

Figure 3: Advertised window induced bursting behavior.

ond half of the loss recovery period. However, because the sender has filled the advertised window no new data can be sent. Therefore, when the retransmitted segment is ACKed (along with the rest of the outstanding data – in this case) a burst of data is transmitted into the network. In the situation shown in figure 3(a) the burst is nearly 20 segments. This phenomena has been observed and detailed elsewhere for different strategies of TCP loss recovery and different network environments [8, 11].

While algorithmically different, all the burst mitigation strategies perform identically in this situation (with a common `MB_SIZE` of 3), as shown in figure 3(b). Since the *ownd* is collapsed to zero by the large cumulative ACK that arrives just before 5 seconds, all of the schemes start from the same place. Further, with an `MB_SIZE` of 3 segments the *maxburst*-based schemes exactly mimic slow start with delayed ACKs and no ACK loss (where each ACK clocks out 3 data segments). Finally, the non-determinism shown in UI/LI in the last subsection is not present because *ssthresh* is set to a known point at the time of the fast retransmit.

Unlike the ACK loss case explored in the last section, the advertised window limit shown in this section offers a reasonably straightforward situation in which to use rate-based pacing. In this case, all the data has drained from the network and so there is a lull in activity after the burst is transmitted that lasts roughly one RTT (from just before 5 seconds to around 5.05 seconds). Therefore, an RBP scheme could easily space out the segments evenly over the course of the RTT following the burst detection.

4.3 Application Limiting

The next case of bursting we examine is caused by the application layer protocol's sending pattern. Figure 4(a) shows the time-sequence plot of a TCP transfer where the application does not send data from just after 0.65 seconds until 0.8 seconds. The plot shows that no data is sent in this interval even as ACKs arrive. However, when the application begins sending again at 0.8 seconds the underlying TCP transmits a burst of roughly 20 segments. The burst caused by such an idle period can be mitigated by using an idle timer (as introduced in [13] and discussed in [12]). After the idle timer fires the TCP connection must start sending with a small *cwnd* (per RFC 3390 [3]) and use slow start to increase *cwnd*. In addition, Congestion Window Validation (CWV) [10] can come into play in this scenario. CWV calls for TCP to use only "valid" window sizes — i.e., windows that have been fully utilized and therefore are known to be reasonable, but windows that are not fully used are not known to be appropriate for the current network conditions and therefore the window will be reduced.

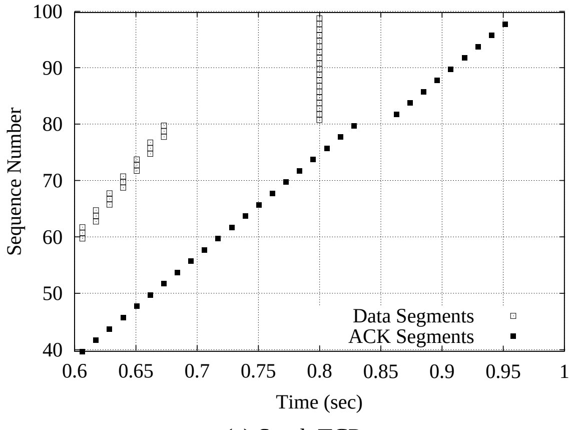
Figure 4(b) shows the behavior of all mitigations given in figure 1 in the face of the application sending pattern sketched above. As in the last subsection MB, AMB, UI/LI and CWL perform the same in this simulation (with a common `MB_SIZE` of 3 segments). MB and AMB perform the same because there are not out-of-order ACKs in this simulation. The *maxburst* schemes perform the same as the strategies based on limiting the *cwnd* because both *maxburst* and slow start call for transmitting 3 segments on each ACK received. In this simulation UI/LI and CWL perform the same. However, as discussed above the non-determinism of UI/LI does not guarantee that these two schemes will behave the same. In particular, CWL could use slow start longer than UI/LI (as shown in § 4.1), yielding a larger *cwnd*.

Application limited situations sometimes present a straightforward opportunity for using RBP, while at other times offering a more muddled situation. For instance, if all data on a connection has drained from the network and is acknowledged and then the application produces more data the TCP sender can easily pace out the congestion window over its estimate of the RTT. The flip side is shown in figure 4(a), whereby there is a period where no data is available for transmission, but not all the data drains from the network. Therefore, when the burst occurs there is still an ACK clock and there is not a natural gap in the data transmission over which the burst can be smoothed.

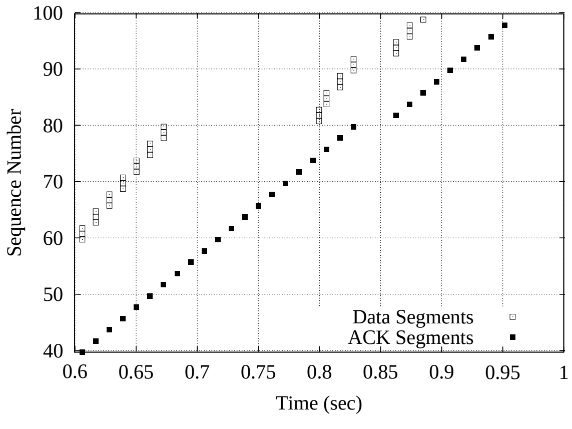
4.4 ACK Reordering

The last bursting situation we examine in detail involves ACK reordering[6]. [5] finds that packet reordering is not a rare occurrence over the MAE-East exchange, suggesting that ACK reordering may not be an uncommon phenomenon on at least some network paths. Figure 5(a) shows the behavior of stock TCP in the face of ACK reordering. In the simulation we changed the delay imposed on

[6]This is not to say that bursting does not occur in additional situations. However, we believe the four we sketch in this note cover the space of general types of bursting scenarios.



(a) Stock TCP



(b) MB 3 and AMB 3 and UI/LI 3 and CWL 3

Figure 4: Bursting caused by application layer sending patterns.

the link between the routers that carries the ACKs from 75 ms to 1 ms at 6.32 seconds and then back to 75 ms at 6.33 seconds. This caused a single ACK to "pass" a number of previously sent ACKs in the trip from the receiver to the sender. When this ACK arrives the TCP window slides and a burst of segments is sent, as shown around 6.33 seconds in figure 5(a). Following the burst, a number of ACKs arrive that are not used to clock out data segments because (a) the ACKs convey no new information and (b) the *cwnd* is full.

Figure 5(b) shows the behavior of the MB technique (with an `MB_SIZE` of 3 segments). The burst limit does not allow the full use of *cwnd* until just after 6.5 seconds. Figure 5(c) shows the behavior of AMB, which uses each "invalid" ACK to clock out an `MB_SIZE` burst of segments. While these ACKs convey no new information for the connection, from a reliability standpoint, they can be used to clock out new segments because, unlike stock TCP, the TCP is not utilizing the entire window due to the burst mitigation. As shown in the figure, the last two ACKs are, in fact, not used to clock new data into the network. This is explained by the TCP sending 3 segments on each of the previous invalid ACKs, rather than 2 segments as TCP would normally transmit during congestion avoidance. Therefore, the *cwnd* is filled using less ACKs than normal and so the last two "invalid" ACKs are ignored.

Figure 5(d) shows the behavior of the UI/LI technique. This figure shows that when the burst is detected (just after 6.4 seconds) the *cwnd* is clamped to mitigate the burst and congestion avoidance (linear *cwnd* increase) ensues. Finally, figure 5(e) shows the behavior of CWL. In contrast to the UI/LI scheme, CWL utilizes slow start to increase *cwnd* to the value it had prior to the burst detection. As in the previous sections, MB and CWL show identical on-the-wire behavior in our simulations, even though the two schemes use different methods for obtaining their behavior.

ACK reordering presents a tricky situation for RBP. As shown in figure 5(a), there is a natural lull in the connection after the burst is transmitted. At first glance, it may seem natural to attempt to smooth the burst over this pause. However, the reception of the ACK that causes the burst could indicate either ACK reordering (as is the case in figure 5) or simply a case of dropped ACKs (as discussed previously). If the sender could know that ACK reordering was the root cause then conceivably RBP could be used over an interval that depends on the length of the reordering. On the other hand, if the root cause was known to be dropped ACKs then there is no clear way to utilize RBP. Without knowledge about the cause of a larger than expected cumulative ACK it is difficult to make sound decisions as to what course of action to take.

5. CONCLUSIONS

This note's contribution is in (i) the methodical analysis of the behavior of several burst mitigation schemes and (ii) the extension of several previously defined burst mitigation strategies. In doing so, several high-level points have surfaced:

- The behavior and performance of UI/LI is dependent on the congestion control state when UI/LI is invoked. We introduced the notion of using *ssthresh* as a history mechanism to avoid this non-determinism in CWL.

- If faster than slow-start transmission rate increase is desired after a burst is detected then MB or AMB are needed because *cwnd*-based schemes can increase the transmission rate no faster than slow start. The flip side of this issue is the question of whether it is safe to increase faster than slow start would. We suspect that the answer is that it is indeed safe, given that the connection is increasing only to a previously (and recently) known appropriate operating point.

- CWL provides a *single control* for the amount of data a TCP connection can transmit into the network at any given point. This is arguably a clean approach to controlling the load imposed on the network. On the other hand, MB provides for *separation of concerns*. In other words, limiting the sizes of micro-bursts is, in some sense, a different task than limiting the overall transmission rate to control network congestion. Therefore, using two different mechanisms may make sense. As noted above, the MB scheme is more flexible than the CWL scheme. However, an additional drawback is that MB adds a second control and brings with it the possibility of the two transmission controllers interacting poorly and causing problems.

- The simulations in § 4.4 shows that there are times when traditionally discarded "invalid" ACKs could be useful in keeping the ACK clock going. Of course, these ACKs have been

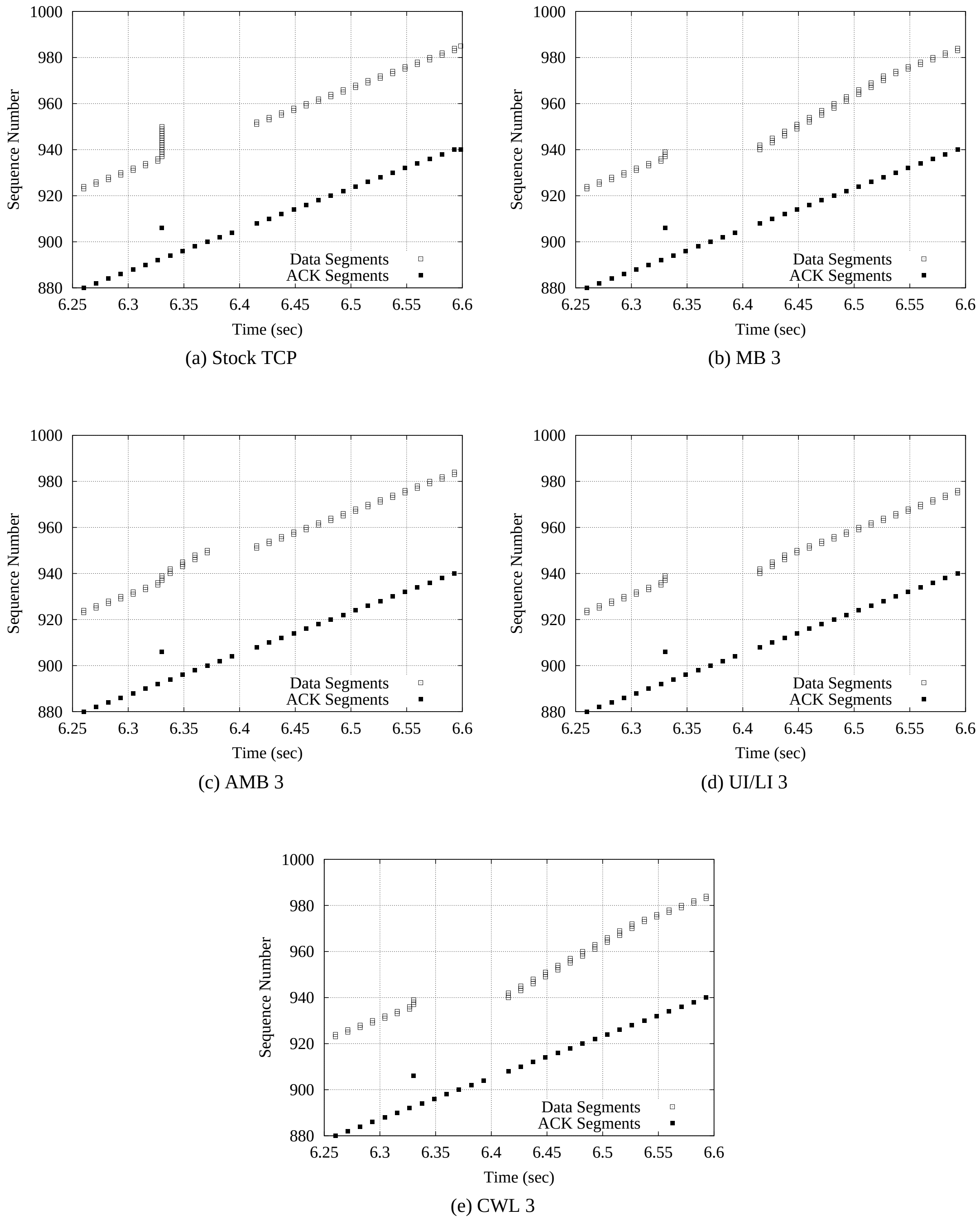


Figure 5: Bursting caused by ACK reordering.

traditionally disregarded for a reason and these ACKs could be bogus for any number of reasons (network duplicates, old segments from previous connections, etc.). Therefore, careful thought is required before using such ACKs to trigger further data transmission.

There are pros and cons to all of the strategies studied in this note. Therefore, we do not concretely find any one "best" mechanism. Rather, we hope that this note provides useful information for researchers and implementers to use when reasoning about the various possibilities.

Acknowledgments

Armando Caro, Sally Floyd, Tom Henderson, Kacheong Poon, Scott Shenker and Randall Stewart provided helpful discussions about the topics covered in this note. The first author's work was funded by NSF grant number 0205519. Our thanks to all!

6. REFERENCES

[1] Amit Aggarwal, Stefan Savage, and Tom Anderson. Understanding the Performance of TCP Pacing. In *IEEE INFOCOM*, March 2000.

[2] Mark Allman. TCP Congestion Control with Appropriate Byte Counting (ABC), February 2003. RFC 3465.

[3] Mark Allman, Sally Floyd, and Craig Partridge. Increasing TCP's Initial Window, October 2002. RFC 3390.

[4] Mark Allman, Vern Paxson, and W. Richard Stevens. TCP Congestion Control, April 1999. RFC 2581.

[5] Jon Bennett, Craig Partridge, and Nicholas Sheetman. Packet Reordering is Not Pathological Network Behavior. *IEEE/ACM Transactions on Networking*, December 1999.

[6] Ethan Blanton and Mark Allman. On the Impact of Bursting on TCP Performance. In *Passive and Active Measurement Workshop*, March 2005.

[7] Robert Braden. Requirements for Internet Hosts – Communication Layers, October 1989. RFC 1122.

[8] Kevin Fall and Sally Floyd. Simulation-based Comparisons of Tahoe, Reno, and SACK TCP. *Computer Communications Review*, 26(3), July 1996.

[9] Sally Floyd and Eddie Kohler. Profile for DCCP Congestion Control ID 2: TCP-like Congestion Control, March 2005. Internet-Draft draft-ietf-dccp-ccid2-10.txt (work in progress).

[10] Mark Handley, Jitendra Padhye, and Sally Floyd. TCP Congestion Window Validation, June 2000. RFC 2861.

[11] Chris Hayes. Analyzing the Performance of New TCP Extensions Over Satellite Links. Master's thesis, Ohio University, August 1997.

[12] Amy Hughes, Joe Touch, and John Heidemann. Issues in TCP Slow-Start Restart After Idle, December 2001. Internet-Draft draft-hughes-restart-00.txt (work in progress).

[13] Van Jacobson. Congestion Avoidance and Control. In *ACM SIGCOMM*, 1988.

[14] Hao Jiang and Constantinos Dovrolis. Source-Level IP Packet Bursts: Causes and Effects. In *ACM SIGCOMM/Usenix Internet Measurement Conference*, October 2003.

[15] Eddie Kohler, Mark Handley, and Sally Floyd. Datagram Congestion Control Protocol (DCCP), March 2005. Internet-Draft draft-ietf-dccp-spec-11.txt (work in progress).

[16] Craig Partridge. ACK Spacing for High Delay-Bandwidth Paths with Insufficient Buffering, September 1998. Internet-Draft draft-rfced-info-partridge-01.txt (work in progress).

[17] Craig Partridge, Dennis Rockwell, Mark Allman, Rajesh Krishnan, and James P.G. Sterbenz. A Swifter Start for TCP. Technical Report TR-8339, BBN Technologies, March 2002.

[18] Vern Paxson. Automated Packet Trace Analysis of TCP Implementations. In *ACM SIGCOMM*, September 1997.

[19] Vern Paxson, Mark Allman, Scott Dawson, William Fenner, Jim Griner, Ian Heavens, Kevin Lahey, Jeff Semke, and Bernie Volz. Known TCP Implementation Problems, March 1999. RFC 2525.

[20] Jon Postel. Transmission Control Protocol, September 1981. RFC 793.

[21] Randall Stewart, Qiaobing Xie, Ken Morneault, Chip Sharp, Hanns Juergen Schwarzbauer, Tom Taylor, Ian Rytina, Malleswar Kalla, Lixia Zhang, and Vern Paxson. Stream Control Transmission Protocol, October 2000. RFC 2960.

[22] Vikram Visweswaraiah and John Heidemann. Improving Restart of Idle TCP Connections. Technical Report 97-661, University of Southern California, August 1997.

[23] Lixia Zhang, Scott Shenker, and David Clark. Observations on the Dynamics of a Congestion Control Algorithm: The Effects of Two- Way Traffic. In *ACM SIGCOMM*, September 1991.

Efficient Security for IPv6 Multihoming

Marcelo Bagnulo
Universidad Carlos III de Madrid
Av. Universidad, 30
Leganés, Madrid, España
+34 916248837
marcelo@it.uc3m.es

Alberto García-Martínez
Universidad Carlos III de Madrid
Av. Universidad, 30
Leganés, Madrid, España
+34 916248782
alberto@it.uc3m.es

Arturo Azcorra
Universidad Carlos III de Madrid
Av. Universidad, 30
Leganés, Madrid, España
+34 916248778
azcorra@it.uc3m.es

ABSTRACT

In this note, we propose a security mechanism for protecting IPv6 networks from possible abuses caused by the malicious usage of a multihoming protocol. In the presented approach, each multihomed node is assigned multiple prefixes from its upstream providers, and it creates the interface identifier part of its addresses by incorporating a cryptographic one-way hash of the available prefix set. The result is that the addresses of each multihomed node form an unalterable set of intrinsically bound IPv6 addresses. This allows any node that is communicating with the multihomed node to securely verify that all the alternative addresses proposed through the multihoming protocol are associated to the address used for establishing the communication. The verification process is extremely efficient because it only involves hash operations.

Categories and Subject Descriptors

C.2.0 [**Computer Communication Networks**] Security and protection.

General Terms

Security

Keywords

IPv6, multihoming, hijacking protection.

1. INTRODUCTION

In order to preserve global routing system scalability, the IPv6 community is advocating the massive adoption of Provider Aggregatable addressing and limiting the assignment of Provider Independent address blocks to the subscribers of the ISPs (i.e. end sites). Such an approach forces multihomed sites, i.e. sites connecting to the Internet through multiple providers, to obtain multiple Provider Aggregatable prefixes, one from each of their provider's address blocks. Moreover, since ISPs only announce their own prefix block into the global routing system, a multihomed host is reachable at a given address only through the corresponding ISP. Consequently, in order to be reachable through all the available ISPs, a multihomed host, i.e. a host within the multihomed site, needs to configure as many addresses as prefixes are available in the multihomed site.

While this setup guarantees the scalability of the multihoming solution, such multi-addressed configuration presents additional difficulties when attempting to provide the fault tolerance capabilities required to a multihoming solution. In particular, the preservation of established communications when an outage affects the provider through which the communication is flowing becomes challenging, since in order to re-home the communication to an alternative ISP, an alternative address must be used to exchange packets. What is more, such adaptation of the addresses used during the lifetime of the communication to the available providers has to be performed in a transparent fashion with respect to transport and application layers, in order to actually preserve the established communication. This is so because current applications and transport layers, such as TCP and UDP, identify the endpoints of a communication through the IP addresses of the nodes involved, implying that the IP addresses selected at the communication establishment time must remain invariant through the lifetime of the communication. Therefore, after an outage, packets need to carry an alternative address, corresponding to an available ISP, in order to be able to reach their destination, but they need to be presented to transport and application layers as if they contain the original address, in order to be recognized as belonging to the established communication. Such an approach requires additional mechanisms in both ends of the communication in order to perform a coherent mapping between the IP addresses presented to the transport and application layers and those addresses actually contained in the packets [1].

This mapping mechanism between addresses used for forwarding packets (also known as locators) and the addresses presented to the upper layers (also known as identifiers) may be vulnerable to redirection attacks [2] if no proper protection is provided. The vulnerability is introduced when an attacker can benefit from the mapping mechanism to induce a victim to believe that he/she is communicating with the owner of a given identifier while he/she is actually exchanging packets with a locator that does not belong to the owner of the identifier. In other words, a redirection attack consists in creating a false mapping between an identifier and a locator. In particular, if no end to end cryptographic integrity protection is used, a redirection attack can result in communication hijacking, allowing the attacker to impersonate one of the parties involved in the communication.

In this note we propose a security mechanism to protect a protocol for preserving established communication through outages in multihomed environments from redirection attacks. The proposed mechanism relies on the capability of generating all the addresses of a multihomed host as an unalterable set of intrinsically bound

IPv6 addresses. In this system, a multihomed host incorporates a cryptographic one-way hash of the prefix-set available in the multihomed site into the interface identifier part of its own addresses, i.e. the lower 64 bits of the IPv6 address. The result is that the binding between all the addresses of a multihomed host is encoded within the addresses themselves, providing hijacking protection. Any party that is communicating with a multihomed node can efficiently verify that the alternative addresses proposed for continuing the communication are bound to the initial address through a simple hash calculation.

The remainder of this paper is organized as follows. In Section 2 we provide some essential background about multihoming and multihoming security. In Section 3 we present the proposed solution, including a detailed security analysis. Next, in Section 4 we present alternative approaches based in the usage of public key cryptography and we compare them with the proposed solution. We finish this note with a section that includes our conclusions.

2. BACKGROUND

In this section we first present a brief overview of a solution for preserving established communications in multihomed environments and then we attempt to identify potential threats to the resulting system.

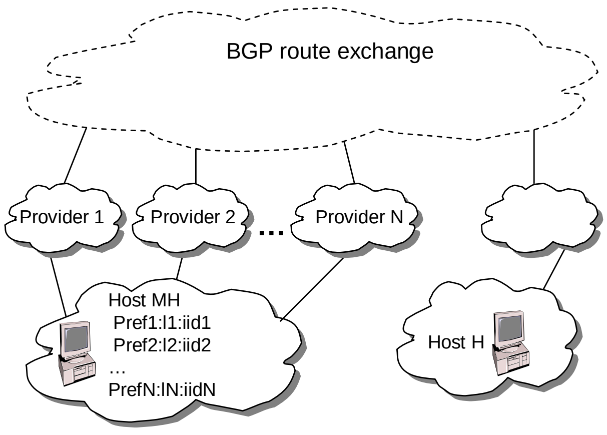


Figure 1. Multihoming with PA addressing.

As we described in the Introduction, a multihomed host that is connected to the Internet through N providers, ISP1,…, ISPN, obtains N prefixes, *Pref1::/n1*,…, *PrefN::/nN*. Consequently, a host located within the multihomed site will have N addresses, one per available ISP/prefix, *Pref1:l1:iid1*, *Pref2:l2:iid2*,…,*PrefN:lN:iidN*, (being *li* the corresponding subnet id as defined in [3]) as presented in figure 1. In order to preserve established communications through outages, a multihoming mechanism located in a shim layer within the IP layer is proposed [4]. The shim layer is located between the IP endpoint sub-layer (that performs end to end functions like fragmenting and IPSec) and the IP routing sub-layer (that performs network related functions like forwarding), as depicted in figure 2. The multihoming mechanism of the shim layer adjusts the address used for exchanging packets according to the available providers, while always presenting a constant address to the upper layers of the stack. The result is that the shim layer performs a mapping between the identifier presented to the upper layers and the locator actually used to exchange packets on the wire. It should be noted that both nodes involved in the communication have to support the mechanism in order to present a coherent view of the addresses involved in the communication. Both ends exchange the information about alternative locators using a multihoming protocol between the shim layers, as presented in figure 2.

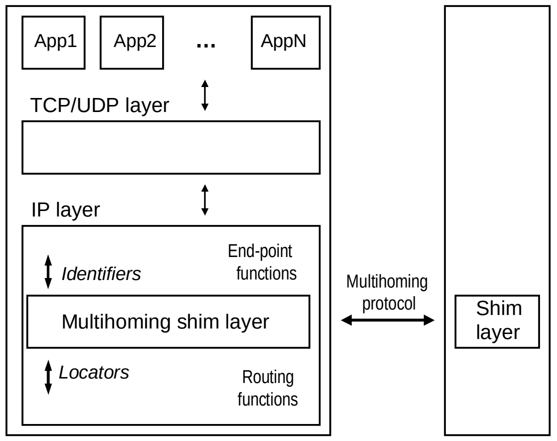


Figure 2. Multihoming protocol layer architecture.

The adoption of such a mechanism enables the possibility of new attacks [2]. The major concern is posed by the possibility of performing so-called *redirection attacks*, where an attacker can persuade a victim to re-home a communication to a locator that is not associated with the identifier used in the communication. There are essentially two types of redirection attack: *hijacking* and *flooding*. In a *hijacking attack* the attacker impersonates one of the parties of the communication. The new hijacking attack enabled by the adoption of a multihoming shim layer is performed by binding the target identifier to the attacker's locator in the multihoming mechanism of the victim node. In this way, when the victim node communicates with the target identifier, it will be actually exchanging packets with the attacker. This is a serious attack, since the attacker is managing to steal the identity of the target node. The shim layer also enables a new type of flooding attack, in which the attacker establishes a communication using its own identifier and then re-homes the communication to a victim's locator. The result is that the victim will be flooded by the flow of the communication initiated by the attacker.

In general terms, it seems wise to require that any additional mechanism introduced to the Internet architecture must, at least, not introduce new vulnerabilities to the network. In this particular case of multihoming, this means that the security of the multihoming solution is not required to protect against man-in-the-middle attacks but it definitely must prevent the so-called *future attacks* [2], also known as *time-shifted attacks* [5], as presented next. In the current Internet, it is clear that any attacker can perform a redirection attack as long as he/she is placed along the path if no additional security measures are adopted. This means it would be somehow acceptable that a multihoming solution is susceptible to man-in-the-middle attacks, since this vulnerability exists in the current Internet. However, in the current Internet, the effects of a hijacking attack are limited to the period during which the attacker is placed along the path. As soon as the attacker leaves his/her on-path location, the attack finishes. In a future attack the attacker launches the attack from an on-path location and then he/she leaves, but the effects of the attack remain long after the attacker has left. Such attacks could be enabled through a multihoming mechanism, since the attacker may only need to be along the path during the time required to add additional locators through the multihoming mechanism. Proper security measures are required to prevent such attacks. In the next, section we present a mechanism to prevent future hijacking attacks to the multihoming protocol.

3. EFFICIENT SECURITY FOR MULTIHOMING

In this section we present the Efficient Security for Multihoming (ESM) architecture to prevent future hijacking attacks and flooding attacks in multihomed environments. The ESM architecture consists of a novel technique to generate a new type of IPv6 addresses called Hash Based Addresses (named HBAs), and a security protocol to protect the multihoming information. In this section we will first describe the algorithm for generating sets of Hash Based Addresses as sets of intrinsically bound IPv6 addresses associated with an arbitrary set of prefixes. Then we will describe how these addresses are used in the ESM protocol for protecting multihoming exchanges of information.

3.1 Hash Based Addresses

We next describe a procedure to generate sets of Hash Based Addresses (HBAs). HBAs are cryptographic in nature, because they contain a one-way hash of the prefix set available in the multihomed site and other parameters in the identifier part of the addresses. In other words, given an arbitrary set of N prefixes, the HBA set generation algorithm produces a HBA set of N addresses. Each one of the generated addresses has a different prefix from the input prefix set, while their interface identifier part contains information about the complete prefix set in the form of a hash of the set. Because of their nature, each address contains information about all the other addresses of the set and a receiver can easily verify if two addresses belong to the same set through an efficient operation such as a hash. After this verification, the receiver can securely use them interchangeably, as we will see in Section 3.2.

So, in order to benefit from the proposed security mechanism, the addresses of each multihomed host have to constitute an HBA set. In a general multihoming scenario as the one presented in Section 2, a multihomed host attached to a link where N 64-bit prefixes [3] are available (*Pref1:l1::/64, Pref2:l2::/64,…, PrefN:lN::/64*) generates the interface identifier part of each one of its addresses as a 64-bit hash of the prefix set available in the link and a random nonce. Including a random nonce enables the generation of multiple HBA sets associated with the same prefix set. The "u" and the "g" bits of the interface identifier are set to zero in order to avoid confusing the resulting addresses with globally unique EUI-64 identifiers [6]. After generating the interface identifier parts, the addresses of the HBA set are finally generated by prepending the different prefixes of the prefix set with the interface identifier parts.

So, summarizing, the procedure for generating an HBA set of N intrinsically bound addresses for a prefix set containing *Pref1:l1::/64, Pref2:l2::/64,…, PrefN:lN::/64* is the following:

First, a 128-bit random nonce *RN* is generated. Proper care should be taken for ensuring randomness as it is discussed in [7]

Second, for each prefix *Prefi:li::/64*, an interface identifier is generated as

iid = `Hash`$_{\texttt{64bit}}$(*RN* , *Pref1:l1* ,…, *PrefN:lN*) (the "u" and the "g" bits are reset)

The corresponding address is generated by prepending the prefix with the interface identifier part (*Prefi:li:iid)*

After generating the address set, the node performs the Duplicate Address Detection procedure as defined in [8]. If any of the addresses collides with an existing one, a new random nonce is generated and a new HBA set is generated.

The output of the described procedure is a HBA set of N addresses that carry information about the prefixes available in the multihomed site within their interface identifier part. The generation procedure is completely automatic, and it does not require any manual configuration, eliminating any administrative burden usually required by security procedures.

It should be noted that it would be possible to generate the addresses of an HBA set with different interface identifiers to provide some privacy features. In order to do that, it is necessary to change the order of the inputs of the hash function when calculating the interface identifiers for each prefix.

3.2 Security Protocol

Once the multihomed host has generated its addresses as a HBA set, we propose the ESM protocol to securely exchange multihoming information. We next describe the ESM protocol using the following notation: MH and H are principals, being MH the multihomed host and H any other host of the Internet. *Pref1:l1:iid*, *Pref2:l2:iid* ,…, *PrefN:lN:iid* are the addresses of MH that were generated as a HBA set. *RN* is the random nonce associated with the HBA set. A_H is the address of H (without loss of generality, we are assuming here that H has a single address for simplicity).

Suppose that MH is communicating with H and that they are using addresses *Prefi:li:iid* and A_H respectively. So far, no multihoming specific features were used; in particular, *Prefi:li:iid* and A_H are used both as identifiers for upper layer protocols (transport and application protocols) and as locators for exchanging packets. In order to benefit from enhanced fault tolerance capabilities provided by multihoming, MH informs H about the alternative addresses available for the communication, so that they can be used in case of an outage.

Since MH's addresses form an HBA set, it is enough for MH to convey the information needed by H to re-generate the HBA set, i.e. the prefix set and the random nonce *RN*.

MH ⟶ H: {*Pref1:l1/64, Pref2:l2/64,…, PrefN:lN/64*}, *RN*

Upon the reception of the HBA set information, H verifies that the address used for establishing the communication, *Prefi:li:iid*, belongs to the HBA set. For that purpose, H first verifies that *Prefi:li::/64* is included in the received prefix set. If this verification is successful, H then verifies that *Prefi:li:iid* corresponds to the address #i of the HBA set generated using the generation algorithm described in the previous section with the received parameters.

Once H has verified that the HBA set associated with the received parameters contains the initial address *Prefi:li:iid*, H generates the full HBA set using the generation algorithm described in the previous section. So far, H is certain that all the alternative addresses available for MH have been generated by the same entity.

Then H verifies that the party located at the alternative address is willing to receive the traffic associated to the established communication at this new location to prevent flooding attacks. To achieve this, a reachability test is included in the ESM protocol that consists of a two-way exchange. The defined packet exchange includes the random nonce *RN* to verify that the same entity is located at the initial and the alternative address. The provision of additional protection against flooding attacks may require tools that are out of the scope of this note.

It should be noted, that the last two operations, i.e. the regeneration of the HBA set and the reachability tests for the alternative locators, do not need to be performed upon the reception of the prefix set, but they can be deferred until an alternative locator is needed because of an outage (which may never occur). Moreover, when an alternative locator, is needed, it is not required to regenerate the whole HBA set, but alternative locators can be regenerated independently using the involved parts of the aforementioned HBA creation procedure.

By means of the ESM protocol, H can securely use any of the addresses of the HBA set interchangeably for exchanging packets in the established communication.

3.3 Security Analysis

3.3.1 Protection from hijacking attacks

The security of the protocol to protect from hijacking attacks is based on the property that any modification of the inputs of the HBA set generation process would result in a different set. Since what is being protected is the mapping between different addresses, producing a valid mapping between other addresses that are not contained in the original set is not an actual threat. We will next illustrate this argument by presenting a possible attack and calculating the effort required to perform it.

In the scenario presented in the previous section, a multihomed host MH is communicating with another host H. MH has generated its addresses through the HBA set generation algorithm, resulting in *Pref1:l1:iid, Pref2:l2:iid ,…, PrefN:lN:iid*. Host H has a single address A_H. MH and H are communicating using addresses *Prefi:li:iid* and A_H respectively. Consider now an attacker X that has the intention of redirecting the communication to an alternative address. We assume that it is enough for the attacker to redirect the communication to any address of a given prefix, *PrefX::/64*. The rationale behind this assumption is that X has access to any address of the considered prefix.

So, in order to hijack the communication, X must introduce a new prefix in the prefix set used for generating the HBA set of MH. For that, X is required to obtain a combination of prefix set and random nonce such as:

1- *Prefi:li/64* and *PrefX::/64* are included in the prefix set

2- *Prefi:li:iid* is included in the resulting HBA set

The other inputs may be changed at will by the attacker; for instance, the random nonce and the other prefixes of the prefix set can be altered. In any case, in order to obtain the desired HBA set, the attacker needs to try with different inputs, for instance with different random nonces, until the above two conditions are met. The expected number of times that the generation procedure needs to be repeated until the desired outcome is reached depends on the number of hash bits included in the interface identifier part of the HBAs. Since we are considering 64-bit long interface identifiers and that the "u" and the "g" bits are not used, the expected number of iterations required for a brute force attack is O(2^61).

In order to quantify the actual effort required to perform such attack, we next calculate the amount of time that is required for an attacker to obtain the proper parameter set. As stated before, the attacker needs to perform O(2^61) hash operations and the corresponding comparisons. Assuming that the attacker uses only two prefixes and the modifier, in order to minimize the amount of data to be hashed, each round the attacker will need to hash 32 bytes. According to `openssl speed`[1], a computer with a Pentium 4 processor (2.66 Ghz) and 440 MB of RAM, can hash 20945 kB per second, when hashing blocks of 32 bytes. This means that it would take approximately 110.000 years to perform the number of hash operations required to obtain the proper parameter set.

We believe that the resulting security is enough for protecting regular traffic, which currently flows unprotected through the network, from potential redirection attacks introduced by the multihoming mechanisms, since the resulting protection is similar to the one offered by other current network security protocols such as the protection provided by Cryptographically Generated Addresses [9] (CGA) in SeND [10].

As processing power increases, the protection provided by this mechanism decreases, since the amount of time required to try with 2^61 different random nonces also decreases. However, additional mechanisms can be used to improve the protection provided by the ESM protocol. For instance artificially increasing the effort required for generating a valid HBA set, similar to the Sec parameter used in CGAs, would result in additional protection.

3.3.2 Protection from flooding attacks

In this section we will present that the usage of the HBA format makes very difficult to launch a flooding attack against a specific address, while it does not prevent from flooding attacks against a specific prefix. Therefore additional protection such as a reachability test is required.

Consider the case where an attacker X has easy access to a prefix *PrefX::/64*. X wants to launch a flooding attack to a host located

[1] `openssl speed` is a command part of the OpenSSL Project, www.openssl.org.

in the address *Prefi:li:iid*. The attack would consist of establishing a communication with a server S and requesting a heavy flow from it. Then simply redirect the flow to *Prefi:li:iid*, flooding the target. In order to perform this attack X needs to generate an HBA set including the prefixes *Prefi:li::/64* and *PrefX*::*/64* in the prefix set. Additionally the resulting HBA set must contain *Prefi:li:iid*. In order to obtain this, the attacker needs to find the appropriate Random Nonce *RN*. The expected number of attempts required to find such RN value is O(2^61). Because of this we can conclude that HBAs provide sufficient protection from this type of attacks.

However, the target of a flooding attack is not limited to specific hosts, but the attack can also be launched against other elements of the infrastructure, such as routers or access links. In order to do that, the attacker can establish a communication with a server S starting the download of a heavy flow. Then, the attacker redirects the communication to any address of the prefix assigned to the target network. Even if the target address is not assigned to any host, the flow will flood the access link of the target site, and the site access router will also suffer the overload. Such attack cannot be prevented using HBAs, since the attacker can easily generate an HBA set using his own prefix and the target network prefix. In order to prevent such attacks, additional mechanisms such as reachability tests are required.

4. COMPARATIVE ANALYSIS

As it is described in Section 2, the goal of the proposed protocol is to secure the binding between the address that is being used as identifier by the upper layer protocols and the alternative addresses that will be used as locators for that communication. There are alternative mechanisms to achieve the same goal. However, we argue that the ESM protocol is the most efficient one, because it does not involve asymmetric key operations and it does not require any infrastructure for key distribution. In this section we present a brief description of some alternative approaches based on public key cryptography, and then we perform a qualitative and a quantitative comparison between the public key based approaches and ESM.

4.1 Alternative Approaches

Several alternative approaches are based on the usage of public key cryptography. Among them we can find Strong Identity Multihoming (SIM) protocol [11], the application of the Host Identity Protocol (HIP) [12] to multihoming [13] and the application of Cryptographic Generated Addresses (CGA) [9] [14] to multihoming [15].

The first two approaches, i.e. SIM and HIP, create a new 128-bit identifier namespace. The new endpoint identifier is the 128-bit hash of the public key of the node. In a CGA based approach, no new identifier namespace is created, but the address of the multihomed node is a CGA that contains a hash of a public key in its interface identifier. In any case, the result is a secure binding between the identifier (whether a new 128-identifier or the CGA) and the associated key pair. This allows the multihomed host to use the corresponding private key to sign the multihomed messages that convey alternative address information. The trust chain in this case is the following: the identifier used for the communication is securely bound to the key pair because it contains the hash of the public key, and the alternative address is bound to the public key through the signature. This approach provides the required protection, since it is invulnerable to *future hijacking attacks*. Additional flooding protection similar to the one used in ESM is still required though.

Other approaches are also possible, but when the address used as an identifier by the upper layers is not intrinsically bound to something else (e.g. a public key or a prefix set), an external trust source is required to provide the binding. This means that a third trusted party, like a public key infrastructure (PKI) is required. Such approaches are extremely difficult to deploy because they require a global infrastructure for key distribution making its application unsuitable for the general case where any two arbitrary nodes in the Internet are communicating. On the other hand, this approach may make sense in a restricted environment where such infrastructure is available for the involved parties. However, this solution would still relay on extensive usage of public key operations. This implies that the computational cost of the operation would be similar to the case of CGAs, that will be shown on section 4.3.

4.2 Qualitative Comparison

The major advantage of a public key based approach with respect to ESM is that they support dynamic address sets. In ESM, the HBA set is determined at the generation moment and cannot be changed afterwards, implying that no new alternative addresses can be added during the communication. In a public key based approach, it is possible to add new alternative locators at any point of the lifetime of the communication, facilitating an integrated mobility-multihoming support. This is due to the additional level of indirection provided by the binding with the key pair. However, the public key based approach requires the intensive usage of public key cryptography, since for each addition of a new alternative address public key operations are performed. On the other hand, the ESM protocol only requires hash operations, which are cheaper, as the results of the quantitative analysis performed in the next section show. This is particularly relevant in some scenarios like highly loaded public servers that maintain thousands of simultaneous communications.

As presented earlier, the different public key based approaches use diverse identifier namespaces. In particular, SIM and HIP create a new identifier namespace while the CGA-based approach use IPv6 addresses as identifier. This difference has a great impact in terms of deployment and backward compatibility. As opposed to IPv6 addresses, the new identifiers used in SIM and HIP cannot be used as locators. This means, that a new directory service that provides a mapping between identifiers and locators is needed to allow endpoints to obtain the locators corresponding to a given identifier. The implementation of such directory service is far from trivial, since the proposed identifier namespace is flat because of its own cryptographic nature. The lack of such directory service results in a poor support of some of the existent applications. For instance, applications that perform referrals or call-backs would simply fail if no identifier-to-locator mapping is available [16]. Because of this limitation, we consider that the SIM and HIP approaches are not directly comparable in quantitative terms to the ESM approach.

On the other hand, a CGA-based approach would indeed provide similar application support. Moreover, it should be noted that the resulting security level is the same in both approaches, since the strength of the resulting binding is determined by the number of bits of the interface identifier part of the IPv6 address. The result is that these two approaches provide similar capabilities and application support, enabling a detailed quantitative comparison between them.

4.3 Quantitative Comparison

In this section we compare the computational cost of the CGA-based approach and the proposed ESM mechanism. We analyze the two major operations involved: the bootstrap process where the elements required in each approach are generated and the establishment of the session context where the alternative locators are exchanged and verified.

4.3.1 Bootstrapping

During the bootstrap process, the elements required for each mechanism are generated. In the case of ESM, the HBA set is generated while in the CGA-based mechanism the set of CGAs is generated. We will next compare the computational effort required in each case.

In the case of ESM, the bootstrap process involves the generation of the HBA set. For this, the generation procedure described in Section 3.1 is executed. The computational effort of such procedure is due to the generation of the 128-bit random nonce *RN* and the posterior hash operation. So, in a configuration where N prefixes are available in the multihomed site, then the computational effort of generating an HBA set is:

$$G_{HBA}(N) = G_{RN} + H(N)$$

being G_{RN} the effort of generating a 128-bit random number and H(N) the effort of computing a hash of N prefixes.

In the case of CGA, the bootstrap process starts with the generation of the public and private key pair. Then, for each of the N prefixes, a random nonce is generated and the address is generated including the hash of the random nonce, the public key and the correspondent prefix. So, in a scenario where there are N prefixes in the multihomed site, the computational effort of generating the CGA addresses is:

$$G_{CGA}(N) = G_{Pkpair} + N * (G_{RN} + H)$$

being G_{Pkpair} the effort of generating a public/private key pair, G_{RN} the effort of generating a 128-bit random nonce and H the effort of hashing the public key, the random nonce and a prefix.

The efforts $G_{HBA}(N)$ and $G_{CGA}(N)$ can be measured and compared in terms of the time required for the computation of the involved operations. We have calculated the values for $G_{HBA}(N)$ and $G_{CGA}(N)$ using the *OpenSSL* tools in a computer with a Pentium 4 processor (2.66 Ghz) and 440 MB of RAM for different values of N (2, 3, 5, 10) and for two different RSA public key lengths (512 and 1024 bits). The hash algorithm used is SHA-1. The results are included in Table 1.

Table 1: Time required for HBA and CGA bootstrapping

N	G_{HBA}	$G_{CGA-512}$	$G_{CGA-1024}$
2	6 μs	31,9 ms	157,8 ms
3	6 μs	31,9 ms	157,8 ms
5	6 μs	31,9 ms	157,8 ms
10	7 μs	31,9 ms	157,8 ms

The above results show that the effort required for bootstrapping the ESM mechanisms is near to 4 orders of magnitude lower that the effort required for bootstrapping a CGA based solution (using 512 bits keys).

However, the bootstrapping process is not frequently executed, so a poor performance is not so critical and cannot be determinant to select the ESM solution over the CGA-based solution. In the next section we analyze the effort of establishing a session context, which is a much more significant operation because of its frequency.

4.3.2 Session Context Establishment

After the bootstrapping, the node is ready to benefit from the capabilities provided by the multihoming solution. This means that when a new communication is established, the node can setup a multihoming session context so that the communication can be preserved through outages. Such session context establishment includes the exchange and validation of the alternative locators for that communication. In this section we will compare the effort required for the verification of the alternative locator set in each one of the analyzed approaches.

As described in section 3.2, in the case of the ESM protocol, the node that receives the alternative locator set of its peer verifies it though the following operation: it first generates the HBA set associated with the received parameter and then it verifies if the address used as identifier for the communication is included in the resulting set. Therefore, the effort of the verification process in a scenario with N prefixes is:

$$V_{HBA}(N) = H(N)$$

being H(N) the effort of computing the hash of N prefixes.

In the case of the CGA-based protocol, the node receiving the alternative locators from its peer will perform two operations: first, it verifies that the received public key corresponds to the CGA used as identifier of the established communication, and next it verifies the signature of the message that contains the alternative locator set. The effort of the verification process with N prefixes is:

$$V_{CGA}(N) = H + S_{PK}(N\text{-}1)$$

being H the effort of computing a hash of the public key and $S_{PK}(N\text{-}1)$ the effort of verifying a signature of a message with N-1 alternative locators.

Similarly to the previous section, we measure the effort in terms of the time required to perform the computations. So, using the same tools and the same infrastructure as in the previous section, we calculate the computational effort for establishing a session context for different values of N (2, 3, 5, 10) and for two different RSA public key length values (512 and 1024 bits). The hash algorithm used is SHA-1. The results are presented in Table 2.

Table 2: Time required for a locator set verification

N	V_{HBA}	$V_{CGA-512}$	$V_{CGA-1024}$
2	1,5 µs	141 µs	425 µs
3	1,5 µs	176 µs	531 µs
5	1,5 µs	244 µs	742 µs
10	1,6 µs	418 µs	1271 µs

The obtained results show that the verification procedure of the ESM approach is at least 2 orders of magnitude more efficient than the CGA based verification process. The session context establishment operation is used extensively by the protocol, since each new communication performs it. So, an increase of a couple of orders of magnitude in this operation seems definitely determinant for selecting a security mechanism for a general purpose multihoming mechanism as the one being consider in this paper.

5. CONCLUSIONS

We have described the ESM architecture, a novel security solution for protecting multihoming mechanisms from redirection attacks. The resulting protection prevents future hijacking attacks and flooding attacks. However, the proposed solution does not provide protection against man-in-the-middle attackers.

The ESM architecture achieves the described protection by defining a new type of addresses called HBAs that contain within the interface identifier a one-way hash of the prefix set available in the multihomed site. This technique allows each multihomed host to create a secure binding between all its available addresses. The resulting binding is easily verifiable by the communicating nodes since it is contained in the addresses themselves. Probably, the most remarkable feature of the proposed solution is that no cryptographic keys or secrets are used in the protocol. Moreover, all the information used in the ESM protocol is exchanged in clear text and can be sniffed and/or spoofed by any attacker without compromising the security. Instead, the security of the protocol is based in the fact that any modification of the parameters used to generate the address would result in a different address, i.e. the attack would not affect the target but an alternative address.

The ESM protocol is extremely efficient because it only involves hash operations; no public key operations are required. In addition, the proposed solution is easy to use, because manual configuration is not required. Compared with a similar approach based on the usage of CGA, the proposed ESM mechanisms is near to 4 orders of magnitude more efficient during the bootstrapping and it is over 2 orders of magnitude more efficient during the lifetime of the communication.

However, it should be noted that the CGA based mechanism provides some features that may be required for certain scenarios, like mobile environments. In this case, it is possible to merge these two approaches into an integrated mechanism by including a hash of both a public key and a prefix set as inputs into the interface identifier of the IPv6 address. For this it is possible to define a new CGA extension that contains the prefix set [17]. The resulting mechanism would allow the usage of the HBA and/or CGA features depending on the situation preserving the enhanced efficiency when the HBA mode is used.

6. ACKNOWLEDGMENTS

The ideas contained in this note benefited from discussions with Jari Arkko, Iljitsch van Beijnum, Geoff Huston, Erik Nordmark, Margaret Wasserman, and Jukka Ylitalo.

Our thanks also go to Huw Oliver, Ignacio Soto, Albert Banchs, Isaías Martínez and the anonymous reviewers for their comments.

This work has been partially supported by the European Union under the E-Next Project FP6506869, and by the Spanish National Research and Development Programme through OPTINET6 (TIC-2003-09042-C03-01) and SAM (TIC2002-04531-C04-03) projects.

7. REFERENCES

[1] Huston, G. *Architectural Approaches to Multi-Homing for IPv6*. Internet Engineering Task Force (IETF), Internet Draft (work in progress), October 2004.

[2] Nordmark, E. and Li, T. *Threats relating to IPv6 multihoming solutions,* Internet Engineering Task Force (IETF), Internet Draft (work in progress), September 2004.

[3] Hinden, R. and S. Deering, *IP Version 6 Addressing Architecture*, Internet Engineering Task Force (IETF), RFC 3513, April 2003.

[4] Nordmark, E. and Bagnulo, M. *Multihoming L3 Shim Approach,* Internet Engineering Task Force (IETF), Internet Draft (work in progress), October 2004.

[5] Nikander, P., Arkko, J., Aura, T., Montenegro, G., Nordmark, E., *Mobile IP version 6 Route Optimization Security Design Background,* Internet Engineering Task Force (IETF), Internet Draft (work in progress), July 2004.

[6] Deering, S. and Hinden, R. *Internet Protocol, Version 6 (IPv6) Specification,* Internet Engineering Task Force (IETF), RFC 2460, December 1998.

[7] Eastlake, D., Crocker, S., Schiller, J. *Randomness Recommendations for Security.* Internet Engineering Task Force (IETF), RFC 1750. December 1994.

[8] Narten, T., Nordmark, E. and Simpson, W. *Neighbor Discovery for IP Version 6 (IPv6),* Internet Engineering Task Force (IETF), RFC 2461, December 1998.

[9] Aura, T., *Cryptographically Generated Addresses (CGA),* Internet Engineering Task Force (IETF), RFC 3972, March 2005.

[10] Arkko, J., Kempf, J., Sommerfeld, B., Zill, B., Nikander, P., *SEcure Neighbor Discovery (SEND),* Internet Engineering Task Force (IETF), RFC 3971, Marzo 2005.

[11] Nordmark, E., *Strong Identity Multihoming using 128 bit Identifiers (SIM/CBID128),* Internet Engineering Task Force (IETF), Internet Draft (work in progress), October 2003.

[12] Moskowitz, R., Nikander, P., Jokela, P., Henderson, T., *Host Identity Protocol*, Internet Engineering Task Force (IETF), Internet Draft (work in progress), October 2004.

[13] Nikander, P., Arkko, J., Henderson, T., *End-Host Mobility and Multi-Homing with Host Identity Protocol*, Internet Engineering Task Force (IETF), Internet Draft (work in progress), October 2004.

[14] O'Shea, G., Roe, M., *Child-proof Authentication for MIPv6, CAM*. ACM Computer Communications Review, 31(2), April 2001.

[15] Nordmark, E., *Multihoming using 64-bit Crypto-based IDs*, Internet Engineering Task Force (IETF), Internet Draft (work in progress), October 2003.

[16] Nordmark, E., *Multi6 Application Referral Issues*, Internet Engineering Task Force (IETF), Internet Draft (work in progress), January 2005.

[17] Bagnulo, M., *Hash Based Addresses (HBA)*, Internet Engineering Task Force (IETF), Internet Draft (work in progress), October 2004.

Vint Cerf and Bob Kahn are this year's recipients of the ACM Turing Award (the highest award in Computer Science) for their pioneering work in internetworking. To commemorate their award, and with the gracious consent of IEEE (which originally published the paper) and the authors, SIGCOMM is reprinting their famous early paper on internetworking, "A Protocol for Packet Network Interconnection," from IEEE Transactions on Communications of May 1974.

When the paper was written, Bob was early in his thirteen year-career at the US Defense Advanced Research Projects agency and Vint was a junior professor at Stanford. Bob eventually left DARPA and has continued in a distinguished career of nurturing research through the not-for-profit Corporation for National Research Initiatives (CNRI), which he founded in 1986 and has led ever since. Vint soon left Stanford to spend several years at DARPA. After DARPA, Vint worked at MCI, leading the development of MCI Mail, then worked with Bob at CNRI, then rejoined MCI in 1994, where he is senior vice president for technology strategy.

Throughout this time, both Vint and Bob have been actively involved in Internet activities. For many years, Vint served on (and for some years, chaired) the Internet Activities Board. Vint is currently chairman of the board of the Internet Corporation for Assigned Names and Numbers (ICANN). Bob, Vint and CNRI have provided essential support services for the Internet Engineering Task Force (IETF) for many years.

Closer to home, both Bob and Vint have served as chairs of SIGCOMM. And both have received the ACM SIGCOMM Award: Bob in 1993 and Vint in 1996.

I continue to find reading this paper intellectually rewarding, both for what it contains and what it does not. What it contains, of course, is a lot of the ideas that eventually became TCP and IP. It is somewhat astonishing to see how much was known in 1974, only 5 years after ARPANET was turned on. Equally interesting are things that were missing, such as congestion coordination with routers (e.g. the TCP congestion window). And there are also a few (not many) ideas that turned out to be bad (such as shrinking the receiver window from the right). All in all a rewarding read and I hope you'll agree, worth reprinting 31 years later!

Craig Partridge
Chief Scientist, BBN Technologies
past-chair, ACM SIGCOMM

acm sigcomm

A Protocol for Packet Network Intercommunication

VINTON G. CERF AND ROBERT E. KAHN, MEMBER, IEEE

***Abstract*—A protocol that supports the sharing of resources that exist in different packet switching networks is presented. The protocol provides for variation in individual network packet sizes, transmission failures, sequencing, flow control, end-to-end error checking, and the creation and destruction of logical process-to-process connections. Some implementation issues are considered, and problems such as internetwork routing, accounting, and timeouts are exposed.**

INTRODUCTION

IN THE LAST few years considerable effort has been expended on the design and implementation of packet switching networks [1]-[7],[14],[17]. A principle reason for developing such networks has been to facilitate the sharing of computer resources. A packet communication network includes a transportation mechanism for delivering data between computers or between computers and terminals. To make the data meaningful, computers and terminals share a common protocol (i.e., a set of agreed upon conventions). Several protocols have already been developed for this purpose [8]-[12],[16]. However, these protocols have addressed only the problem of communication on the same network. In this paper we present a protocol design and philosophy that supports the sharing of resources that exist in different packet switching networks.

After a brief introduction to internetwork protocol issues, we describe the function of a GATEWAY as an interface between networks and discuss its role in the protocol. We then consider the various details of the protocol, including addressing, formatting, buffering, sequencing, flow control, error control, and so forth. We close with a description of an interprocess communication mechanism and show how it can be supported by the internetwork protocol.

Even though many different and complex problems must be solved in the design of an individual packet switching network, these problems are manifestly compounded when dissimilar networks are interconnected. Issues arise which may have no direct counterpart in an individual network and which strongly influence the way in which internetwork communication can take place.

A typical packet switching network is composed of a set of computer resources called HOSTS, a set of one or more *packet switches*, and a collection of communication media that interconnect the packet switches. Within each HOST, we assume that there exist *processes* which must communicate with processes in their own or other HOSTS. Any current definition of a process will be adequate for our purposes [13]. These processes are generally the ultimate source and destination of data in the network. Typically, within an individual network, there exists a protocol for communication between any source and destination process. Only the source and destination processes require knowledge of this convention for communication to take place. Processes in two distinct networks would ordinarily use different protocols for this purpose. The ensemble of packet switches and communication media is called the *packet switching subnet.* Fig. 1 illustrates these ideas.

In a typical packet switching subnet, data of a fixed maximum size are accepted from a source HOST, together with a formatted destination address which is used to route the data in a store and forward fashion. The transmit time for this data is usually dependent upon internal network parameters such as communication media data rates, buffering and signaling strategies, routing, propagation delays, etc. In addition, some mechanism is generally present for error handling and determination of status of the networks components.

Individual packet switching networks may differ in their implementations as follows.

1) Each network may have distinct ways of addressing the receiver, thus requiring that a uniform addressing scheme be created which can be understood by each individual network.

2) Each network may accept data of different maximum size, thus requiring networks to deal in units of the smallest maximum size (which may be impractically small) or requiring procedures which allow data crossing a network boundary to be reformatted into smaller pieces.

3) The success or failure of a transmission and its performance in each network is governed by different time delays in accepting, delivering, and transporting the data. This requires careful development of internetwork timing procedures to insure that data can be successfully delivered through the various networks.

4) Within each network, communication may be disrupted due to unrecoverable mutation of the data or missing data. End-to-end restoration procedures are desirable to allow complete recovery from these conditions.

Paper approved by the Associate Editor for Data Communications of the IEEE Communications Society for publication without oral presentation. Manuscript received November 5, 1973. The research reported in this paper was supported in part by the Advanced Research Projects Agency of the Department of Defense under Contract DAHC 15-73-C-0370.

V. G. Cerf is with the Department of Computer Science and Electrical Engineering, Stanford University, Stanford, Calif.

R. E. Kahn is with the Information Processing Technology Office, Advanced Research Projects Agency, Department of Defense, Arlington, Va.

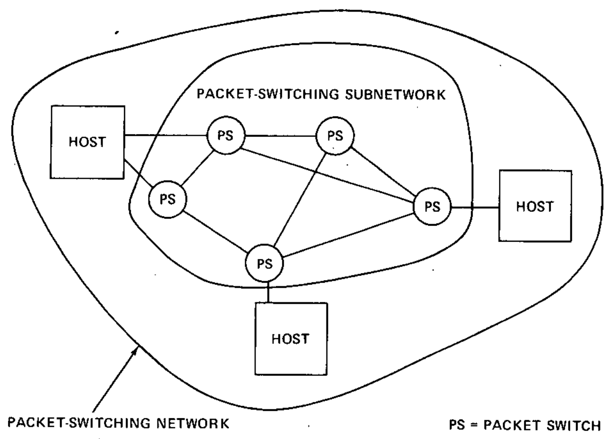


Fig. 1. Typical packet switching network.

5) Status information, routing, fault detection, and isolation are typically different in each network. Thus, to obtain verification of certain conditions, such as an inaccessible or dead destination, various kinds of coordination must be invoked between the communicating networks.

It would be extremely convenient if all the differences between networks could be economically resolved by suitable interfacing at the network boundaries. For many of the differences, this objective can be achieved. However, both economic and technical considerations lead us to prefer that the interface be as simple and reliable as possible and deal primarily with passing data between networks that use different packet switching strategies.

The question now arises as to whether the interface ought to account for differences in HOST or process level protocols by transforming the source conventions into the corresponding destination conventions. We obviously want to allow conversion between packet switching strategies at the interface, to permit interconnection of existing and planned networks. However, the complexity and dissimilarity of the HOST or process level protocols makes it desirable to avoid having to transform between them at the interface, even if this transformation were always possible. Rather, compatible HOST and process level protocols must be developed to achieve effective internetwork resource sharing. The unacceptable alternative is for every HOST or process to implement every protocol (a potentially unbounded number) that may be needed to communicate with other networks. We therefore assume that a common protocol is to be used between HOST's or processes in different networks and that the interface between networks should take as small a role as possible in this protocol.

To allow networks under different ownership to interconnect, some accounting will undoubtedly be needed for traffic that passes across the interface. In its simplest terms, this involves an accounting of packets handled by each net for which charges are passed from net to net until the buck finally stops at the user or his representative. Furthermore, the interconnection must preserve intact the internal operation of each individual network. This is easily achieved if two networks interconnect as if each were a HOST to the other network, but without utilizing or indeed incorporating any elaborate HOST protocol transformations.

It is thus apparent that the interface between networks must play a central role in the development of any network interconnection strategy. We give a special name to this interface that performs these functions and call it a GATEWAY.

THE GATEWAY NOTION

In Fig. 2 we illustrate three individual networks labeled A, B, and C which are joined by GATEWAYS M and N. GATEWAY M interfaces network A with network B, and GATEWAY N interfaces network B to network C. We assume that an individual network may have more than one GATEWAY (e.g., network B) and that there may be more than one GATEWAY path to use in going between a pair of networks. The responsibility for properly routing data resides in the GATEWAY.

In practice, a GATEWAY between two networks may be composed of two halves, each associated with its own network. It is possible to implement each half of a GATEWAY so it need only embed internetwork packets in local packet format or extract them. We propose that the GATEWAYS handle internetwork packets in a standard format, but we are not proposing any particular transmission procedure between GATEWAY halves.

Let us now trace the flow of data through the interconnected networks. We assume a packet of data from process X enters network A destined for process Y in network C. The address of Y is initially specified by process X and the address of GATEWAY M is derived from the address of process Y. We make no attempt to specify whether the choice of GATEWAY is made by process X, its HOST, or one of the packet switches in network A. The packet traverses network A until it reaches GATEWAY M. At the GATEWAY, the packet is reformatted to meet the requirements of network B, account is taken of this unit of flow between A and B, and the GATEWAY delivers the packet to network B. Again the derivation of the next GATEWAY address is accomplished based on the address of the destination Y. In this case, GATEWAY N is the next one. The packet traverses network B until it finally reaches GATEWAY N where it is formatted to meet the requirements of network C. Account is again taken of this unit of flow between networks B and C. Upon entering network C the packet is routed to the HOST in which process Y resides and there it is delivered to its ultimate destination.

Since the GATEWAY must understand the address of the source and destination HOSTS, this information must be available in a standard format in every packet which arrives at the GATEWAY. This information is contained in an *internetwork header* prefixed to the packet by the source HOST. The packet format, including the internet

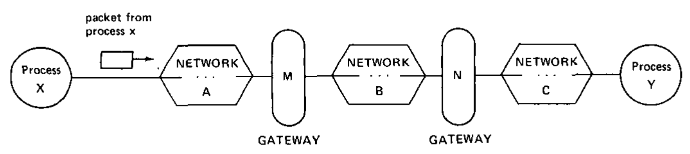


Fig. 2. Three networks interconnected by two GATEWAYS.

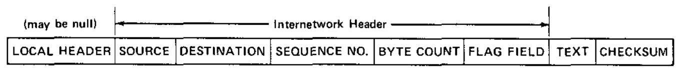


Fig. 3. Internetwork packet format (fields not shown to scale).

work header, is illustrated in Fig. 3. The source and destination entries uniformly and uniquely identify the address of every HOST in the composite network. Addressing is a subject of considerable complexity which is discussed in greater detail in the next section. The next two entries in the header provide a sequence number and a byte count that may be used to properly sequence the packets upon delivery to the destination and may also enable the GATEWAYS to detect fault conditions affecting the packet. The flag field is used to convey specific control information and is discussed in the section on retransmission and duplicate detection later. The remainder of the packet consists of text for delivery to the destination and a trailing check sum used for end-to-end software verification. The GATEWAY does *not* modify the text and merely forwards the check sum along without computing or recomputing it.

Each network may need to augment the packet format before it can pass through the individual network. We have indicated a *local header* in the figure which is prefixed to the beginning of the packet. This local header is introduced merely to illustrate the concept of embedding an internetwork packet in the format of the individual network through which the packet must pass. It will obviously vary in its exact form from network to network and may even be unnecessary in some cases. Although not explicitly indicated in the figure, it is also possible that a local trailer may be appended to the end of the packet.

Unless all transmitted packets are legislatively restricted to be small enough to be accepted by every individual network, the GATEWAY may be forced to split a packet into two or more smaller packets. This action is called fragmentation and must be done in such a way that the destination is able to piece together the fragmented packet. It is clear that the internetwork header format imposes a minimum packet size which all networks must carry (obviously all networks will want to carry packets larger than this minimum). We believe the long range growth and development of internetwork communication would be seriously inhibited by specifying how much larger than the minimum a packet size can be, for the following reasons.

1) If a maximum permitted packet size is specified then it becomes impossible to completely isolate the internal packet size parameters of one network from the internal packet size parameters of all other networks.

2) It would be very difficult to increase the maximum permitted packet size in response to new technology (e.g., large memory systems, higher data rate communication facilities, etc.) since this would require the agreement and then implementation by all participating networks.

3) Associative addressing and packet encryption may require the size of a particular packet to expand during transit for incorporation of new information.

Provision for fragmentation (regardless of where it is performed) permits packet size variations to be handled on an individual network basis without global administration and also permits HOSTS and processes to be insulated from changes in the packet sizes permitted in any networks through which their data must pass.

If fragmentation must be done, it appears best to do it upon entering the next network at the GATEWAY since only this GATEWAY (and not the other networks) must be aware of the internal packet size parameters which made the fragmentation necessary.

If a GATEWAY fragments an incoming packet into two or more packets, they must eventually be passed along to the destination HOST as fragments or reassembled for the HOST. It is conceivable that one might desire the GATEWAY to perform the reassembly to simplify the task of the destination HOST (or process) and/or to take advantage of a larger packet size. We take the position that GATEWAYS should not perform this function since GATEWAY reassembly can lead to serious buffering problems, potential deadlocks, the necessity for all fragments of a packet to pass through the same GATEWAY, and increased delay in transmission. Furthermore, it is not sufficient for the GATEWAYS to provide this function since the final GATEWAY may also have to fragment a packet for transmission. Thus the destination HOST must be prepared to do this task.

Let us now turn briefly to the somewhat unusual accounting effect which arises when a packet may be fragmented by one or more GATEWAYS. We assume, for simplicity, that each network initially charges a fixed rate per packet transmitted, regardless of distance, and if one network can handle a larger packet size than another, it charges a proportionally larger price per packet. We also assume that a subsequent increase in any network's packet size does not result in additional cost per packet to its users. The charge to a user thus remains basically constant through any net which must fragment a packet. The unusual effect occurs when a packet is fragmented into smaller packets which must individually pass through a subsequent network with a larger packet size than the original unfragmented packet. We expect that most networks will naturally select packet sizes close to one another, but in any case, an increase in packet size in one net, even when it causes fragmentation, will not increase the cost of transmission and may actually decrease it. In the event that any other packet charging policies (than the one we suggest) are adopted, differences in cost can be used as an economic lever toward optimization of individual network performance.

PROCESS LEVEL COMMUNICATION

We suppose that processes wish to communicate in full duplex with their correspondents using unbounded but finite length messages. A single character might constitute the text of a message from a process to a terminal or vice versa. An entire page of characters might constitute the text of a message from a file to a process. A data stream (e.g., a continuously generated bit string) can be represented as a sequence of finite length messages.

Within a HOST we assume the existence of a transmission control program (TCP) which handles the transmission and acceptance of messages on behalf of the processes it serves. The TCP is in turn served by one or more packet switches connected to the HOST in which the TCP resides. Processes that want to communicate present messages to the TCP for transmission, and TCP's deliver incoming messages to the appropriate destination processes. We allow the TCP to break up messages into segments because the destination may restrict the amount of data that may arrive, because the local network may limit the maximum transmission size, or because the TCP may need to share its resources among many processes concurrently. Furthermore, we constrain the length of a segment to an integral number of 8-bit bytes. This uniformity is most helpful in simplifying the software needed with HOST machines of different natural word lengths. Provision at the process level can be made for padding a message that is not an integral number of bytes and for identifying which of the arriving bytes of text contain information of interest to the receiving process.

Multiplexing and demultiplexing of segments among processes are fundamental tasks of the TCP. On transmission, a TCP must multiplex together segments from different source processes and produce internetwork packets for delivery to one of its serving packet switches. On reception, a TCP will accept a sequence of packets from its serving packet switch(es). From this sequence of arriving packets (generally from different HOSTS), the TCP must be able to reconstruct and deliver messages to the proper destination processes.

We assume that every segment is augmented with additional information that allows transmitting and receiving TCP's to identify destination and source processes, respectively. At this point, we must face a major issue. How should the source TCP format segments destined for the same destination TCP? We consider two cases.

Case 1): If we take the position that segment boundaries are immaterial and that a byte stream can be formed of segments destined for the same TCP, then we may gain improved transmission efficiency and resource sharing by arbitrarily parceling the stream into packets, permitting many segments to share a single internetwork packet header. However, this position results in the need to reconstruct exactly, and in order, the stream of text bytes produced by the source TCP. At the destination, this stream must first be parsed into segments and these in turn must be used to reconstruct messages for delivery to the appropriate processes.

There are fundamental problems associated with this strategy due to the possible arrival of packets out of order at the destination. The most critical problem appears to be the amount of interference that processes sharing the same TCP–TCP byte stream may cause among themselves. This is especially so at the receiving end. First, the TCP may be put to some trouble to parse the stream back into segments and then distribute them to buffers where messages are reassembled. If it is not readily apparent that all of a segment has arrived (remember, it may come as several packets), the receiving TCP may have to suspend parsing temporarily until more packets have arrived. Second, if a packet is missing, it may not be clear whether succeeding segments, even if they are identifiable, can be passed on to the receiving process, unless the TCP has knowledge of some process level sequencing scheme. Such knowledge would permit the TCP to decide whether a succeeding segment could be delivered to its waiting process. Finding the beginning of a segment when there are gaps in the byte stream may also be hard.

Case 2): Alternatively, we might take the position that the destination TCP should be able to determine, upon its arrival and without additional information, for which process or processes a received packet is intended, and if so, whether it should be delivered then.

If the TCP is to determine for which process an arriving packet is intended, every packet must contain a *process header* (distinct from the internetwork header) that completely identifies the destination process. For simplicity, we assume that each packet contains text from a single process which is destined for a single process. Thus each packet need contain only one process header. To decide whether the arriving data is deliverable to the destination process, the TCP must be able to determine whether the data is in the proper sequence (we can make provision for the destination process to instruct its TCP to ignore sequencing, but this is considered a special case). With the assumption that each arriving packet contains a process header, the necessary sequencing and destination process identification is immediately available to the destination TCP.

Both Cases 1) and 2) provide for the demultiplexing and delivery of segments to destination processes, but only Case 2) does so without the introduction of potential interprocess interference. Furthermore, Case 1) introduces extra machinery to handle flow control on a HOST-to-HOST basis, since there must also be some provision for process level control, and this machinery is little used since the probability is small that within a given HOST, two processes will be coincidentally scheduled to send messages to the same destination HOST. For this reason, we select the method of Case 2) as a part of the *internetwork transmission protocol.*

ADDRESS FORMATS

The selection of address formats is a problem between networks because the local network addresses of TCP's may vary substantially in format and size. A uniform internetwork TCP address space, understood by each GATEWAY and TCP, is essential to routing and delivery of internetwork packets.

Similar troubles are encountered when we deal with process addressing and, more generally, port addressing. We introduce the notion of *ports* in order to permit a process to distinguish between multiple message streams. The port is simply a designator of one such message stream associated with a process. The means for identifying a port are generally different in different operating systems, and therefore, to obtain uniform addressing, a standard port address format is also required. A port address designates a full duplex message stream.

TCP ADDRESSING

TCP addressing is intimately bound up in routing issues, since a HOST or GATEWAY must choose a suitable destination HOST or GATEWAY for an outgoing internetwork packet. Let us postulate the following address format for the TCP address (Fig. 4). The choice for network identification (8 bits) allows up to 256 distinct networks. This size seems sufficient for the forseeable future. Similarly, the TCP identifier field permits up to 65 536 distinct TCP's to be addressed, which seems more than sufficient for any given network.

As each packet passes through a GATEWAY, the GATEWAY observes the destination network ID to determine how to route the packet. If the destination network is connected to the GATEWAY, the lower 16 bits of the TCP address are used to produce a local TCP address in the destination network. If the destination network is not connected to the GATEWAY, the upper 8 bits are used to select a subsequent GATEWAY. We make no effort to specify how each individual network shall associate the internetwork TCP identifier with its local TCP address. We also do not rule out the possibility that the local network understands the internetwork addressing scheme and thus alleviates the GATEWAY of the routing responsibility.

PORT ADDRESSING

A receiving TCP is faced with the task of demultiplexing the stream of internetwork packets it receives and reconstructing the original messages for each destination process. Each operating system has its own internal means of identifying processes and ports. We assume that 16 bits are sufficient to serve as internetwork port identifiers. A sending process need not know how the destination port identification will be used. The destination TCP will be able to parse this number appropriately to find the proper buffer into which it will place arriving packets. We permit a large port number field to support processes which want to distinguish between many different messages streams concurrently. In reality, we do not care how the 16 bits are sliced up by the TCP's involved.

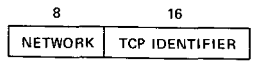


Fig. 4. TCP address.

Even though the transmitted port name field is large, it is still a compact external name for the internal representation of the port. The use of short names for port identifiers is often desirable to reduce transmission overhead and possibly reduce packet processing time at the destination TCP. Assigning short names to each port, however, requires an initial negotiation between source and destination to agree on a suitable short name assignment, the subsequent maintenance of conversion tables at both the source and the destination, and a final transaction to release the short name. For dynamic assignment of port names, this negotiation is generally necessary in any case.

SEGMENT AND PACKET FORMATS

As shown in Fig. 5, messages are broken by the TCP into segments whose format is shown in more detail in Fig. 6. The field lengths illustrated are merely suggestive. The first two fields (source port and destination port in the figure) have already been discussed in the preceding section on addressing. The uses of the third and fourth fields (window and acknowledgment in the figure) will be discussed later in the section on retransmission and duplicate detection.

We recall from Fig. 3 that an internetwork header contains both a sequence number and a byte count, as well as a flag field and a check sum. The uses of these fields are explained in the following section.

REASSEMBLY AND SEQUENCING

The reconstruction of a message at the receiving TCP clearly requires[1] that each internetwork packet carry a sequence number which is unique to its particular destination port message stream. The sequence numbers must be monotonic increasing (or decreasing) since they are used to reorder and reassemble arriving packets into a message. If the space of sequence numbers were infinite, we could simply assign the next one to each new packet. Clearly, this space cannot be infinite, and we will consider what problems a finite sequence number space will cause when we discuss retransmission and duplicate detection in the next section. We propose the following scheme for performing the sequencing of packets and hence the reconstruction of messages by the destination TCP.

A pair of ports will exchange one or more messages over a period of time. We could view the sequence of messages produced by one port as if it were embedded in an infinitely long stream of bytes. Each byte of the message has a unique sequence number which we take to be its byte location relative to the beginning of the stream. When a

[1] In the case of encrypted packets, a preliminary stage of reassembly may be required prior to decryption.

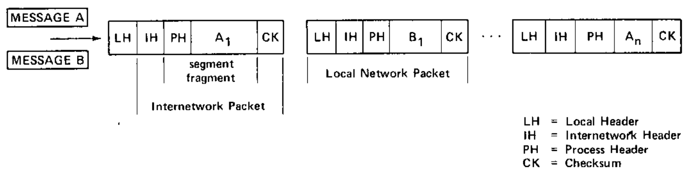


Fig. 5. Creation of segments and packets from messages.

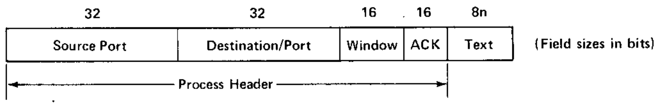


Fig. 6. Segment format (process header and text).

segment is extracted from the message by the source TCP and formatted for internetwork transmission, the relative location of the first byte of segment text is used as the sequence number for the packet. The byte count field in the internetwork header accounts for all the text in the segment (but does not include the check-sum bytes or the bytes in either internetwork or process header). We emphasize that the sequence number associated with a given packet is unique only to the pair of ports that are communicating (see Fig. 7). Arriving packets are examined to determine for which port they are intended. The sequence numbers on each arriving packet are then used to determine the relative location of the packet text in the messages under reconstruction. We note that this allows the exact position of the data in the reconstructed message to be determined even when pieces are still missing.

Every segment produced by a source TCP is packaged in a single internetwork packet and a check sum is computed over the text and process header associated with the segment.

The splitting of messages into segments by the TCP and the potential splitting of segments into smaller pieces by GATEWAYS creates the necessity for indicating to the destination TCP when the end of a segment (ES) has arrived and when the end of a message (EM) has arrived. The flag field of the internetwork header is used for this purpose (see Fig. 8).

The ES flag is set by the source TCP each time it prepares a segment for transmission. If it should happen that the message is completely contained in the segment, then the EM flag would also be set. The EM flag is also set on the last segment of a message, if the message could not be contained in one segment. These two flags are used by the destination TCP, respectively, to discover the presence of a check sum for a given segment and to discover that a complete message has arrived.

The ES and EM flags in the internetwork header are known to the GATEWAY and are of special importance when packets must be split apart for propagation through the next local network. We illustrate their use with an example in Fig. 9.

The original message A in Fig. 9 is shown split into two segments A_1 and A_2 and formatted by the TCP into a pair

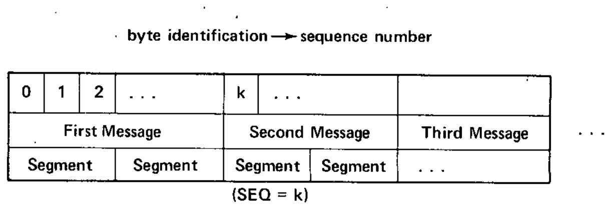


Fig. 7. Assignment of sequence numbers.

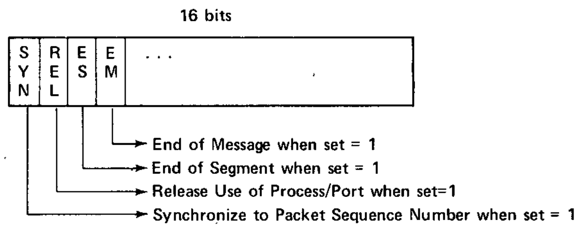

16 bits
SYN
REL
ES
EM
. . .
End of Message when set = 1
End of Segment when set = 1
Release Use of Process/Port when set=1
Synchronize to Packet Sequence Number when set = 1

Fig. 8. Internetwork header flag field.

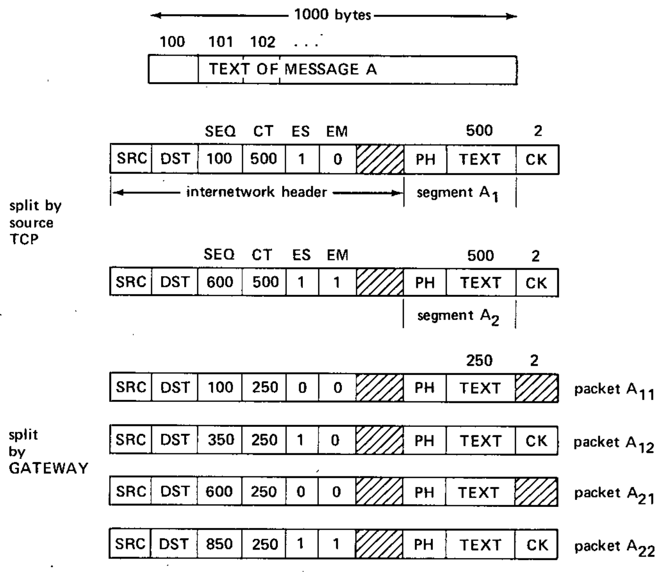


Fig. 9. Message splitting and packet splitting.

of internetwork packets. Packets A_1 and A_2 have the ES bits set, and A_2 has its EM bit set as well. Whe packet A_1 passes through the GATEWAY, it is split into tw pieces: packet A_{11} for which neither EM nor ES bits a set, and packet A_{12} whose ES bit is set. Similarly, pack A_2 is split such that the first piece, packet A_{21}, has neith bit set, but packet A_{22} has both bits set. The sequen number field (SEQ) and the byte count field (CT) of eac packet is modified by the GATEWAY to properly identi the text bytes of each packet. The GATEWAY need onl examine the internetwork header to do fragmentation.

The destination TCP, upon reassembling segment A will detect the ES flag and will verify the check sum knows is contained in packet A_{12}. Upon receipt of pack A_{22}, assuming all other packets have arrived, the dest nation TCP detects that it has reassembled a comple message and can now advise the destination process of i receipt.

RETRANSMISSION AND DUPLICATE DETECTION

No transmission can be 100 percent reliable. We propose a timeout and positive acknowledgment mechanism which will allow TCP's to recover from packet losses from one HOST to another. A TCP transmits packets and waits for replies (acknowledgements) that are carried in the reverse packet stream. If no acknowledgment for a particular packet is received, the TCP will retransmit. It is our expectation that the HOST level retransmission mechanism, which is described in the following paragraphs, will not be called upon very often in practice. Evidence already exists[2] that individual networks can be effectively constructed without this feature. However, the inclusion of a HOST retransmission capability makes it possible to recover from occasional network problems and allows a wide range of HOST protocol strategies to be incorporated. We envision it will occasionally be invoked to allow HOST accommodation to infrequent overdemands for limited buffer resources, and otherwise not used much.

Any retransmission policy requires some means by which the receiver can detect duplicate arrivals. Even if an infinite number of distinct packet sequence numbers were available, the receiver would still have the problem of knowing how long to remember previously received packets in order to detect duplicates. Matters are complicated by the fact that only a finite number of distinct sequence numbers are in fact available, and if they are reused, the receiver must be able to distinguish between new transmissions and retransmissions.

A *window* strategy, similar to that used by the French CYCLADES system (voie virtuelle transmission mode [8]) and the ARPANET very distant HOST connection [18], is proposed here (see Fig. 10).

Suppose that the sequence number field in the internetwork header permits sequence numbers to range from 0 to $n - 1$. We assume that the sender will not transmit more than w bytes without receiving an acknowledgment. The w bytes serve as the window (see Fig. 11). Clearly, w must be less than n. The rules for sender and receiver are as follows.

Sender: Let L be the sequence number associated with the left window edge.

1) The sender transmits bytes from segments whose text lies between L and up to $L + w - 1$.

2) On timeout (duration unspecified), the sender retransmits unacknowledged bytes.

3) On receipt of acknowledgment consisting of the receiver's current left window edge, the sender's left window edge is advanced over the acknowledged bytes (advancing the right window edge implicitly).

Receiver:

1) Arriving packets whose sequence numbers coincide with the receiver's current left window edge are acknowledged by sending to the source the next sequence number expected. This effectively acknowledges bytes in between. The left window edge is advanced to the next sequence number expected.

2) Packets arriving with a sequence number to the left of the window edge (or, in fact, outside of the window) are discarded, and the current left window edge is returned as acknowledgment.

3) Packets whose sequence numbers lie within the receiver's window but do not coincide with the receiver's left window edge are optionally kept or discarded, but are not acknowledged. This is the case when packets arrive out of order.

We make some observations on this strategy. First, all computations with sequence numbers and window edges must be made modulo n (e.g., byte 0 follows byte $n - 1$). Second, w must be less than $n/2$[3]; otherwise a retransmission may appear to the receiver to be a new transmission in the case that the receiver has accepted a window's worth of incoming packets, but all acknowledgments have been lost. Third, the receiver can either save or discard arriving packets whose sequence numbers do not coincide with the receiver's left window. Thus, in the simplest implementation, the receiver need not buffer more than one packet per message stream if space is critical. Fourth, multiple packets can be acknowledged simultaneously. Fifth, the receiver is able to deliver messages to processes in their proper order as a natural result of the reassembly mechanism. Sixth, when duplicates are detected, the acknowledgment method used naturally works to resynchronize sender and receiver. Furthermore, if the receiver accepts packets whose sequence numbers lie within the current window but

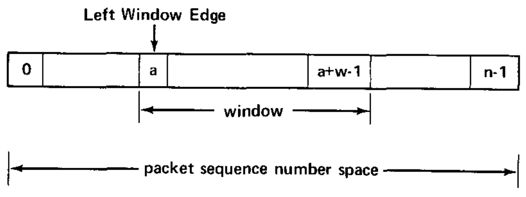


Fig. 10. The window concept.

1	Source Address	
2	Destination Address	
3	Next Packet Seq.	
4	Current Buffer Size	
5	Next Write Position	
6	Next Read Position	
7	End Read Position	
8	No. Retrans.	Max Retrans.
9	Timeout	Flags
10	Curr. Ack	Window

Fig. 11. Conceptual TCB format.

[2] The ARPANET is one such example.

[3] Actually $n/2$ is merely a convenient number to use; it is only required that a retransmission not appear to be a new transmission.

which are not coincident with the left window edge, an acknowledgment consisting of the current left window edge would act as a stimulus to cause retransmission of the unacknowledged bytes. Finally, we mention an overlap problem which results from retransmission, packet splitting, and alternate routing of packets through different GATEWAYS.

A 600-byte packet might pass through one GATEWAY and be broken into two 300-byte packets. On retransmission, the same packet might be broken into three 200-byte packets going through a different GATEWAY. Since each byte has a sequence number, there is no confusion at the receiving TCP. We leave for later the issue of initially synchronizing the sender and receiver left window edges and the window size.

FLOW CONTROL

Every segment that arrives at the destination TCP is ultimately acknowledged by returning the sequence number of the next segment which must be passed to the process (it may not yet have arrived).

Earlier we described the use of a sequence number space and window to aid in duplicate detection. Acknowledgments are carried in the process header (see Fig. 6) and along with them there is provision for a "suggested window" which the receiver can use to control the flow of data from the sender. This is intended to be the main component of the process flow control mechanism. The receiver is free to vary the window size according to any algorithm it desires so long as the window size never exceeds half the sequence number space.[3]

This flow control mechanism is exceedingly powerful and flexible and does not suffer from synchronization troubles that may be encountered by incremental buffer allocation schemes [9],[10]. However, it relies heavily on an effective retransmission strategy. The receiver can reduce the window even while packets are en route from the sender whose window is presently larger. The net effect of this reduction will be that the receiver may discard incoming packets (they may be outside the window) and reiterate the current window size along with a current window edge as acknowledgment. By the same token, the sender can, upon occasion, choose to send more than a window's worth of data on the possibility that the receiver will expand the window to accept it (of course, the sender must not send more than half the sequence number space at any time). Normally, we would expect the sender to abide by the window limitation. Expansion of the window by the receiver merely allows more data to be accepted. For the receiving HOST with a small amount of buffer space, a strategy of discarding all packets whose sequence numbers do not coincide with the current left edge of the window is probably necessary, but it will incur the expense of extra delay and overhead for retransmission.

TCP INPUT/OUTPUT HANDLING

The TCP has a component which handles input/output (I/O) to and from the network.[4] When a packet has arrived, it validates the addresses and places the packet on a queue. A pool of buffers can be set up to handle arrivals, and if all available buffers are used up, succeeding arrivals can be discarded since unacknowledged packets will be retransmitted.

On output, a smaller amount of buffering is needed, since process buffers can hold the data to be transmitted. Perhaps double buffering will be adequate. We make no attempt to specify how the buffering should be done except to require that it be able to service the network with as little overhead as possible. Packet sized buffers one or more ring buffers, or any other combination are possible candidates.

When a packet arrives at the destination TCP, it is placed on a queue which the TCP services frequently. For example, the TCP could be interrupted when a queue placement occurs. The TCP then attempts to place the packet text into the proper place in the appropriate process receive buffer. If the packet terminates a segment, then it can be checksummed and acknowledged. Placement may fail for several reasons.

1) The destination process may not be prepared to receive from the stated source, or the destination port ID may not exist.

2) There may be insufficient buffer space for the text.

3) The beginning sequence number of the text may not coincide with the next sequence number to be delivered to the process (e.g., the packet has arrived out of order).

In the first case, the TCP should simply discard the packet (thus far, no provision has been made for error acknowledgments). In the second and third cases, the packet sequence number can be inspected to determine whether the packet text lies within the legitimate window for reception. If it does, the TCP may optionally keep the packet queued for later processing. If not, the TCP can discard the packet. In either case the TCP can optionally acknowledge with the current left window edge.

It may happen that the process receive buffer is not present in the active memory of the HOST, but is stored on secondary storage. If this is the case, the TCP can prompt the scheduler to bring in the appropriate buffer and the packet can be queued for later processing.

If there are no more input buffers available to the TCP for temporary queueing of incoming packets, and if the TCP cannot quickly use the arriving data (e.g., a TCP to TCP message), then the packet is discarded. Assuming a sensibly functioning system, no other processes than the one for which the packet was intended should be affected by this discarding. If the delayed processing queue grows

[4] This component can serve to handle other protocols whose associated control programs are designated by internetwork destination address.

excessively long, any packets in it can be safely discarded since none of them have yet been acknowledged. Congestion at the TCP level is flexibly handled owing to the robust retransmission and duplicate detection strategy.

TCP/PROCESS COMMUNICATION

In order to send a message, a process sets up its text in a buffer region in its own address space, inserts the requisite control information (described in the following list) in a transmit control block (TCB) and passes control to the TCP. The exact form of a TCB is not specified here, but it might take the form of a passed pointer, a pseudointerrupt, or various other forms. To receive a message in its address space, a process sets up a receive buffer, inserts the requisite control information in a receive control block (RCB) and again passes control to the TCP.

In some simple systems, the buffer space may in fact be provided by the TCP. For simplicity we assume that a ring buffer is used by each process, but other structures (e.g., buffer chaining) are not ruled out.

A possible format for the TCB is shown in Fig. 11. The TCB contains information necessary to allow the TCP to extract and send the process data. Some of the information might be implicitly known, but we are not concerned with that level of detail. The various fields in the TCB are described as follows.

1) *Source Address*: This is the full net/HOST/TCP/port address of the transmitter.

2) *Destination Address*: This is the full net/HOST/TCP/port of the receiver.

3) *Next Packet Sequence Number*: This is the sequence number to be used for the next packet the TCP will transmit from this port.

4) *Current Buffer Size*: This is the present size of the process transmit buffer.

5) *Next Write Position*: This is the address of the next position in the buffer at which the process can place new data for transmission.

6) *Next Read Position*: This is the address at which the TCP should begin reading to build the next segment for output.

7) *End Read Position*: This is the address at which the TCP should halt transmission. Initially 6) and 7) bound the message which the process wishes to transmit.

8) *Number of Retransmissions/Maximum Retransmissions*: These fields enable the TCP to keep track of the number of times it has retransmitted the data and could be omitted if the TCP is not to give up.

9) *Timeout/Flags*: The timeout field specifies the delay after which unacknowledged data should be retransmitted. The flag field is used for semaphores and other TCP/process synchronization, status reporting, etc.

10) *Current Acknowledgment/Window*: The current acknowledgment field identifies the first byte of data still unacknowledged by the destination TCP.

The read and write positions move circularly around the transmit buffer, with the write position always to the left (module the buffer size) of the read position.

The next packet sequence number should be constrained to be less than or equal to the sum of the current acknowledgment and the window fields. In any event, the next sequence number should not exceed the sum of the current acknowledgment and half of the maximum possible sequence number (to avoid confusing the receiver's duplicate detection algorithm). A possible buffer layout is shown in Fig. 12.

The RCB is substantially the same, except that the end read field is replaced by a partial segment check-sum register which permits the receiving TCP to compute and remember partial check sums in the event that a segment arrives in several packets. When the final packet of the segment arrives, the TCP can verify the check sum and if successful, acknowledge the segment.

CONNECTIONS AND ASSOCIATIONS

Much of the thinking about process-to-process communication in packet switched networks has been influenced by the ubiquitous telephone system. The HOST–HOST protocol for the ARPANET deals explicitly with the opening and closing of simplex connections between processes [9],[10]. Evidence has been presented that message-based "connection-free" protocols can be constructed [12], and this leads us to carefully examine the notion of a connection.

The term *connection* has a wide variety of meanings. It can refer to a physical or logical path between two entities, it can refer to the flow over the path, it can inferentially refer to an action associated with the setting up of a path, or it can refer to an association between two or more entities, with or without regard to any path between them. In this paper, we do not explicitly reject the term connection, since it is in such widespread use, and does connote a meaningful relation, but consider it exclusively in the sense of an association between two or more entities without regard to a path. To be more precise about our intent, we shall define the relationship between two or more ports that are in communication, or are prepared to communicate to be an *association*. Ports that are associated with each other are called *associates*.

It is clear that for any communication to take place between two processes, one must be able to address the other. The two important cases here are that the destination port may have a global and unchanging address or that it may be globally unique but dynamically reassigned. While in either case the sender may have to learn the destination address, given the destination name, only in the second instance is there a requirement for learning the address from the destination (or its representative) each time an association is desired. Only after the source has learned how to address the destination can an association be said to have occurred. But this is not yet sufficient. If

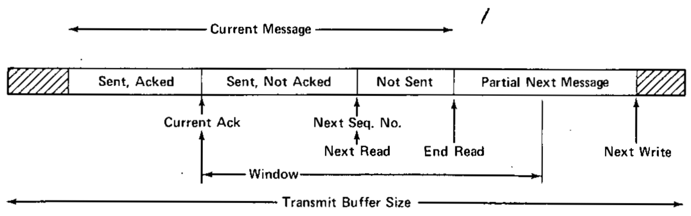


Fig. 12. Transmit buffer layout.

ordering of delivered messages is also desired, both TCP's must maintain sufficient information to allow proper sequencing. When this information is also present at both ends, then an association is said to have occurred.

Note that we have not said anything about a path, nor anything which implies that either end be aware of the condition of the other. Only when both partners are prepared to communicate with each other has an association occurred, and it is possible that neither partner may be able to verify that an association exists until some data flows between them.

CONNECTION-FREE PROTOCOLS WITH ASSOCIATIONS

In the ARPANET, the interface message processors (IMP's) do not have to open and close connections from source to destination. The reason for this is that connections are, in effect, always open, since the address of every source and destination is never[5] reassigned. When the name and the place are static and unchanging, it is only necessary to label a packet with source and destination to transmit it through the network. In our parlance, every source and destination forms an association.

In the case of processes, however, we find that port addresses are continually being used and reused. Some ever-present processes could be assigned fixed addresses which do not change (e.g., the logger process). If we supposed, however, that every TCP had an infinite supply of port addresses so that no old address would ever be reused, then any dynamically created port would be assigned the next unused address. In such an environment, there could never be any confusion by source and destination TCP as to the intended recipient or implied source of each message, and all ports would be associates.

Unfortunately, TCP's (or more properly, operating systems) tend not to have an infinite supply of internal port addresses. These internal addresses are reassigned after the demise of each port. Walden [12] suggests that a set of unique uniform external port addresses could be supplied by a central registry. A newly created port could apply to the central registry for an address which the central registry would guarantee to be unused by any HOST system in the network. Each TCP could maintain tables matching external names with internal ones, and use the external ones for communication with other processes. This idea violates the premise that interprocess communication should not require centralized control. One would have to extend the central registry service to include all HOST's in all the interconnected networks to apply this idea to our situation, and we therefore do not attempt to adopt it.

Let us consider the situation from the standpoint of the TCP. In order to send or receive data for a given port, the TCP needs to set up a TCB and RCB and initialize the window size and left window edge for both. On the receive side, this task might even be delayed until the first packet destined for a given port arrives. By convention, the first packet should be marked so that the receiver will synchronize to the received sequence number.

On the send side, the first request to transmit could cause a TCB to be set up with some initial sequence number (say, zero) and an assumed window size. The receiving TCP can reject the packet if it wishes and notify the sending TCP of the correct window size via the acknowledgment mechanism, but only if either

1) we insist that the first packet be a complete segment
2) an acknowledgment can be sent for the first packet (even if not a segment, as long as the acknowledgment specifies the next sequence number such that the source also understands that no bytes have been accepted).

It is apparent, therefore, that the synchronizing of window size and left window edge can be accomplished without what would ordinarily be called a connection setup.

The first packet referencing a newly created RCB sent from one associate to another can be marked with a bit which requests that the receiver synchronize his left window edge with the sequence number of the arriving packet (see SYN bit in Fig. 8). The TCP can examine the source and destination port addresses in the packet and in the RCB to decide whether to accept or ignore the request.

Provision should be made for a destination process to specify that it is willing to LISTEN to a specific port or "any" port. This last idea permits processes such as the logger process to accept data arriving from unspecified sources. This is purely a HOST matter, however.

The initial packet may contain data which can be stored or discarded by the destination, depending on the availability of destination buffer space at the time. In the other direction, acknowledgment is returned for receipt of data which also specifies the receiver's window size.

If the receiving TCP should want to reject the synchronization request, it merely transmits an acknowledgment carrying a release (REL) bit (see Fig. 8) indicating that the destination port address is unknown or inaccessible. The sending HOST waits for the acknowledgment (after accepting or rejecting the synchronization request) before sending the next message or segment. This rejection is quite different from a negative data acknowledgment. We do not have explicit negative acknowledgments. If no acknowledgment is returned, the sending HOST may

[5] Unless the IMP is physically moved to another site, or the HOST is connected to a different IMP.

retransmit without introducing confusion if, for example, the left window edge is not changed on the retransmission.

Because messages may be broken up into many packets for transmission or during transmission, it will be necessary to ignore the REL flag except in the case that the EM flag is also set. This could be accomplished either by the TCP or by the GATEWAY which could reset the flag on all but the packet containing the set EM flag (see Fig. 9).

At the end of an association, the TCP sends a packet with ES, EM, and REL flags set. The packet sequence number scheme will alert the receiving TCP if there are still outstanding packets in transit which have not yet arrived, so a premature dissociation cannot occur.

To assure that both TCP's are aware that the association has ended, we insist that the receiving TCP respond to the REL by sending a REL acknowledgment of its own.

Suppose now that a process sends a single message to an associate including an REL along with the data. Assuming an RCB has been prepared for the receiving TCP to accept the data, the TCP will accumulate the incoming packets until the one marked ES, EM, REL arrives, at which point a REL is returned to the sender. The association is thereby terminated and the appropriate TCB and RCB are destroyed. If the first packet of a message contains a SYN request bit and the last packet contains ES, EM, and REL bits, then data will flow "one message at a time." This mode is very similar to the scheme described by Walden [12], since each succeeding message can only be accepted at the receiver after a new LISTEN (like Walden's RECEIVE) command is issued by the receiving process to its serving TCP. Note that only if the acknowledgment is received by the sender can the association be terminated properly. It has been pointed out[6] that the receiver may erroneously accept duplicate transmissions if the sender does not receive the acknowledgment. This may happen if the sender transmits a duplicate message with the SYN and REL bits set and the destination has already destroyed any record of the previous transmission. One way of preventing this problem is to destroy the record of the association at the destination only after some known and suitably chosen timeout. However, this implies that a new association with the same source and destination port identifiers could not be established until this timeout had expired. This problem can occur even with sequences of messages whose SYN and REL bits are separated into different internetwork packets. We recognize that this problem must be solved, but do not go into further detail here.

Alternatively, both processes can send one message, causing the respective TCP's to allocate RCB/TCB pairs at both ends which rendezvous with the exchanged data and then disappear. If the overhead of creating and destroying RCB's and TCB's is small, such a protocol might be adequate for most low-bandwidth uses. This idea might also form the basis for a relatively secure transmission system. If the communicating processes agree to change their external port addresses in some way known only to each other (i.e., pseudorandom), then each message will appear to the outside world as if it is part of a different association message stream. Even if the data is intercepted by a third party, he will have no way of knowing that the data should in fact be considered part of a sequence of messages.

We have described the way in which processes develop associations with each other, thereby becoming associates for possible exchange of data. These associations need not involve the transmission of data prior to their formation and indeed two associates need not be able to determine that they are associates until they attempt to communicate.

[6] S. Crocker of ARPA/IPT.

CONCLUSIONS

We have discussed some fundamental issues related to the interconnection of packet switching networks. In particular, we have described a simple but very powerful and flexible protocol which provides for variation in individual network packet sizes, transmission failures, sequencing, flow control, and the creation and destruction of process-to-process associations. We have considered some of the implementation issues that arise and found that the proposed protocol is implementable by HOST's of widely varying capacity.

The next important step is to produce a detailed specification of the protocol so that some initial experiments with it can be performed. These experiments are needed to determine some of the operational parameters (e.g., how often and how far out of order do packets actually arrive; what sort of delay is there between segment acknowledgments; what should be retransmission timeouts be?) of the proposed protocol.

ACKNOWLEDGMENT

The authors wish to thank a number of colleagues for helpful comments during early discussions of international network protocols, especially R. Metcalfe, R. Scantlebury, D. Walden, and H. Zimmerman; D. Davies and L. Pouzin who constructively commented on the fragmentation and accounting issues; and S. Crocker who commented on the creation and destruction of associations.

REFERENCES

[1] L. Roberts and B. Wessler, "Computer network development to achieve resource sharing," in *1970 Spring Joint Computer Conf., AFIPS Conf. Proc.*, vol. 36. Montvale, N. J.: AFIPS Press, 1970, pp. 543–549.
[2] L. Pouzin, "Presentation and major design aspects of the CYCLADES computer network," in *Proc. 3rd Data Communications Symp.*, 1973.
[3] F. R. E. Dell, "Features of a proposed synchronous data network," in *Proc. 2nd Symp. Problems in the Optimization of Data Communications Systems*, 1971, pp. 50–57.

[4] R. A. Scantlebury and P. T. Wilkinson, "The design of a switching system to allow remote access to computer services by other computers and terminal devices," in *Proc. 2nd Symp. Problems in the Optimization of Data Communications Systems*, 1971, pp. 160–167.

[5] D. L. A. Barber, "The European computer network project," in *Computer Communications: Impacts and Implications*, S. Winkler, Ed. Washington, D. C., 1972, pp. 192–200.

[6] R. Despres, "A packet switching network with graceful saturated operation," in *Computer Communications: Impacts and Implications*, S. Winkler, Ed. Washington, D. C., 1972, pp. 345–351.

[7] R. E. Kahn and W. R. Crowther, "Flow control in a resource-sharing computer network," *IEEE Trans. Commun.*, vol. COM-20, pp. 539–546, June 1972.

[8] J. F. Chambon, M. Elie, J. Le Bihan, G. LeLann, and H. Zimmerman, "Functional specification of transmission station in the CYCLADES network. ST-ST protocol" (in French), I.R.I.A. Tech. Rep. SCH502.3, May 1973.

[9] S. Carr, S. Crocker, and V. Cerf, "HOST-HOST Communication Protocol In the ARPA Network," in *Spring Joint Computer Conf., AFIPS Conf. Proc.*, vol. 36. Montvale, N. J.: AFIPS Press, 1970, pp. 589–597.

[10] A. McKenzie, "HOST/HOST protocol for the ARPA network," in *Current Network Protocols*, Network Information Cen., Menlo Park, Calif., NIC 8246, Jan. 1972.

[11] L. Pouzin, "Address format in Mitranet," NIC 14497, INWG 20, Jan. 1973.

[12] D. Walden, "A system for interprocess communication in a resource sharing computer network," *Commun. Ass. Comput. Mach.*, vol. 15, pp. 221–230, Apr. 1972.

[13] B. Lampson, "A scheduling philosophy for multiprocessing systems," *Commun. Ass. Comput. Mach.*, vol. 11, pp. 347–360, May 1968.

[14] F. E. Heart, R. E. Kahn, S. Ornstein, W. Crowther, and D. Walden, "The interface message processor for the ARPA computer network," in *Proc. Spring Joint Computer Conf., AFIPS Conf. Proc.*, vol. 36. Montvale, N. J.: AFIPS Press, 1970, pp. 551–567.

[15] N. G. Anslow and J. Hanscoff, "Implementation of international data exchange networks," in *Computer Communications: Impacts and Implications*, S. Winkler, Ed. Washington, D. C., 1972, pp. 181–184

[16] A. McKenzie, "HOST/HOST protocol design considerations," INWG Note 16, NIC 13879, Jan. 1973.

[17] R. E. Kahn, "Resource-sharing computer communication networks," *Proc. IEEE*, vol. 60, pp. 1397–1407, Nov. 1972.

[18] Bolt, Beranek, and Newman, "Specification for the interconnection of a host and an IMP," Bolt Beranek and Newman, Inc., Cambridge, Mass., BBN Rep. 1822 (revised), Apr. 1973.

Vinton G. Cerf was born in New Haver Conn., in 1943. He did undergraduate wor in mathematics at Stanford Universit Stanford, Calif., and received the Ph.D. de gree in computer science from the Universit of California at Los Angeles, Los Angele Calif., in 1972.

He was with IBM in Los Angeles fro 1965 through 1967 and consulted and/o worked part time at UCLA from 1967 throug 1972. Currently he is Assistant Professor Computer Science and Electrical Engineering at Stanford Universit and consultant to Cabledata Associates. Most of his current researc is supported by the Defense Advanced Research Projects Agency an by the National Science Foundation on the technology and economi of computer networking. He is Chairman of IFIP TC6.1, an inte national network working group which is studying the proble of packet network interconnection.

★

Robert E. Kahn (M'65) was born in Brookly N. Y., on December 23, 1938. He received th B.E.E. degree from the City College of Ne York, New York, in 1960, and the M. and Ph.D. degrees from Princeton Universit Princeton, N. J., in 1962 and 1964, re spectively.

From 1960 to 1962 he was a Member of th Technical Staff of Bell Telephone Labora tories, Murray Hill, N. J., engaged in traff and communication studies. From 1964 t 1966 he was a Ford Postdoctoral Fellow and an Assistant Professo of Electrical Engineering at the Massachusetts Institute of Tec nology, Cambridge, where he worked on communications and i formation theory. From 1966 to 1972 he was a Senior Scientist Bolt Beranek and Newman, Inc., Cambridge, Mass., where h worked on computer communications network design and techniqu for distributed computation. Since 1972 he has been with the A vanced Research Projects Agency, Department of Defens Arlington, Va.

Dr. Kahn is a member of Tau Beta Pi, Sigma Xi, Eta Kappa N the Institute of Mathematical Statistics, and the Mathemati Association of America. He was selected to serve as a Nation Lecturer for the Association for Computing Machinery in 1972.

Why Cell Phones Will Dominate the Future Internet

S. Keshav
Canada Research Chair in Tetherless Computing
School of Computer Science, University of Waterloo
200 University Ave. W, Waterloo, ON, Canada N2L 3G1
keshav@cs.uwaterloo.ca

Categories and Subject Descriptors
C,2,1 [**Network Architecture and Design**]: Network Communications

General Terms
Management, Performance, Design, Economics, Reliability

Keywords
Internet architecture, Cellular communications

1. INTRODUCTION
Quick! What does the Internet look like? Chances are pretty good that you're thinking of desktop computers on a LAN connected to servers, middleboxes, and other LAN-based desktops by routers and local and long-distance point-to-point links. Implicit in this model is the assumption that all users access the Internet from a desktop computer. Although substantially true five years ago, I will argue that the picture is already wrong today and that in the near future most people will use cell phones rather than desktops to access the Internet. A good model for the future Internet would therefore be a very large number of cell-phone-like, mobile, wireless, lightweight, end-systems, connected using CDMA and GPRS (and potentially, IEEE802.11 as well), to well-managed cell-phone provider networks, that provide access to a highly-connected bandwidth-rich wired core and associated centralized servers. Desktops will not disappear, of course, but they will play an increasingly smaller role in the typical way an Internet user accesses the network; a large fraction of future Internet users may never use what we would think of as a desktop today.

As a surprising consequence, the holy grails of cost-effective, always-available network access, multimedia networking, and end-to-end quality of service may finally be achieved. A cell-phone dominated Internet may thus resolve the decades-old tension between the telephony and *laissez faire* packet-networking views of the world.

Section 2 outlines cell phone technology and demonstrates why today's mobiles are as much IP end points as desktop machines. In terms of Internet access, for simplicity, I'll only consider three alternatives to cell phones: wired desktops, wired or wireless laptops, and wireless Personal Digital Assistants (PDAs), where the wireless laptops and PDAs use 802.11 (WiFi) access points. Section 3 describes non-technical advantages of cell phones over these alternatives, and Section 4 presents their technical advantages. Section 5 discusses cell phone evolution trends, and presents some wild speculations on what this means for future Internet architecture.

2. OVERVIEW
A cell phone is essentially a battery-powered microprocessor with one or more wireless transmitters and receivers optimized for voice I/O. Even a bare-bones model provides a keyboard, an LCD screen, and a general-purpose computing platform, typically supporting Java2 Mobile Edition (J2ME) or .NET Compact APIs. More sophisticated models provide a camera, 1MB-5GB of local storage, a full-color screen, multiple wireless interfaces, and even a QWERTY keypad.

Importantly, a cell phone cannot be used without a globally unique, per-user, hard-to-forge identifier, called the International Mobile Subscriber Identifier or IMSI[1]. IMSIs are allocated by cell phone providers and allow them to track and bill for usage. A cell phone provider maintains a comprehensive database, called the Home Location Register (HLR), that keeps track of the current location of each IMSI, its usage, and associated subscriber information, such as a credit card number, or prepaid usage authorization. HLRs make it possible for cell phone providers to do very fine-grained billing.

Nearly all cell phones today provide voice and data I/O over either CDMA or GPRS networks[2]. In both networks, data access is over a channelized medium, where separate wireless frequency channels (or, equivalently, timeslots) are dedicated to data communication and signaling [BVE 99]. For example, in GPRS, cell phones contend for access to the data channel using Slotted-ALOHA on one of the control channels (called the PRACH channel). In response, the base station uses the packet grant channel (PAGCH) to explicitly grant it one or more time slots on the data channel (PDTCH). The cell phone uses these slots to send an IP packet, encapsulated in a convergence layer protocol (SNDCP), to the base station, which forwards it to a local packet router (the SGSN), which tunnels it to an IP *gateway* (the GGSN), where it enters the Internet. Symmetrically, data meant for a cell phone is routed through the Internet to the gateway (GGSN), which tunnels it to the SGSN, and thence to its base station. The base station uses a data channel to deliver the packet to the cell phone.

A cell phone that wants to send and receive IP packets starts by requesting a *packet data protocol context* from the cell phone provider. This context assigns it a packet data protocol (IPv4 or v6), a corresponding IP address, a quality of service specification, and, optionally, a DNS name. This process allows the cell phone

[1] Strictly speaking, a cell phone used for an emergency call (such as to 911 in North America) does not need an IMSI.

[2] This paper focuses on packetized data I/O, which includes Voice over IP, and ignores circuit-switched voice.

network (specifically, the GGSN) to associate the cell phone's unique ID (i.e. IMSI) with its IP address. Thereafter, standard cell phone locationing ensures that any packets sent to the cell phone's IP address can be routed to it no matter where it roams. Note that after receiving a packet data context, to all intents and purposes, the cell phone is on the Internet and can exchange IP packets with any other Internet host, making it a *bona fide* Internet host. In the next generation infrastructure, called IP Multimedia Subsystem (IMS), all mobiles will receive an IPv6 address, and the entire backbone will be IPv6-enabled.

In addition to basic data transport, cell phone networks provide two types of messaging. The Short Message Service (SMS) allows up to 160-character messages to be sent to a cell phone with little delay. Over 50 billion SMS messages were sent in 2004. Next-generation Multimedia Message Service (MMS) messages are compatible with SMTP and are unlimited length, thereby transforming a cell phone into standard Internet email endpoint.

3. NON-TECHNICAL ADVANTAGES

Having briefly surveyed cell phone technology, let us consider the non-technical advantages of cell phones over desktops, laptops, and PDAs.

3.1 Sheer numbers

By the end of 2003, there were 1.4 billion cell phones, serving about 25% of the world's population [ITU 05]. In comparison, there were only 607 million PCs (which includes both desktops and laptops), and negligibly small number of PDAs. In other words, in 2003 about a billion people had cell phones but not desktops/laptops or PDAs. Cell phones continue to maintain this lead because of a rapid rise in subscriber numbers in China, India, and Russia. For instance, in 2004, China reported 310 million users, about 25% of its total population, and India saw an increase of 11 million, or 25%, and reached a total of 44.5 million subscribers. In Russia, mobile phone subscriber numbers jumped 65% from 36.5 million in 2003 to 60 million by September 2004 [IT Facts 05].

Thus, simply in terms of numbers, cell phones are *already* the dominant platform for Internet access. Unfortunately, very few people use cell phones for data access today. However, this is likely to change. Here's why:

The large and a rapidly growing market makes cell phones attractive to handset vendors, operating system providers, software houses, and service providers, leading to fast-paced innovation. This is already apparent by considering the range of handset choices today – ranging from the Samsung SCH-V770 that has a 7-Megapixel camera, to Sanyo's HDR-B5GM that includes a 1-inch 5GB drive. Given that voice revenue has wafer-thin margins, there is a huge financial incentive to roll out innovative data services, such as those pioneered by NTT DoCoMo, rapidly [Imode 05]. For example, DoCoMo subscribers use data services at up to 384 Kbps to access video clips, upload camera images, and get street maps, stock quotes, restaurant menus, weather, and 'yellow pages' information. This sort of innovation will continue to transform cell phones from voice-only devices to integrated voice-data devices, to eventually becoming data-dominant devices.

3.2 Cost

A second effect of a large user population is that costs shrink dramatically, both for handsets and for service. Handsets are already manufactured in the hundreds of millions every year, which makes it possible to dedicate expensive chip fabs and ASIC developers to the task. These up-front capital investments greatly reduce the marginal cost per handset. Similarly, large volumes allow service providers to make healthy profits with small markups, reducing the price of service. Lower handset and service prices in turn increases the total addressable market, leading to positive feedback.

Similar positive feedback effects have led to decreasing prices for desktops and laptops, but PDA volumes are simply too small to ride the technology cost curve, and consequently several manufacturers (such as Sony and Sharp) have withdrawn from the market. Also, in contrast to wireless access prices, a lack of competitive pressure for wired local access has actually increased the service fees for monthly access in the United States! So, in terms of handset and service prices, cell phones have taken the lead, and are likely to maintain it for the near future.

3.3 Marketing model

Cell phone providers usually give away or highly subsidize handsets in an effort to gain market share. The handset cost is recouped over a two- or three-year period as part of a monthly service fee. Essentially, the provider acts like a bank to finance the cost of the handset. This reduction in the cost of the handset makes it very affordable. In contrast, few desktop or PDA vendors offer comparable terms, especially to consumers.

3.4 Well-established providers

Unlike WiFi hotspot providers and most ISPs, cell phone providers are well-capitalized, well-established, and have a large cash flow because they own significant shares of their home markets. Consequently, they are able to adequately provision their networks and, more importantly, enter into long-term settlement contracts with each other. This allows subscribers to seamlessly roam between coverage areas, receiving a single monthly bill. In contrast, one cannot imagine obtaining seamless roaming Internet access from either wired or WiFi service providers today: there are too many ISPs and they rarely trust each other! The ability to roam will greatly reinforce end user preferences to access the Internet using cell phone providers.

3.5 Form factor

Ideally, Internet access, like telephone access, should be high quality, cheap, and available everywhere. Cell phones have overtaken fixed-line phones for voice transport because people are willing to compromise on quality and cost to gain mobile access. One cannot carry the analogy too far: cell phones and fixed-line phones both provide roughly equivalent voice quality, but mobile cell phones will never have the screen size and ease of use of a desktop. So, there will always be a market for fixed, wired, powered desktops. Nevertheless, (a) some users may not need a desktop as we know it today and (b) cell phones are likely to supplant laptops and PDAs as mobile devices.

To begin with, as Moore's law allows increasingly more processing power to be crammed into a chip, cell phone processors will eventually be powerful enough to run common office productivity applications. Imagine that such a cell phone also comes with a dock that takes power in, and provides keyboard, video, and mouse out. This 'desktop' is well within reach by 2010, and may be adequate for most users who do not want to own another computing device. This 'docked' cell phone may be connected to the network on a fixed line, or may provide CDMA/GPRS and WiFi access.

Second, as a mobile computer, laptops are too heavy, too awkward to carry, and do not permit opportunistic data access, such as for reading email while waiting for an elevator. In contrast, cell phones and PDAs have the right form factor. However, PDAs that use WiFi for Internet access cannot benefit from the cost, subsidy and market size benefits that are available to cell phones. For these reasons, it is clear that, in the long term, cell phones, especially multi-NIC cell phones, will replace PDAs; and perhaps 'docked' cell phones may replace some laptops and desktops.

To sum up, we see that cell phones already numerically dominate the number of Internet end points, though data services accessed through cell phones today are relatively scarce. However, the cell phone market is likely to grow due to shrinking hardware and service costs and subsidized handsets. Moreover, with voice margins shrinking, service providers are sure to leverage their existing relationships and huge cash flows to fund innovation and provide data services and seamless worldwide roaming. These factors will make data services on cell phones rapidly gain popularity, making cell phones the dominant Internet access technology. 'Docked' cell phones of the future may also supplant laptops and desktops.

4. TECHNICAL ADVANTAGES

As good as this sounds, this still leaves a major question unanswered. Even if cell phones dominate future Internet access, are they the best technical solution to the problem? Are desktops, WiFi-enabled PDAs, or laptops better Internet access devices that are losing out to a technically inferior solution? In this section, I argue that cell phones not only will win out, but actually have several significant technical advantages over these competing solutions.

I will first outline technical advantages of cell phone handsets and wireless access over WiFi-based laptops and PDAs (Sections 4.1 and 4.2). I will then discuss why the cell phone-provider managed portion of the Internet is better than the 'general' Internet (Sections 4.3-4.7).

4.1 Power management

Power management is critical for battery-powered mobile devices. Laptop and PDA vendors tend to sacrifice power for backlit screens and processing speed, leading to short device lifetimes of two to eight hours. In contrast, cell phone vendors have always paid fanatical attention to power management, leading to much longer device lifetimes. Talk times are typically six to eight hours, and standby times are measured in days. For instance, unlike cell phones, laptops and PDAs do not support a standby mode where they can power off most services while still being available for incoming data. Therefore, from this perspective, cell phones are definitely a better technical solution than laptops and PDAs.

4.2 Channelization

IEEE 802.11 is not channelized, so control packets, such as RTS, CTS, and ACK, use the same channel as data packets. This leads to complex arbitration schemes and potentially unfair bandwidth allocation due to hidden and exposed terminals [BDSZ 94]. Cell phones use channelized media, which intrinsically share the wireless medium better and are immune to a variety of hidden terminal problems (See [BTV 04] and the references therein). Moreover, unlike a WiFi-based PDA, a cell phone cannot be blocked from accessing the channel because the data channel is hogged by another cell phone. For these reasons, it appears that a channelized cell phone may better use wireless spectrum than a WiFi-based laptop or PDA. Incidentally, cell phone spectrum is licensed, so it is also immune to interference from cordless phones and microwave ovens that occupy the unlicensed ISM bands.

4.3 Identity and location management

A major problem with the Internet today is that IP addresses are topologically significant. In contrast, a cell phone's identifier (IMSI) identifies its *user*, has no topological significance, and is bound dynamically to its location using a Home Location Register (HLR). As a cell phone moves, its location is always (more or less) known to the HLR. The 3G Partnership Project [AJM 04] defines how this is coordinated with Mobile IP. Essentially, packets destined to a phone are sent to its home network, and then forwarded using Mobile IP to the cell phone's current location, which is obtained from the HLR. No such locationing information is available for standard Internet endpoints: a PDA or laptop using a WiFi hotspot with a NAT'ted DHCP private address is simply unlocatable!

4.4 Quality of service

Packets to and from cell phones are transported (at least partially) over a provider's private IP network. Note that every cell phone is uniquely identified using hardware identifiers and has a billing relationship with the service provider. This makes it both technically and *economically* feasible to provide them quality of service guarantees, especially for multimedia applications and VoIP. Imagine that a cell phone user wants to view a video stream from a video server on its provider's private Internet. Because the source and destination endpoints are known, the path can be pinned down using MPLS, and quality of service guarantees can be provided using either IntServ or DiffServ. All of this makes economic sense because the provider can charge the cell phone user per-byte or per-video using an existing billing relationship. None of these pre-conditions for quality of service provision exist in the general Internet.

Note that GPRS-based cell phone providers are tied together using the private GRX network [GRX 04]. In principle, this allows global cell phone-to-cell phone provisioning of quality of service parameters, with built-in support for billing and settlement. This is an essential pre-condition to providing end-to-end quality of service, which is economically infeasible in the general Internet. and, by extension to laptops, PDAs, and even desktops on it.

4.5 Over-the-air software upgrades

Keeping application versions on endpoints up-to-date and consistent is a significant headache for every enterprise today. Cell phones, by design, do not suffer from this problem. When software on a cell phone has to be updated, it is automatically downloaded to the phone by the cell phone provider, using software such as that provided by Bitfone [Bitfone 05].

4.6 IPv6 support

The IETF has been struggling for about 15 years to migrate the general Internet from IPv4 to IPv6, and it might never happen. In contrast, the next-generation cell-phone provider IP network is specified to be IPv6. This is because cell phones are managed systems that are dynamically allocated IP addresses and whose software can be dynamically updated by a cell phone provider. Cutting over to IPv6 requires installation of the appropriate protocol stack on cell phones, and switching over to a parallel internal infrastructure. Neither the capability to download software to an end system nor the control over the network infrastructure to force a cut over to IPv6 exist in the general Internet.

4.7 Ease of maintenance

Both the wired and the wireless components of a cell phone provider's network are *managed*. This means that the provider is responsible for network provisioning, monitoring, and management. This delegation of responsibilities from the end system to the network makes the overall system stable: links can be provisioned on the basis of a measured traffic matrix, consistent routing tables can be centrally computed and installed on routers, routes can be pinned using MPLS, and link quality can be uniformly measured by a network-wide operations center, much like the telephone network. Because there are only a few hundred cell phone providers worldwide, in contrast to the tens of thousands of ISPs, worldwide coordination and management is feasible. Similarly, providers can roll out system-wide changes or improvements in infrastructure without having to impact the service seen by the end points.

5. IMPLICATIONS

With rapidly increasing coverage, data-enabled cell phones will soon deliver the long-held vision of 'anytime anywhere information access.' Moreover, based on their technical and non-technical advantages, I believe that cell phone handsets will quickly displace PDAs. Over the longer term, dockable cell phones, as described in Section 3.5, may even displace laptops and desktops.

Much like an iPod, such dockable devices with large local storage will allow users to carry all their data with them all the time. In other words, these devices will not only provide data access, but also *data* and *compute* mobility. This will make it desirable to allow updates of the local data store, either from CDMA/GPRS networks, or opportunistically from WiFi/Bluetooth networks, so that periods of disconnection can be hidden from the user.

User demand for higher bandwidths and lower operational costs will inevitably lead to the proliferation of WiFi- and WiMax-enabled cell phones that can switch to WiFi or WiMax when available. To provide good voice and data quality, especially for VoIP, this will require cell phone providers to either build out or partner with WiFi/WiMax providers, and provide the same degree of management on these networks and their backhaul as they provide on their own backbones.

Following this train of logic, it seems clear that the proliferation of cell phones, their use for data access, and the concomitant growth of cell phone-based Internet service providers will lead to an increasingly larger portion of the Internet being managed and provisioned by cell phone providers. As the fraction of users accessing the Internet from cell phones grows, there will be a strong financial incentive for Internet application service providers like Yahoo and Google to establish a presence on this provisioned network. It may well be that over time, most, if not all Internet service providers (VoIP providers and web site hosters included) have links both to the provisioned and the 'best effort' Internet, and eventually the provisioned Internet may make the best-effort Internet vestigial. If this happens, the provisioned Internet would integrate the best ideas of the last fifty years of telecommunications: temporal statistical multiplexing gains through packet switching, traffic management using provisioned MPLS paths, and end-to-end quality of service guarantees using effective scheduling disciplines. This very welcome state of affairs is achievable in the next ten years, and may perhaps be inevitable, given that cell phone users will dominate the future Internet.

6. ACKNOWLEDGMENTS

The ideas in this work owe much to ongoing discussions with my students, Aaditeshwar Seth, Nabeel Ahmed, and Majid Ghaderi at the University of Waterloo. This research was undertaken, in part, thanks to funding from the Canada Research Chairs Program. Thanks also to Intel Corp., Sprint ATL, Nortel Networks Institute, and Sun Microsystems for the generous financial support.

7. REFERENCES

[AJM 04] I. Akyildiz, X. Jiang, and S. Mohanty, "A Survey of Mobility Management in Next-Generation All-IP-Based Wireless Systems," *IEEE Wireless Communications,* Vol. 11, No. 4, Aug. 2004, pp16 - 28.

[BDSZ 94] V. Bharghavan, A. Demers, S. Shenker , and L. Zhang. "MACAW: A Media Access Protocol for Wireless LANs," *Proc. ACM SIGCOMM*, September 1994, pp. 212-225.

[Bitfone 05] http://www.bitfone.com/usa/product_mprove_architecture.html

[BTV 04] A. Baiocchi, A. Todini, A. Valletta, "Why a Multichannel Protocol can Boost IEEE 802.11 Performance," *Proc. MSWiM '04,* October 2004.

[BVE 99] C. Bettstetter, H-J. Vogel, and J. Eberspeher, "GSM Phase 2+ General Packet Radio Service GPRS: Architecture, Protocols, and Air Interface," *IEEE Communication Survey*, 1999.

[GRX 04] GSM Association, "Inter-PLMN Backbone Guidelines." Official Publication IR.34, August 2004.

[Imode 05] http://www.nttdocomo.com/corebiz/services/imode

[IT Facts 05] http://www.itfacts.biz/index.php?id=C0_4_1

[ITU 05] http://www.itu.int/ITU-D/ict/statistics/at_glance/

Future and Emerging Technologies: A Vision for Tomorrow of EU IST Research

Fabrizio Sestini
European Commission DG Information Society
Future & Emerging Technologies
BU33, 3/22 B-1049 Brussels, Belgium
Tel. +32-2-2995260
fabrizio.sestini @cec.eu.int

Thierry Van der Pyl
European Commission DG Information Society
Head of Unit, Future & Emerging Technologies
BU33, 3/35 B-1049 Brussels, Belgium
Tel. +32-2-2968105
thierry.vanderpyl @cec.eu.int

ABSTRACT[1]

FET (Future and Emerging Technologies) is a part of the EU-funded research activities which addresses emerging ICT-related research domains, explores options in ICT technology roadmaps where road blocks are anticipated but where "no known solutions" are available and evaluates these options. FET activities have proved very efficient in feeding the core of the IST programme with emerging concepts or radically new technological paradigms, often arising at the crossroads of existing disciplines. In the coming years these activities, which are expected to be complemented by an envisaged European Research Council, will continue nurturing and stimulating creativity and excellence in ICTs research also in combination with other relevant S&T fields.

1. INTRODUCTION

Latest scientific trends confirm that the explosive pace of technological development that we are experiencing in the last decade will not cease soon. It is fuelled by continuous advances in nanotechnology, in software and in communication technologies.

Exploring the new miniaturisation frontiers, harnessing increasing levels of complexity in the very small, nano-scale, or in the very large, planetary scale, building more intelligent systems, more personalised products and services, all these are just a few of the challenges that lie ahead of us.

Today we witness the increasing integration of ICT with other sciences and in particular with nano-, bio- and life sciences. These sciences are providing powerful insights that are at the basis of new approaches in ICT. For example, solutions for more robust, adaptable and highly complex systems are being inspired from biology and evolution theory. A better understanding of the biological and neural basis of cognition will enable ICT systems to learn, evolve, reason and adapt to their environments.

This cross fertilisation of ICT with many other relevant sciences will intensify as scientists from different disciplines learn each other's ways of thinking. Prepared by this conceptual convergence, a new era of technological convergence between the nano-, bio-, neuro- and cognitive sciences is announcing itself.

These are subjects for basic research, which is being funded by the European Commission as part of the Framework Research Programmes of collaborative research in Europe. The current 6th Framework Programme, covering the period 2002 to 2006, includes a priority called Information Society Technologies (IST), with a budget of more than 3.6 billion euro, which is meant to finance or co-finance Research projects in the domain of Information and Communication Technologies, projects carried out by multinational European consortia.

Nearly 10% of this public funding is allocated to "Future and Emerging Technologies" (FET), which is addressing the part of research that is of a longer-term nature. FET has a horizon well beyond any current Framework Programme, towards research that is visionary, risky and that has a potential for technological breakthroughs.

A two-tiered approach is followed, composed of an Open Scheme and various Proactive Initiatives (see Figure 1):

- the Open Scheme is open, at any time, to the broadest possible spectrum of ideas as they come directly 'from the roots'; as such, it is thematically unconstrained, ensuring seamless coverage of all information society technologies by keeping the door open to any new and challenging ideas, in a bottom-up fashion;
- Proactive initiatives, following the inverse approach, have the more strategic objective of focusing resources on a few visionary and challenging long-term goals that hold particular promise for the future, in a top-down fashion.

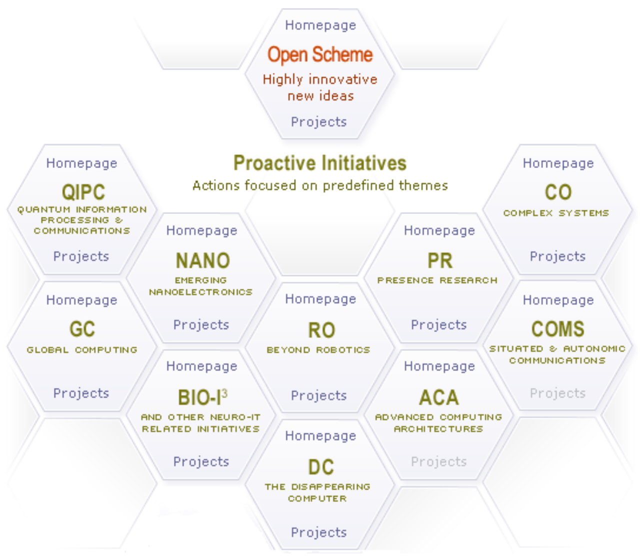

Homepage
Open Scheme
Highly innovative new ideas
Projects
Proactive Initiatives
Actions focused on predefined themes
Homepage QIPC QUANTUM INFORMATION PROCESSING & COMMUNICATIONS Projects
Homepage CO COMPLEX SYSTEMS Projects
Homepage NANO EMERGING NANOELECTRONICS Projects
Homepage PR PRESENCE RESEARCH Projects
Homepage GC GLOBAL COMPUTING Projects
Homepage RO BEYOND ROBOTICS Projects
Homepage COMS SITUATED & AUTONOMIC COMMUNICATIONS Projects
Homepage BIO-I3 AND OTHER NEURO-IT RELATED INITIATIVES Projects
Homepage ACA ADVANCED COMPUTING ARCHITECTURES Projects
Homepage DC THE DISAPPEARING COMPUTER Projects

Figure 1 - structure of FET activities
(see http://www.cordis.lu/ist/fet/areas.htm)

[1] This paper presents solely the opinions of the authors, which do not prejudice in any way those of the European Commission.

In summary, the “core values” of FET are:

- Promoting truly visionary high-risk, high-return research, designed to set future trends and create new research areas and agendas;
- Supporting high–quality, long-term research with sound objectives - in particular generic research that underpins a wide range of application areas, and provides a persistent and long-term commitment to emerging research communities;
- Focusing on new areas of research, not covered by other programmes, and where relatively limited FET funding can have an impact or make the difference;
- Recognising that scientific/technical failure is an inherent and inevitable feature in funding long-term and high-risk research - “ideas have high infant mortality rates” - and realising that failure can still generate a considerable body of knowledge and understanding;
- Interdisciplinarity is a fundamental characteristic of FET actions: while accepting that the starting point for work is to obtain an answer to a major scientific or technological challenge, it is realised that such answers can often only be found through an interdisciplinary research effort;
- Retaining lighter and more flexible procedures which are more suited to basic research - with the perspective to migrate such procedures to the main body of the IST programme.

In addition to this, it is worth reminding that FET, such as the rest of the IST programme, is open to participants not only from EU Member States, In particular, a number of “Associated States” can fully participate and receive funding, whereas participation from other Countries is always possible and participants from Countries targeted by specific international cooperation activities are also entitled to receive EU funding. For further details on this:

http://www.cordis.lu/fp6/stepbystep/who.htm ,
ftp://ftp.cordis.lu/pub/fp6/docs/tableau_260804_en.pdf ,
http://www.cordis.lu/fp6/inco_policies.htm.

The rest of the paper is structured as follows. Section 2 describes in more detail what FET Open is about and how it is implemented, whereas Section 3 is devoted to the Proactive initiatives. Finally Section 4 presents a perspective of how long-term and basic research in ICT will be addressed at European level in the future.

2. WHAT IS FET OPEN?

FET is open to any idea related to information society technologies. It does not only include the development of new technologies, but also encompasses new ways of doing things as well as creating new roles for technology. The philosophy is to explore new ideas - even if these ideas are only based on a dream, or a hunch, with the promise of really leading to something in the future. In this context there is no distinction of how far or how close to the market an idea might be - the important issue is the potential that it has for leading to a breakthrough.

It is also true that one idea leads to another and that progress sometimes comes from the accumulation of many small innovations (for example, innovative super-efficient algorithms). Many ideas may thus have matured past the 'wild phase' and been tested and proved valid to some degree, but still need persistent and long term work in order to take them to levels acceptable for industrial or commercial take-up.

2.1 Types of Projects in FET Open

The activities in FET Open are mainly funded though “Specific Targeted Research Projects” (STREP). STREPs are multipartner projects supporting research, technological development and demonstration or innovation activities.

Their budget may range from hundreds of thousands of Euros to a few millions of Euros and is paid as a grant (typically 50%) to the budget of the project. The typical duration is 24 - 36 months and there must be a minimum of three participants from three different Member States or Associated States.

FET-Open also supports the shaping, consolidation, or emergence of research communities and the coordination of national research programmes or activities in any IST-relevant area of advanced and longer term research.

Such activities are implemented through “Coordination Actions”, supporting the coordination and networking activities aiming at improving community building and integration, and “Specific Support Measures”, used to implement activities to support and prepare the FET policies and actions, like studies, impact measure reports, roadmapping of future research areas, conferences, workshops and expert meetings for defining FET future activities, etc.

2.2 Submission of proposals

Proposals in the FET Open scheme are submitted in a two step procedure: first, a short proposal (5 pages) is submitted and evaluated, normally within 2 months from the reception; if this short proposal is successful, proposers are invited to submit a full proposal. Short proposals can be submitted at any time, until 20 September 2005. The scheme will be open again in the following year, at a date to be established.

3. FET PROACTIVE INITIATIVES

Proactive initiatives aim at focusing resources on visionary and challenging long-term goals that are timely and have strong potential for future impact. The initiatives usually involve multidisciplinary work at the frontier of information technology and other disciplines such as physics, chemistry, life sciences, psychology, etc.

The long-term goals provide a common strategic perspective for all research work within the initiative and a focal point around which critical mass can be built and synergies developed.

The subjects of the proactive initiatives are identified by FET in a consultation process with the most active researchers and stakeholders in Europe, which normally involves organising brainstorming workshops and wider consultation meetings around promising topics. When there is a convergence of interests the topic becomes the subject of a call for proposal with a fixed deadline and a predefined budget.

3.1 Types of projects in Proactive Initiatives

In the 6th Framework Programme, each proactive initiative typically consists of one or more Integrated Projects (IPs) and, in some cases, a Network of Excellence (NoE).

Integrated Projects are multipartner projects supporting research, technological development and demonstration or innovation activities. They normally cover a broader range of activities than a STREP, addressing more ambitious objective-driven research dealing with different issues through a "programme approach". They must include at least three partners from different Member states (typically many more) and have a duration spanning from 3 to 5 years, with a budget that can exceed 10 million euro.

Networks of Excellence, in the context of a proactive initiative, have a specific role: they should bring together the broader community active in the research domain of the initiative in order to provide a framework of co-ordination for research and training activities at the European level, and allow the progressive and lasting integration of these activities around pre-specified themes. This may include the establishment of "distributed" centres of excellence, shared fabrication or experimental facilities, testbeds etc.

3.2 Proactive Initiative launched so far

Several proactive initiatives have been launched in 2003 and 2004 on subjects related to communications and networking. For further details on the projects in each area please refer to http://www.cordis.lu/ist/fet/areas.htm.

Situated and Autonomic Communications: new paradigms for distributed and self-organising communication and networking systems able to adapt to a changing context. The main objective is to define concept and technology for a self-organising communication network that can be situated in multiple and dynamic contexts. These can range from sensor networks to virtual networks of humans. Another objective is to study how strategic needs of social or commercial nature impact on future communication paradigms, and how networks and applications can support society and economy, enabling a service oriented, requirement- and trust- driven development of communication networks. The first projects selected in this initiative will start at the beginning of 2006.

Advanced Computing Architectures: delivering increased computing performance while lowering power consumption and maintaining the long-term stability of platforms for re-use of application software, key issues for emerging networked applications. The aim of this initiative is to substantially increase the performance of computing engines well beyond projected performance of Moore's law while reducing their power consumption. They should also provide leading compiler and operating system technology that will deliver high performance and efficient code optimisation, portable across a wide range of systems. Another aim is to constitute building blocks to be combined with each other and programmed easily and efficiently, even in heterogeneous processing platforms. The first projects selected in this initiative will start at the beginning of 2006.

Global Computing: foundational advances in the understanding, design, implementation and application of "global computers", i.e. programmable computational infrastructures and resources, distributed at world-wide scale and available globally. The Global Computing initiative reinforces and complements previous FET activities in the area. The key aim of this initiative is to define innovative theories, computational paradigms, linguistic mechanisms and implementation techniques. These shall be applied to the design, realisation and deployment of global computational environments and their application and management. The expected result in the long term is to achieve real, integrated global computing in a wide range of application scenarios. There are already several projects in this area which are coming to a conclusion, and a second group of projects in this area are starting in Autumn 2005.

FET also covers other areas which are less directly related to the domain of computing, communications and networking, but still in the field of Information Society Technologies, such as:

Beyond Robotics: development of physical mobile artefacts that could serve as companions to humans, function as bionic parts augmenting human capabilities, or act as autonomous micro-robot groups.

Stimulating Emergent Properties in Complex Systems: building a framework of mathematical and computational techniques for stimulating complex systems in science and engineering.

Bio-inspired Intelligent Information Systems: foundational research on information processing in biology in order to discover new paradigms for information technology.

Emerging Nanoelectronics: molecular scale approaches to information processing devices, circuits and architectures.

Presence: studying presence and interaction in mixed-reality environments, which can immerse in a consistent world of sensory-motor experiences generated from real as well as virtual stimuli.

Quantum Information Processing & Communications: to contribute to building systems that successfully implement quantum algorithms on small scale systems - including writing, processing and reading of qubits.

The Disappearing Computer: to see how information technology can be diffused into everyday objects and settings, and to see how this can lead to new ways of supporting and enhancing people's lives that go above and beyond what is possible with the computer today.

4. FUTURE PERSPECTIVES FOR BASIC RESEARCH IN EUROPE

In June 2004 the European Commission launched a consultation process on the preparation of the forthcoming 7th Framework Programme of community research (FP7, which probably will span over the next 7 years). Among other major objectives, the Commission suggests support of investigator-driven basic research through open competition between individual teams at European level, supported by a "European Research Council" (ERC). The ambition is to create a Europe-wide competitive funding mechanism for supporting frontier research projects proposed by individual teams in all scientific and technological fields, including ICT, within and across disciplines. The sole selection criterion will be scientific excellence evaluated by international peer-review.

This competition is expected to drive up the quality of science and of researchers across the board in Europe, resulting in more critical mass, less duplication and better results. By competing for research funding at the highest level of excellence in Europe researchers should be encouraged to raise their scientific standards and to pursue challenging research careers in an open, competitive manner. European funding should also help in raising the overall standards of scientific quality in national research funding systems throughout Europe.

However, the opening of a basic, non-targeted, research line in the future Framework Programme will not draw basic technological targeted research away from the IST programme, because it is important that basic and applied research be developed closely together, in one single programme. **The ERC mechanism should be seen as additional** to other FP activities and to national basic research activities and not aimed at replacing them.

In complement to the investigator- or curiosity-driven research of the ERC, FET will continue to support long-term trans-national collaborative basic technological research that is target-driven. In the draft Commission proposal for FP7 it is proposed that every thematic priority, including ICT/IST, has a part devoted to collaborative basic research. We thus expect to see collaborative, target-driven, basic research activities cohabitate with more industrially driven collaborative research everywhere in FP7.

It is through FET that Europe has been a pioneer for example in supporting areas such as nano-electronics, microsystems and interfaces using all our senses. Research in FET has started in these fields more than 15 years ago and they are now mainstream industrial activities. We are looking forward to witness the success of other areas which are currently subject of research in FET.

5. ACKNOWLEDGMENTS

The authors wish to thank all the colleagues who have contributed to this work and to the results of FET, and in particular Pekka Karp, Patrick Van Hove, Wide Hogenhout, Gustav Kalbe, Thomas Skordas, Ralph Dum, Anna Katrami-Bezirtzoglou, Antonella Karlsson, Loretta Anania, Walter Van de Velde, Julian Ellis.

6. REFERENCE

http://www.cordis.lu/ist/fet/home.html

802.11 "Decrypted"

Adrian Stephens
Intel Corporation
15 JJ Thompson Ave
Cambridge CB3 0FD, UK
+44 1223 763457

ABSTRACT

This short paper introduces wireless IEEE 802 standards and activities with a focus on explaining the purpose of the many 802.11 amendments.

Categories and Subject Descriptors

A.1 [**INTRODUCTORY AND SURVEY]**

C.2.5 [**Local and Wide-Area Networks**]

General Terms

Standardization

Keywords

TBD

1. INTRODUCTION

As wireless technology increasingly pervades our lives, the decisions made in wireless standards bodies such as the IEEE 802.11 have the potential to impact our lives. From the user viewpoint, emerging standards will support new applications, higher throughput and increasing mobility. From the implementer viewpoint, the increasing complexity must be hidden from the user. New standards create new challenges and emergent behaviors that call for academic scrutiny.

The question addressed here is: what are 802 and 802.11, and what do the various letters after ".11" signify?

2. IEEE 802

IEEE 802 is a project of the Institute of Electrical and Electronic Engineers (IEEE) LAN/MAN Standards Committee (LMSC). It was created in February 1980 – hence the name 802.

The LMSC is a committee of the IEEE Standards Association (IEEE-SA), which is the body that publishes completed standards and their amendments.

Within project 802, are various *working groups* – each of which defines one or more standards or recommended practices.

The currently active working groups are listed in Table 1.

Table 1 - IEEE 802 Working Groups

Working Group	Name
802.1	Higher Layer LAN protocols
802.3	Ethernet
802.11	Wireless LAN
802.15	Wireless PAN
802.16	Broadband Wireless Access
802.17	Resilient Packet Ring
802.18	Radio Regulatory technical advisory group
802.19	Coexistence technical advisory group
802.20	Mobile Broadband Wireless Access
802.21	Media Independent Handoff
802.22	Wireless Regional Area Networks

3. 802.11

The 802.11 working group working held its first meeting in September 1990 and issued the first draft of the 802.11 standard in early 1995, completed in late 1997. This document included a medium access controller (MAC) and physical layer (PHY) definitions for three media:

- Infrared
- Frequency Hopping Spread Spectrum (FHSS) in the 2.4GHz ISM band (1, 2 Mbps)
- Direct Sequence Spread Spectrum (DSSS) in the 2.4 GHz ISM band (1, 2 Mbps)

Originally the FHSS PHY was the most popular because of its lower cost and robustness. The author is not aware of any commercial products using the Infrared PHY. The DSSS PHY did not become popular until 802.11b increased the PHY rates to 5.5 and 11 Mbps. 802.11b is the version that has made wireless networking a popular commercial product.

Since the original version, 802.11 spawned *task groups* to produce amendments to the 802.11 standard. The first task group (TG) is called "TGa", and its amendment is called 802.11a, and so on. Each TG is authorized by the IEEE-SA and has a well-defined scope defined in its Project Authorization Request (PAR) document. The 802.11 task groups are described in Table 2. Those task groups that are currently active are indicated as such.

Table 2 - IEEE 802.11 Task Groups

Task Group	Description
TGa	This group developed a higher speed PHY based on orthogonal frequency division multiplexing (OFDM) in the 5GHz bands. The group cooperated with the European ETSI BRAN project and the two produced very similar

	PHY specifications. 802.11a has the advantage of several hundred of MHz of spectrum in the 5GHz band. However, it did not have the popular impact that 802.11b had due to the increased cost of operating at these frequencies. 802.11g made 802.11a speeds available in the 2.4GHz band. As the costs of 5GHz components has fallen, 802.11a looks increasingly attractive, and 802.11 a/b/g combination products are commonly available.
TGb	802.11b extended the DSSS PHY to support 5.5 and 11 Mbps. It has been the most successful version of 802.11 to date, and is now being replaced by 802.11g.
TGc	This defines 802.11 MAC procedures to support bridge operation. It is a supplement to 802.1D developed in cooperation with the 802.1 working group.
TGd	This extends support to additional regulatory domains and provides on-the-air signaling and control of parameters affected by the regulatory domain (such as channelization and hopping patterns).
TGe	TGe is still active, although it has nearly completed the standards process. It defines support for QoS for both distributed (EDCA) and centralized (HCCA) mechanisms. It supports QoS flows based on a user priority, suitable for connectionless data. Although the 802.1 MAC interface does not support connection-oriented data transfer, the 802.11e traffic specification (TSPEC) comes close to defining a connection – describing a flow in terms of size, rate, period and many other parameters. This makes 802.11e also suitable for periodic data such as VoIP. Additional improvements include power-saving (APSD) and additional efficiency gains through a selective acknowledgement (Block Ack).
TGf	This group developed recommended practices for an Inter-Access Point Protocol (IAPP) intended to provide interoperable management of the distribution system between APs from different manufacturers. It is uncertain what impact this recommended practice has had.
TGg	The 802.11g amendment essentially allows operation of the 802.11a OFDM modulation in the 2.4 GHz band. It provides 802.11a throughput at close to 802.11b prices. The challenge for 802.11g devices is to coexist with the installed base of 802.11b devices. This is achieved through various protection mechanisms, although there is some penalty in performance for operating in such an environment.
TGh	802.11h defined enhancements to 802.11a to support operation in the license exempt bands in Europe. It supports measurement and reporting of channel energy in order to provide dynamic frequency selection (DFS). It also provides control of transmit power (TPC).
TGi	This group was created to address issues and concerns with the original 802.11 WEP security mechanism. 802.11i defines two new mechanisms. TKIP is a medium strength mechanism designed for compatibility with hardware implementing the original 802.11 WEP security mechanism. WEP has proven to be susceptible to various types of attack, and TKIP provides a stopgap solution to these. AES provides the much stronger 128-bit block encryption, which is supported by newer hardware.
TGj	802.11j supports operation in Japan in the 4.9 and 5GHz bands. It extends the operation of the 802.11a PHY to operate in a 10MHz channel (half the channel width of 802.11a), and also allows longer range communication by increasing the turnaround interval to allow for longer propagation delays.
TGk Active	802.11k defines measurement of the radio channel that allows a device (a client device or an access point, or management software above) to make informed decisions relating to selecting an access point and selecting an operating channel. TGk is currently active.
TGma	This group provides maintenance changes (editorial and technical corrections) to 802.11-1999, 2003 edition (incorporating 802.11a-1999, 802.11b-1999, 802.11b-1999 corrigendum 1-2001, and 802.11d-2001).
TGn Active	802.11n will define modifications to both PHY and MAC layers to provide substantially higher throughput than 802.11 a/g. The project requires 100Mbps of useful throughput (at the top of the MAC interface), which requires about 200Mbps at the PHY. TGn is currently in its down-selection process to select between proposed solutions. Current proposals use multiple antenna technology and increased channel width to achieve significantly higher than the target. A maximum throughput of ~600Mbps at the PHY has been described, although first generation products are unlikely to support the optional features that achieve this figure. The PHY fixed overheads are not reduced, and aggregation and other enhancements are necessary in the MAC to restore an acceptable level of efficiency (~70%).
TGp Active	The TGp amendment will support communication between vehicles and the roadside and between vehicles while operating at speeds up to a minimum of 200 km/h for communication ranges up to 1000 meters. It will use the 5.850-5.925 GHz band within North

	America defined for this purpose.
TGr Active	TGr is chartered with developing a secure, fast BSS transition solution, when a Station (STA) roams from one Access Point (AP) to another AP, within an Extended Service Set (ESS). High BSS transition latencies using existing 802.11 mechanisms (including 802.11i Security addendum), along with a lack of inter-operability between STA and AP vendors in harmoniously executing these procedures in performing this transition, are technical hurdles for widespread deployment of Voice over Internet Protocol (VoIP) over 802.11 LANs. Delay and jitter sensitive applications like multimedia, video, and voice, which have to co-exist with traditional, intermittent data traffic, demand a flexible and scalable solution, which maintains the security guarantees provided by IEEE 802.11i. It is a market-driven requirement that the BSS transitions be executed with minimal latencies, while maintaining the same Quality of Service (QoS), and confidentiality and integrity protection, that the STA was being afforded at the existing AP, when the STA moves to the next AP. By some estimates, the procedures recommended by TGr should take less than 50 milliseconds, in order to be effective for the voice/video class of applications. TGr is progressing the merger of two proposals that were voted in at the January 2005 meeting, through a down-select process, from an initial pool of eight.
TGs Active	TGs is considering how to create a Mesh of APs to provide a Wireless Distribution System (WDS) using the existing IEEE 802.11 MAC/PHY layers. The mesh needs to support broadcast and directed transmissions over potentially multiple "hops" between APs. It has to be self-configuring. TGs are executing their selection process. They have a call for proposals out that will results in proposals being heard in July 2005.
TGu Active	TGu will define an amendment to IEEE 802.11 to support interworking with external networks. The group is currently working on its functional requirements.
TGv Active	This group will provide Wireless Network Management enhancements to the 802.11 MAC, and PHY, to extend prior work in radio measurement to effect a complete and coherent upper layer interface for managing 802.11 devices in wireless networks.
TGw	802.11 and later 802.11i established mechanisms to protect data frames, but does nothing to protect control frames internal to 802.11. For example, it is possible to forge disassociation requests. 802.11w is being chartered to extend the 802.11i protections to data frames. It is expected that 802.11w will begin its work in May 2005.

4. References

The most accessible source of information are the IEEE web-sites.

The IEEE 802 LMSC home page is: http://grouper.ieee.org/groups/802/

The IEEE 802.11 WG home page is: http://www.ieee802.org/11/. This contains more detailed description of the scope and status of the individual task groups.

The IEEE 802.11 WG working documents are available (after free registration) from: http://802wirelessworld.com.

Approved amendments are available for download here: http://standards.ieee.org/getieee802/.

The author speaks for himself. Views expressed by the author are not necessarily endorsed by his employer.

The SIGCOMM Digest

Erich Nahum
SIGCOMM Information Services Director
infodir_sigcomm (at) acm.org

Vint Cerf and Bob Kahn will be delivering the ACM Turing Award Lecture at SIGCOMM 2005, Monday August 22, 2005 (early evening) at the Irvine Auditorium on the University of Penn's campus. More information will be posted as it becomes available at the following URL:
http://www.acm.org/sigs/sigcomm/sigcomm2005/turinglecture.html

SIGCOMM's Jennifer Rexford is the 2004 recipient of ACM's Grace Murray Hopper Award, which honors the outstanding young computer professional of the year. Details at http://campus.acm.org/public/pressroom/press_releases/3_2005/gmh_award_3_15_2005.cfm

Nominations are open for the 2005 SigComm award. See guidelines at
http://www.acm.org/sigs/sigcomm/award.html and send nominations to Mark Crovella, crovella at cs dot bu dot edu**.**

Deadlines are also coming soon for the following SIGCOMM applications:
SigComm posters: May 1st
SigComm travel grants: May 20th

Upcoming deadlines of Conferences and publications financially supported and planned by SIGCOMM:

***Call for Papers* — SigComm Workshop on Mining Network Data (MineNet)**
Location: Philadelphia, PA, USA
Submission Deadline: April 6th
http://www.acm.org/sigs/sigcomm/sigcomm2005/cfp-minenet.html

***Call for Papers* — SigComm Workshop on Delay Tolerant Networking and Related Topics (WDTN-05)**
Location: Philadelphia, PA, USA
Submission Deadline: April 6th
http://www.acm.org/sigs/sigcomm/sigcomm2005/cfp-wdtn.html

***Call for Papers* — SigComm Workshop on Economics in P2P Networks**
Location: Philadelphia, PA, USA
Submission Deadline: April 15th
http://www.acm.org/sigs/sigcomm/sigcomm2005/cfp-wp2pecon.html

***Call for Papers* — SigComm Workshop on Experimental Approaches to Wireless Network Design and Analysis (E-WIND)**
Location: Philadelphia, PA, USA
Submission Deadline: April 15th
http://www.acm.org/sigs/sigcomm/sigcomm2005/cfp-wp2pecon.html

***Call for Papers* — SigComm Work-in-Progress**
Submission Deadline: May 13, 2005 (Full Papers)
Location: Philadelphia, PA, USA
Electronic submission at:
sigcomm05-posters@cs.ucr.edu
http://www.acm.org/sigs/sigcomm/sigcomm2005/poster.html

***Call for Submissions* — SigComm Travel Grants**
Submission Deadline: May 1, 2005
Location: Philadelphia, PA, USA
Electronic submission at sigcomm05-travel@research.att.com
http://www.acm.org/sigs/sigcomm/sigcomm2005/travel-grant.html

***Call for Papers* — 1st Symposium on Architectures for Networking and Communications Systems (ANCS 2005)**
Submission deadline: May 9, 2005
http://www.ancsconf.org
Location: Princeton, NJ, USA

Upcoming deadlines of Conferences planned in cooperation with SIGCOMM
(Remember that you, as a SIGCOMM member, are eligible for the conference "member rate" even it it's an IEEE or USENIX conference.)

***Call for Papers* — CoNext 2005**
Submission Deadline: May 6th, 2005
URL: http://dmi.ensica.fr/conext/

Upcoming deadlines of other conferences and journals related to the data communications area:

***Call for Papers* — ICCCN05 (Intl Conf on Computer Communications and Networks)**
Submission Deadline: April 15, 2005
http://icccn.sce.umkc.edu/icccn05/

***Call for Papers* — 2nd International Workshop on Deployment Models and Technologies for Broadband Community Networks**
Location: Orlando, Florida, USA
Submission Deadline: April 15, 2005
http://www.comnets.org

***Call for Papers* — First International Conference on Communication Systems Software and Middleware**
Submission Deadline: May 2nd, 2005
Location: New Delhi, India
http://www.comsware.org

***Call for Papers* — 13th Annual Meeting of the IEEE/ACM International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)**
Location: Atlanta, Georgia, USA
http://www.mascots-conference.org

***Call for Papers* — 13th International Conference on Network Protocols (ICNP)**
Submission Deadline: May 6th 2005
Location: Lovely Boston, MA, USA
http://csr.bu.edu/icnp2005/

***Call for Papers* — IEEE Conference on Local Computer Networks (LCN 2005)**
Location: Sydney, Australia,
Paper submission deadline: 10 May 2005
http://www.ieeelcn.org/

***Call for Tutorials* — 13th International Conference on Network Protocols (ICNP)**
Submission Deadline: June 3rd, 2005
Location: Lovely Boston, MA, USA
http://csr.bu.edu/icnp2005/

***Call for Papers* — The 3rd Workshop on Rapid Malcode (WORM)**
Submission Deadline: June 23rd, 2005
Location: Fairfax, Virginia, USA
http://www1.cs.columbia.edu/~angelos/worm05/

acm sigcomm