

End-to-End Available Bandwidth: Measurement Methodology, Dynamics, and Relation With TCP Throughput

Manish Jain and Constantinos Dovrolis, *Member, IEEE*

Abstract—The available bandwidth (avail-bw) in a network path is of major importance in congestion control, streaming applications, quality-of-service verification, server selection, and overlay networks. We describe an end-to-end methodology, called self-loading periodic streams (SLoPS), for measuring avail-bw. The basic idea in SLoPS is that the one-way delays of a periodic packet stream show an increasing trend when the stream’s rate is higher than the avail-bw. We implemented SLoPS in a tool called *pathload*. The accuracy of the tool has been evaluated with both simulations and experiments over real-world Internet paths. *Pathload* is nonintrusive, meaning that it does not cause significant increases in the network utilization, delays, or losses. We used *pathload* to evaluate the variability (“dynamics”) of the avail-bw in internet paths. The avail-bw becomes significantly more variable in heavily utilized paths, as well as in paths with limited capacity (probably due to a lower degree of statistical multiplexing). We finally examine the relation between avail-bw and TCP throughput. A persistent TCP connection can be used to roughly measure the avail-bw in a path, but TCP saturates the path and increases significantly the path delays and jitter.

Index Terms—Active probing, bottleneck bandwidth, bulk transfer capacity, network capacity, packet pair dispersion.

I. INTRODUCTION

THE CONCEPT of available bandwidth (avail-bw) has been of central importance throughout the history of packet networks, in both research and practice. In the context of transport protocols, the robust and efficient use of avail-bw has always been a major issue, including Jacobson’s TCP [9]. The avail-bw is also a crucial parameter in capacity provisioning, routing and traffic engineering, quality-of-service (QoS) management, streaming applications, server selection, and several other areas.

Researchers have been trying to create end-to-end measurement algorithms for avail-bw over the last 15 years. From Keshav’s *packet pair* [15] to Carter and Crovella’s *cprobe* [6], the objective was to measure end-to-end avail-bw accurately, quickly, and without affecting the traffic in the path, i.e., non-intrusively. What makes the measurement of avail-bw difficult

is, first, that there is no consensus on how to precisely define it, second, that it varies with time, and third, that it exhibits high variability in a wide range of timescales.

A. Definitions

We next define the avail-bw in an intuitive but precise manner. The definition does not depend on higher level issues, such as the transport protocol or the number of flows that can capture the avail-bw in a path.

A network path \mathcal{P} is a sequence of H store-and-forward links that transfer packets from a sender \mathcal{SND} to a receiver \mathcal{RCV} . We assume that the path is fixed and unique, i.e., no routing changes or multipath forwarding occur during the measurements. Each link i can transmit data with a rate C_i bits per second, which is referred to as *link capacity*. Two throughput metrics that are commonly associated with \mathcal{P} are the end-to-end *capacity* C and *available bandwidth* A . The capacity is defined as

$$C \equiv \min_{i=1 \dots H} C_i \quad (1)$$

and it is the maximum rate that the path can provide to a flow, when there is no other traffic in \mathcal{P} .

Suppose that link i transmitted $C_i u_i(t_0, t_0 + \tau)$ bits during a time interval $(t_0, t_0 + \tau)$. The term $u_i(t_0, t_0 + \tau)$, or simply $u_i^\tau(t_0)$, is the average *utilization* of link i during $(t_0, t_0 + \tau)$, with $0 \leq u_i^\tau(t_0) \leq 1$. Intuitively, the avail-bw $A_i^\tau(t_0)$ of link i in $(t_0, t_0 + \tau)$ can be defined as the fraction of the link’s capacity that has not been utilized during that interval, i.e.,

$$A_i^\tau(t_0) \equiv C_i [1 - u_i^\tau(t_0)]. \quad (2)$$

Extending this concept to the entire path, the end-to-end avail-bw $A^\tau(t_0)$ during $(t_0, t_0 + \tau)$ is the minimum avail-bw among all links in \mathcal{P}

$$A^\tau(t_0) \equiv \min_{i=1 \dots H} \{C_i [1 - u_i^\tau(t_0)]\}. \quad (3)$$

Thus, the end-to-end avail-bw is defined as the maximum rate that the path can provide to a flow, without reducing the rate of the rest of the traffic in \mathcal{P} .

To avoid the term *bottleneck link*, which has been widely used in the context of both capacity and avail-bw, we introduce two new terms. The *narrow link* is the link with the minimum capacity, and it determines the capacity of the path. The *tight link* is the link with the minimum avail-bw, and it determines the avail-bw of the path.

The parameter τ in (3) is the avail-bw *averaging timescale*. If we consider $A^\tau(t)$ as a stationary random process, the variance

Manuscript received October 21, 2002; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor J. Rexford. This work was supported by the Scientific Discovery through Advanced Computing (SciDAC) Program of the Department of Energy under Award DE-FC02-01ER25467. This paper was presented in part at the ACM SIGCOMM Conference, Pittsburgh, PA, August 2002.

The authors are with the College of Computing, Georgia Institute of Technology, Atlanta, GA 30332 USA (e-mail: jain@cc.gatech.edu; dovrolis@cc.gatech.edu).

Digital Object Identifier 10.1109/TNET.2003.815304

$\text{Var}\{A^\tau\}$ of the process decreases as the averaging timescale τ increases. We note that if A^τ is self-similar, the variance $\text{Var}\{A^\tau\}$ decreases slowly, in the sense that the decrease of $\text{Var}\{A^\tau\}$ as τ increases is slower than the reciprocal of τ [19].

B. Main Contributions

In this paper, we present an original end-to-end avail-bw measurement methodology, called self-loading periodic streams (SLoPS). The basic idea in SLoPS is that the one-way delays of a periodic packet stream show an increasing trend when the stream's rate is higher than the avail-bw. SLoPS has been implemented in a measurement tool called *pathload*. The tool has been verified experimentally, by comparing its results with MRTG utilization graphs for the path links [25]. We have also evaluated *pathload* in a controlled and reproducible environment using NS simulations. The simulations show that *pathload* reports a range that includes the average avail-bw in a wide range of load conditions and path configurations. The tool underestimates the avail-bw, however, when the path includes several tight links. The *pathload* measurements are nonintrusive, meaning that they do not cause significant increases in the network utilization, delays, or losses. *Pathload* is described in detail in [12]; here we describe the tool's salient features and show a few experimental and simulation results to evaluate the tool's accuracy.

An important feature of *pathload* is that, instead of reporting a single figure for the average avail-bw in a time interval $(t_0, t_0 + \Theta)$, it estimates the range in which the avail-bw process $A^\tau(t)$ varies in $(t_0, t_0 + \Theta)$, when it is measured with an averaging timescale $\tau < \Theta$. The timescales τ and Θ are related to two tool parameters, namely, the stream duration and the fleet duration.

We have used *pathload* to estimate the variability (or "dynamics") of the avail-bw in different paths and load conditions. An important observation is that the avail-bw becomes more variable as the utilization of the tight link increases (i.e., as the avail-bw decreases). Similar observations are made for paths of different capacity that operate at about the same utilization. Specifically, the avail-bw shows higher variability in paths with smaller capacity, probably due to a lower degree of statistical multiplexing.

Finally, we examined the relation between the avail-bw and the throughput of a "greedy" TCP connection, i.e., a persistent bulk transfer with sufficiently large advertised window. Our experiments show that such a greedy TCP connection can be used to roughly measure the end-to-end avail-bw, but TCP saturates the path, increases significantly the delays and jitter, and potentially causes losses to other TCP flows. The increased delays and losses in the path cause other TCP flows to slow down, allowing the greedy TCP connection to grab more bandwidth than what was previously available.

C. Overview

Section II summarizes previous work on bandwidth estimation. Section III explains the SLoPS measurement methodology. Section IV describes the *pathload* implementation. Section V presents simulation and experimental verification results. Section VI evaluates the dynamics of avail-bw using *pathload*.

Section VII examines the relation between avail-bw and TCP throughput. Section VIII shows that *pathload* is *not* network intrusive, and Section IX concludes the paper.

II. RELATED WORK

Although there are several bandwidth estimation tools, most of them measure capacity rather than avail-bw. Specifically, *pathchar* [10], *clink* [8], *pchar* [20], and the tailgating technique [17] measure per-hop capacity. Also, *bprobe* [6], *nettimer* [16], *pathrate* [7], and the PBM methodology [28] measure end-to-end capacity.

Allman and Paxson noted that an avail-bw estimate can give a more appropriate value for the *ssthresh* variable, improving the slow-start phase of TCP [2]. They recognized, however, the complexity of measuring avail-bw from the timing of TCP packets, and they focused instead on capacity estimates.

The first tool that attempted to measure avail-bw was *cprobe* [6], which estimated the avail-bw based on the dispersion of long packet trains at the receiver. A similar approach was taken in *pipechar* [14]. The underlying assumption in these works is that the dispersion of long packet trains is inversely proportional to the avail-bw. Recently, however, [7] showed that this is not the case. The dispersion of long packet trains does not measure the avail-bw in a path; instead, it measures a different throughput metric which is referred to as the asymptotic dispersion rate (ADR).

A different avail-bw measurement technique, called Delphi, was proposed in [29]. The main idea in Delphi is that the spacing of two probing packets at the receiver can provide an estimate of the amount of traffic at a link, provided that the queue of that link does not empty between the arrival times of the two packets. Delphi assumes that the path can be well modeled by a single queue. This model is not applicable when the tight and narrow links are different, and it interprets queueing delays anywhere in the path as queueing delays at the tight link.

Another technique, called TOPP, for measuring avail-bw was proposed in [23]. TOPP uses sequences of packet pairs sent to the path at increasing rates. From the relation between the input and output rates of different packet pairs, one can estimate the avail-bw and the capacity of the tight link in the path. In certain path configurations, it is possible to also measure the avail-bw and capacity of other links in the path. Both TOPP and our technique, SLoPS, are based on the observation that the queueing delays of successive periodic probing packets increase when the probing rate is higher than the avail-bw in the path. The two techniques, however, are quite different in the actual algorithm they use to estimate the avail-bw. A detailed comparison of the two estimation methods is an important task for further research.

Paxson defined and measured a *relative avail-bw metric* β [28]. This metric is based on the one-way delay variations of a flow's packets. β measures the proportion of packet delays that are due to the flow's own load. If each packet is only queued behind its predecessors, the path is considered empty and $\beta \approx 1$. On the other hand, if the observed delay variations are mostly due to cross traffic, the path is considered saturated and $\beta \approx 0$. Unfortunately, there is no direct relationship between β and the avail-bw in the path or the utilization of the tight link.

An issue of major importance is the predictability of the avail-bw. Paxson’s metric β is fairly predictable: on average, a measurement of β at a given path falls within 10% of later β measurements for periods that last for several hours [28]. Balakrishnan *et al.* examined the throughput stationarity of successive Web transfers to a set of clients [4]. The throughput to a given client appeared to be piecewise stationary in time scales that extend for hundreds of minutes. Additionally, the throughput of successive transfers to a given client varied by less than a factor of 2 over three hours. A more elaborate investigation of the stationarity of avail-bw was recently published in [30]. Zhang *et al.* measured the TCP throughput of 1-MB transfers every minute for 5 h. Their dataset includes 49 000 connections in 145 distinct paths. They found out that the throughput *change-free regions*, i.e., the time periods in which the throughput time series can be modeled as a stationary process, often last for more than an hour. Also, the throughput stays in a range with peak-to-peak variation of a factor of three for several hours. An important point is that these previous works did not correlate the variability of avail-bw with the operating conditions in the underlying paths. We attempt such an approach in Section VI.

To characterize the ability of a path to transfer large files using TCP, the IETF recommends the bulk transfer capacity (BTC) metric [22]. The BTC of a path in a certain time period is the throughput of a persistent (or “bulk”) TCP transfer through that path, when the transfer is only limited by the network resources and not by buffer or other limitations at the end-systems. The BTC can be measured with *Treno* [21] or *cap* [1]. It is important to distinguish between the avail-bw and the BTC of a path. The former gives the total spare capacity in the path, independent of which transport protocol attempts to capture it. The latter, on the other hand, depends on TCP’s congestion control, and it is the maximum throughput that a single and persistent TCP connection can get. Parallel persistent connections, or a large number of short TCP connections (“mice”), can obtain an aggregate throughput that is higher than the BTC. The relation between BTC and avail-bw is investigated in Section VII.

Finally, several congestion control algorithms, such as those proposed in [3], [5], [13], and [24], infer that the path is congested (or that there is no avail-bw) when the round-trip delays in the path start increasing. This is similar to the basic idea of our estimation methodology: the one-way delays of a periodic packet stream are expected to show an increasing trend when the stream’s rate is higher than the avail-bw. The major difference between SLoPS and those proposals is that we use the relation between the probing rate and the observed delay variations to develop an elaborate avail-bw measurement algorithm, rather than a congestion control algorithm. Also, SLoPS is based on periodic rate-controlled streams, rather than window-controlled transmissions, allowing us to compare a certain rate with the avail-bw more reliably.

III. SELF-LOADING PERIODIC STREAMS

In this section, we describe the SLoPS measurement methodology. A periodic stream in SLoPS consists of K packets of size L , sent to the path at a constant rate R . If the stream rate R is

higher than the avail-bw A , then the one-way delays of successive packets at the receiver show an increasing trend. We first illustrate this fundamental effect in its simplest form through an analytical model with stationary and fluid cross traffic. Then, we show how to use this “increasing delays” property in an iterative algorithm that measures end-to-end avail-bw. Finally, we depart from the previous fluid model and observe that the avail-bw may vary during a stream. This requires us to refine SLoPS in several ways, which is the subject of the next section.

A. SLoPS With Fluid Cross Traffic

Consider a path from \mathcal{SND} to \mathcal{RCV} that consists of H links, $i = 1, \dots, H$. The capacity of link i is C_i . We consider a stationary (i.e., time invariant) and fluid model for the cross traffic in the path. So, if the avail-bw at link i is A_i , the utilization is $u_i = (C_i - A_i)/C_i$ and there are $u_i C_i \tau$ bytes of cross traffic departing from and arriving at link i in any interval of length τ . Also, assume that the links follow the first-come–first-served queueing discipline,¹ and that they are adequately buffered to avoid losses. We ignore any propagation or fixed delays in the path, as they do not affect the delay variation between packets. The avail-bw A in the path is determined by the tight link² $t \in \{1, \dots, H\}$ with

$$A_t = \min_{i=1 \dots H} A_i = \min_{i=1 \dots H} C_i(1 - u_i) = A. \quad (4)$$

Suppose that \mathcal{SND} sends a periodic stream of K packets to \mathcal{RCV} at a rate R_0 , starting at an arbitrary time instant. The packet size is L bytes, and so packets are sent with a period of $T = L/R_0$ time units. The one-way delay (OWD) D^k from \mathcal{SND} to \mathcal{RCV} of packet k is

$$D^k = \sum_{i=1}^H \left(\frac{L}{C_i} + \frac{q_i^k}{C_i} \right) = \sum_{i=1}^H \left(\frac{L}{C_i} + d_i^k \right) \quad (5)$$

where q_i^k is the queue size at link i upon the arrival of packet k (q_i^k does not include packet k), and $d_i^k = q_i^k/C_i$ is the queueing delay of packet k at link i . The OWD difference between two successive packets k and $k + 1$ is

$$\Delta D^k \equiv D^{k+1} - D^k = \sum_{i=1}^H \frac{\Delta q_i^k}{C_i} = \sum_{i=1}^H \Delta d_i^k \quad (6)$$

where $\Delta q_i^k \equiv q_i^{k+1} - q_i^k$, and $\Delta d_i^k \equiv \Delta q_i^k/C_i$.

We can now show that if $R_0 > A$, then the K packets of the periodic stream will arrive at \mathcal{RCV} with increasing OWDs, while if $R_0 \leq A$ the stream packets will encounter equal OWDs. This property is stated next and proved in the Appendix.

Proposition 1: If $R_0 > A$, then $\Delta D^k > 0$ for $k = 1, \dots, K - 1$. Else, if $R_0 \leq A$, $\Delta D^k = 0$ for $k = 1, \dots, K - 1$.

One may think that the avail-bw A can be computed directly from the rate at which the stream arrives at \mathcal{RCV} . This is the approach followed in packet train dispersion techniques. The

¹In links with per-flow or per-class queueing, SLoPS can monitor the sequence of queues that the probing packets go through.

²If there are more than one links with avail-bw A , the tight link is the first of them, without loss of generality.

following result, however, shows that, in a general path configuration, this would be possible only if the capacity and avail-bw of all links (except the avail-bw of the tight link) are *a priori* known.

Proposition 2: The rate R_H of the packet stream at \mathcal{RCV} is a function, in the general case, of C_i and A_i for all $i = 1, \dots, H$.

This result follows from the proof in the Appendix [apply recursively (19) until $i = H$].

B. Iterative Algorithm to Measure A

Based on Proposition 1, we can construct an iterative algorithm for the end-to-end measurement of A . Suppose that \mathcal{SND} sends a periodic stream n with rate $R(n)$. The receiver analyzes the OWD variations of the stream, based on Proposition 1, to determine whether $R(n) > A$ or not. Then, \mathcal{RCV} notifies \mathcal{SND} about the relation between $R(n)$ and A . If $R(n) > A$, \mathcal{SND} sends the next periodic stream $n+1$ with rate $R(n+1) < R(n)$. Otherwise, the rate of stream $n+1$ is $R(n+1) > R(n)$.

Specifically, $R(n+1)$ can be computed as follows:

$$\begin{aligned} \text{If } R(n) > A, \quad R^{\max} &= R(n) \\ \text{If } R(n) \leq A, \quad R^{\min} &= R(n) \\ R(n+1) &= (R^{\max} + R^{\min})/2. \end{aligned} \quad (7)$$

R^{\min} and R^{\max} are lower and upper bounds for the avail-bw after stream n , respectively. Initially, $R^{\min} = 0$ and R^{\max} can be set to a sufficiently high value $R_0^{\max} > A$.³ The algorithm terminates when $R^{\max} - R^{\min} \leq \omega$, where ω is the user-specified estimation resolution. If the avail-bw A does not vary with time, the previous algorithm will converge to a range $[R^{\min}, R^{\max}]$ that includes A after $\lceil \log_2(R_0^{\max}/\omega) \rceil$ streams.

C. SLoPS With Real Cross Traffic

We have assumed so far that the avail-bw A is constant during the measurement process. In reality, the avail-bw may vary because of two reasons. First, the avail-bw process $A^\tau(t)$ of (3) may be nonstationary, and so its expected value may also be a function of time. Even if $A^\tau(t)$ is stationary, however, the process A^τ can have a significant statistical variability around its (constant) mean $E[A^\tau]$, and to make things worse, this variability may extend over a wide range of timescales τ . How can we refine SLoPS to deal with the dynamic nature of the avail-bw process?

To gain some insight into this issue, Figs. 1–3 show the OWD variations in three periodic streams that crossed a 12-hop path from Univ-Oregon to Univ-Delaware. All three streams have $K = 100$ packets with $T = 100 \mu\text{s}$. The 5-min average avail-bw in the path during these measurements was about 74 Mb/s, according to the MRTG utilization graph of the tight link. In Fig. 1, the stream rate is $R = 96$ Mb/s, i.e., higher than the long-term avail-bw. Notice that the OWDs between successive packets are not strictly increasing, as one would expect from Proposition 1, but overall, the stream OWDs have a clearly increasing trend. This is shown both by the fact that most packets have a higher OWD than their predecessors, and because the OWD of the last packet is about 4 ms larger than the OWD of the

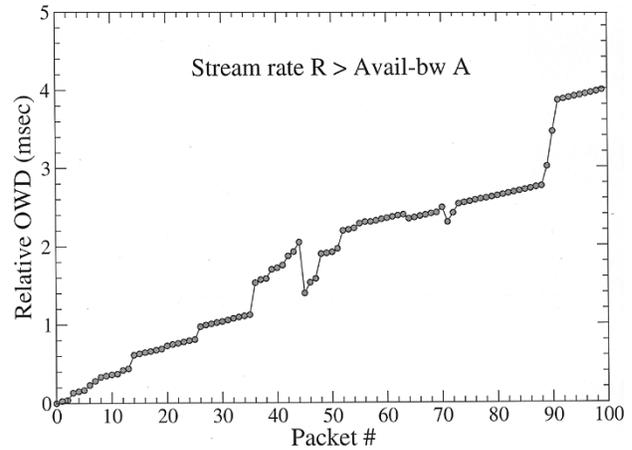


Fig. 1. OWD variations for a periodic stream when $R > A$.

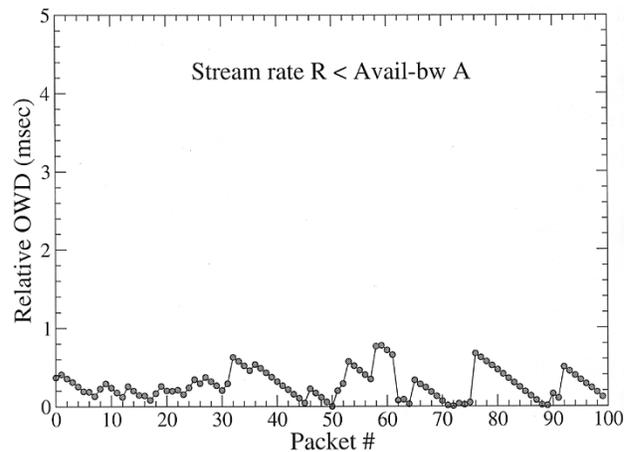


Fig. 2. OWD variations for a periodic stream when $R < A$.

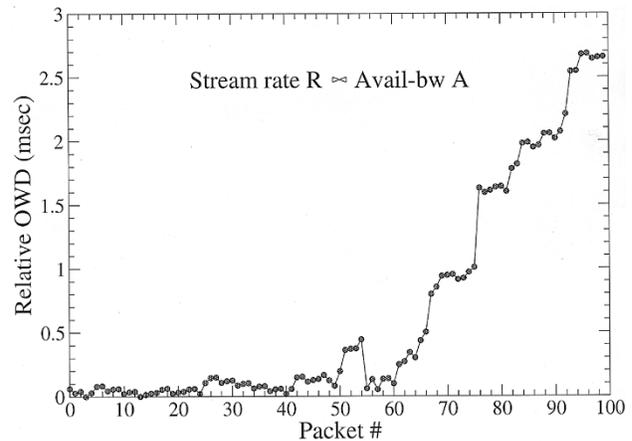


Fig. 3. OWD variations for a periodic stream when $R \approx A$.

first packet. On the other hand, the stream of Fig. 2 has a rate $R = 37$ Mb/s, i.e., lower than the long-term avail-bw. Even though there are short-term intervals in which we observe increasing OWDs, there is clearly not an increasing trend in the stream. The third stream, in Fig. 3, has a rate $R = 82$ Mb/s. The stream does not show an increasing trend in the first half, indicating that the avail-bw during that interval is higher than R . The situation changes dramatically, however, after roughly the

³A better way to initialize R^{\max} is described in [12].

60th packet. In that second half of the stream there is a clear increasing trend, showing that the avail-bw decreases to less than R .

The previous example motivates two important refinements in the SLoPS methodology. First, instead of analyzing the OWD variations of a stream, expecting one of the two cases of Proposition 1 to be strictly true for every pair of packets, we should instead watch for the presence of an overall increasing trend during the entire stream. Second, we have to accept the possibility that the avail-bw may vary around rate R during a probing stream. In that case, there is no strict ordering between R and A and, thus, a third possibility comes up, which we refer to as the “grey region” (denoted as $R \bowtie A$). The next section gives a concrete specification of these two refinements, as implemented in *pathload*.

IV. MEASUREMENT TOOL: *PATHLOAD*

We implemented SLoPS in a tool called *pathload*. *Pathload*, together with its experimental verification, is described in detail in [12]. In this section, we provide a description of the tool’s salient features.

Pathload consists of a process SND running at the sender and a process RCV running at the receiver. The stream packets use UDP, while a TCP connection between the two end points controls the measurements.

Clock and Timing Issues: SND timestamps each packet upon its transmission. So RCV can measure the relative OWD D^k of packet k that differs from the actual OWD by a certain offset. This offset is due to the nonsynchronized clocks of the end-hosts. Since we are only interested in OWD differences though, a constant offset in the measured OWDs does not affect the analysis. Clock skew can be a potential problem (and there are algorithms to remove its effects) but not in *pathload*. The reason is that each stream lasts for only a few milliseconds (Section IV), and so the skew during a stream is in the order of nanoseconds, much less than the OWD variations due to queueing.

Stream Parameters: A stream consists of K packets of size L sent at a constant rate R . R is adjusted at runtime for each stream, as described in Section IV. The packet interspacing T is normally set to T_{\min} , which is based on the minimum possible period that the end hosts can achieve. The receiver measures the interspacing T with which the packets left the sender, using the SND timestamps, to detect context switches and other rate deviations [12].

Given R and T , the packet size is computed as $L = RT$. L , however, has to be smaller than the path MTU L^{\max} (to avoid fragmentation), and larger than a minimum possible size $L^{\min} = 200$ B. The reason for the L^{\min} constraint is to reduce the effect of layer-2 headers on the stream rate [26]. If $R < L^{\min}/T_{\min}$, the interspacing T is increased to L^{\min}/R . The maximum rate that *pathload* can generate, and thus the maximum avail-bw that it can measure, is L^{\max}/T_{\min} .

The stream length K is chosen based on two constraints. First, a stream should be relatively short, so that it does not cause large queues and potential losses in the path routers. Second,

K controls the stream duration $V = KT$, which is related to the averaging timescale τ (see Section VI-C). A larger K (longer stream) increases τ and, thus, reduces the variability in the measured avail-bw. In *pathload*, the default value for K is 100 packets.

Detecting an Increasing OWD Trend: Suppose that the (relative) OWDs of a particular stream are D^1, D^2, \dots, D^K . As a preprocessing step, we partition these measurements into $\Gamma = \sqrt{K}$ groups of Γ consecutive OWDs. Then, we compute the median OWD \hat{D}^k of each group. *Pathload* analyzes the set $\{\hat{D}^k, k = 1, \dots, \Gamma\}$, which is more robust to outliers and errors.

We use two complementary statistics to check if a stream shows an increasing trend. The pairwise comparison test (PCT) metric of a stream is

$$S_{\text{PCT}} = \frac{\sum_{k=2}^{\Gamma} I(\hat{D}^k > \hat{D}^{k-1})}{\Gamma - 1} \quad (8)$$

where $I(X)$ is one if X holds, and zero otherwise. PCT measures the fraction of consecutive OWD pairs that are increasing, and so $0 \leq S_{\text{PCT}} \leq 1$. If the OWDs are independent, the expected value of S_{PCT} is 0.5. If there is a strong increasing trend, S_{PCT} approaches one.

The pairwise difference test (PDT) metric of a stream is

$$S_{\text{PDT}} = \frac{\hat{D}^{\Gamma} - \hat{D}^1}{\sum_{k=2}^{\Gamma} |\hat{D}^k - \hat{D}^{k-1}|} \quad (9)$$

PDT quantifies how strong is the start-to-end OWD variation, relative to the OWD absolute variations during the stream. Note that $-1 \leq S_{\text{PDT}} \leq 1$. If the OWDs are independent, the expected value of S_{PDT} is zero. If there is a strong increasing trend, S_{PDT} approaches one.

There are cases in which one of the two metrics is better than the other in detecting an increasing trend [12]. Consequently, if either the PCT or PDT metrics shows an increasing trend, *pathload* characterizes the stream as *type I*, i.e., increasing. Otherwise, the stream is considered to be *type N*, i.e., nonincreasing. In the current release of *pathload*, the PCT metric shows an increasing trend if $S_{\text{PCT}} > 0.55$, while the PDT shows increasing trend if $S_{\text{PDT}} > 0.4$. The effect of the PCT and PDT thresholds (0.55 and 0.4, respectively) on the *pathload* accuracy is shown in the next section.

Fleets of Streams: *Pathload* does not determine whether $R > A$ based on a single stream. Instead, it sends a *fleet* of N streams, so that it samples whether $R > A$ N successive times. All streams in a fleet have the same rate R . Each stream is sent only when the previous stream has been acknowledged, to avoid a backlog of streams in the path. So, there is always an idle interval Δ between streams, which is larger than the round-trip time (RTT) of the path. The duration of a fleet is $U = N(V + \Delta) = N(KT + \Delta)$. Given V and Δ , N determines the fleet duration, which is related to the *pathload* measurement latency. The default value for N is 12 streams. The effect of N is discussed in Section VI-D.

The average *pathload* rate during a fleet of rate R is

$$\frac{NKL}{N(V + \Delta)} = R \frac{1}{1 + \frac{\Delta}{V}}.$$

In order to limit the average *pathload* rate to less than 10% of R , the current version of *pathload* sets the interstream latency to $\Delta = \max\{\text{RTT}, 9V\}$.

If a stream encounters excessive losses ($>10\%$), or if more than a number of streams within a fleet encounter moderate losses ($>3\%$), the entire fleet is aborted and the rate of the next fleet is decreased. For more details, see [12].

Grey Region: If a large fraction f of the N streams in a fleet are of type I, the entire fleet shows an increasing trend and we infer that the fleet rate is larger than the avail-bw ($R > A$). Similarly, if a fraction f of the N streams are of type N, the fleet does not show an increasing trend and we infer that the fleet rate is smaller than the avail-bw ($R < A$). It can happen, though, that less than $N \times f$ streams are of type I, and also that less than $N \times f$ streams are of type N. In that case, some streams “sampled” the path when the avail-bw was less than R (type I), and some others when it was more than R (type N). We say, then, that the fleet rate R is in the grey region of the avail-bw, and write $R \bowtie A$. The interpretation that we give to the grey region is that when $R \bowtie A$, the avail-bw process $A^\tau(t)$ during that fleet varied above and below rate R , causing some streams to be of type I and some others to be of type N. The averaging timescale τ here is related to the stream duration V . We discuss the effect of f on the *pathload* outcome in the next section.

Rate Adjustment Algorithm: After a fleet n of rate $R(n)$ is over, *pathload* determines whether $R(n) > A$, $R(n) < A$, or $R(n) \bowtie A$. The iterative algorithm that determines the rate $R(n+1)$ of the next fleet is quite similar to the binary-search approach of (7). However, there are two important differences.

First, together with the upper and lower bounds for the avail-bw R^{\max} and R^{\min} , *pathload* also maintains upper and lower bounds for the grey region, namely G^{\max} and G^{\min} . When $R(n) \bowtie A$, one of these bounds is updated depending on whether $G^{\max} < R(n) < R^{\max}$ (update G^{\max}), or $G^{\min} > R(n) > R^{\min}$ (update G^{\min}). If a grey region has not been detected up to that point, the next rate $R(n+1)$ is chosen, as in (7), halfway between R^{\min} and R^{\max} . If a grey region has been detected, $R(n+1)$ is set halfway between G^{\max} and R^{\max} when $R(n) = G^{\max}$, or halfway between G^{\min} and R^{\min} when $R(n) = G^{\min}$. The complete rate adjustment algorithm, including the initialization steps, is given in [12]. It is important to note that this binary-search approach succeeds in converging to the avail-bw, as long as the avail-bw variation range is strictly included in the $[R^{\min}, R^{\max}]$ range. The experimental and simulation results of the next section show that this is the case generally, with the exception of paths that include several tight links.

The second difference is that *pathload* terminates not only when the avail-bw has been estimated within a certain resolution ω (i.e., $R^{\max} - R^{\min} \leq \omega$), but also when $R^{\max} - G^{\max} \leq \chi$ and $G^{\min} - R^{\min} \leq \chi$, i.e., when both avail-bw boundaries are within χ from the corresponding grey-region boundaries. The parameter χ is referred to as grey-region resolution.

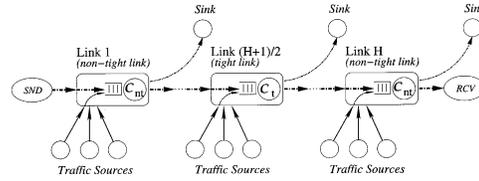


Fig. 4. Simulation topology.

The tool eventually reports the range $[R^{\min}, R^{\max}]$.

Measurement Latency: Since *pathload* is based on an iterative algorithm, it is hard to predict how long a measurement will take. For the default tool parameters, and for a path with $A \approx 100$ Mb/s and $\Delta = 100$ ms, the tool needs less than 15 s to produce a final estimate. The measurement latency increases as the absolute magnitude of the avail-bw and/or the width of the grey region increases, and it also depends on the resolution parameters ω and χ .

V. VERIFICATION

The objective of this section is to evaluate the accuracy of *pathload* with both NS simulations and experiments over real Internet paths.

A. Simulation Results

The following simulations evaluate the accuracy of *pathload* in a controlled and reproducible environment under various load conditions and path configurations. Specifically, we implemented the *pathload* sender (\mathcal{SND}) and receiver (\mathcal{RCV}) in application-layer NS modules. The functionality of these modules is identical as in the original *pathload*, with the exception of some features that are not required in a simulator (such as the detection of context switches).

In the following, we simulate the H -hop topology of Fig. 4. The *pathload* packets enter the path in hop 1 and exit at hop H . The hop in the middle of the path is the tight link, and it has capacity C_t , avail-bw A_t , and utilization u_t . We refer to the rest of the links as *nontight*, and consider the case where they all have the same capacity C_{nt} , avail-bw A_{nt} , and utilization u_{nt} . Cross-traffic is generated at each link from ten random sources that, unless specified otherwise, generate Pareto interarrivals with $\alpha = 1.9$. The cross-traffic packet sizes are distributed as follows: 40% are 40 B, 50% are 550 B, and 10% are 1500 B. The end-to-end propagation delay in the path is 50 ms, and the links are sufficiently buffered to avoid packet losses. Another important factor is the relative magnitude of the avail-bw in the nontight links A_{nt} and in the tight link A_t . To quantify this, we define the *path tightness factor* as

$$\beta = \frac{A_{nt}}{A_t} \geq 1. \quad (10)$$

Unless specified otherwise, the default parameters in the following simulations are $H = 5$ hops, $C_t = 10$ Mb/s, $\beta = 5$, $u_t = u_{nt} = 60\%$, $f = 0.7$, while the PCT threshold is 0.55 and the PDT threshold is 0.4.

Fig. 5 examines the accuracy of *pathload* in four tight link utilization values, ranging from light load ($u_t = 30\%$) to heavy

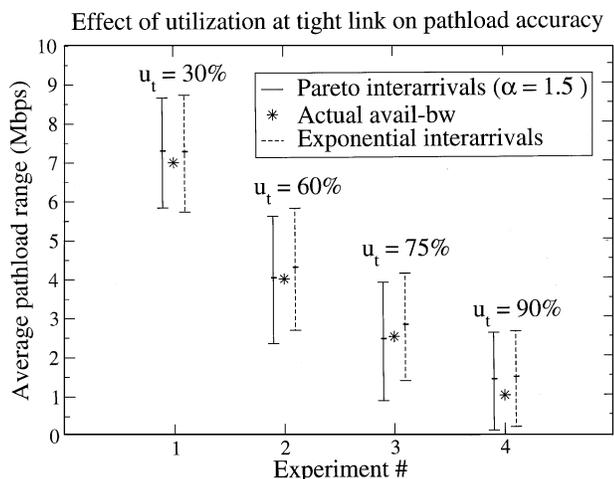


Fig. 5. Simulation results for different traffic types and tight link loads.

load ($u_t = 90\%$). We also consider two cross-traffic models: exponential interarrivals and Pareto interarrivals with infinite variance ($\alpha = 1.5$). For each utilization and traffic model, we run *pathload* 50 times to measure the avail-bw in the path. After each run, the tool reports a range $[R^{\min}, R^{\max}]$ in which the avail-bw varies. The *pathload* range that we show in Fig. 5 results from averaging the 50 lower bounds R^{\min} and the 50 upper bounds R^{\max} . The coefficient of variation for the 50 samples of R^{\min} and R^{\max} in the following simulations was typically between 0.10 and 0.30.

The main observation in Fig. 5 is that *pathload* produces a range that includes the average avail-bw in the path, in both light and heavy load conditions at the tight link. This is true with both the smooth interarrivals of Poisson traffic, and with the infinite-variance Pareto model. For instance, when the avail-bw is 4 Mb/s, the average *pathload* range in the case of Pareto interarrivals is from 2.4 to 5.6 Mb/s. It is also important to note that the center of the *pathload* range is relatively close to the average avail-bw. In Fig. 5, the maximum deviation between the average avail-bw and the center of the *pathload* range is when the former is 1 Mb/s and the latter is 1.5 Mb/s.

The next issue is whether the accuracy of *pathload* depends on the number and load of the nontight links. Fig. 6 shows, as in the previous paragraph, 50-sample average *pathload* ranges for four different utilization points u_{nt} at the nontight links, and for two different path lengths H . Since $C_t = 10$ Mb/s and $u_t = 60\%$, the end-to-end avail-bw in these simulations is 4 Mb/s. The path tightness factor is $\beta = 5$, and so the avail-bw in the nontight links is $A_{nt} = 20$ Mb/s. So, even when there is significant load and queueing at the nontight links (which is the case when $u_{nt} = 90\%$), the end-to-end avail-bw is quite lower than the avail-bw in the $H - 1$ nontight links.

The main observation in Fig. 6 is that *pathload* estimates a range that includes the actual avail-bw in all cases, independent of the number of nontight links or of their load. Also, the center of the *pathload* range is within 10% of the average avail-bw A_t . So, when the end-to-end avail-bw is mostly limited by a single link, *pathload* is able to estimate accurately the avail-bw in a multihop path even when there are several other queueing points. The nontight links introduce noise in the OWD

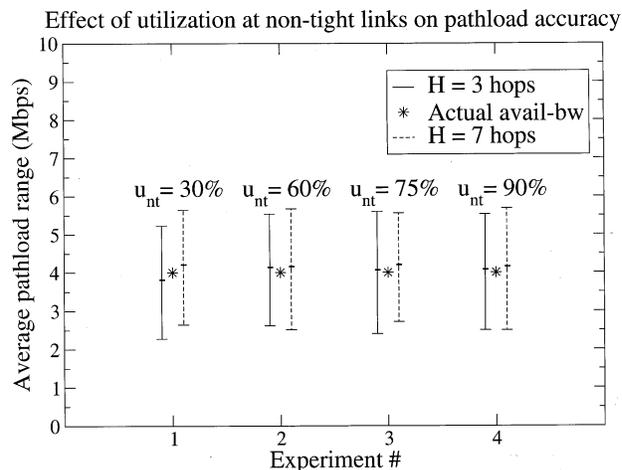
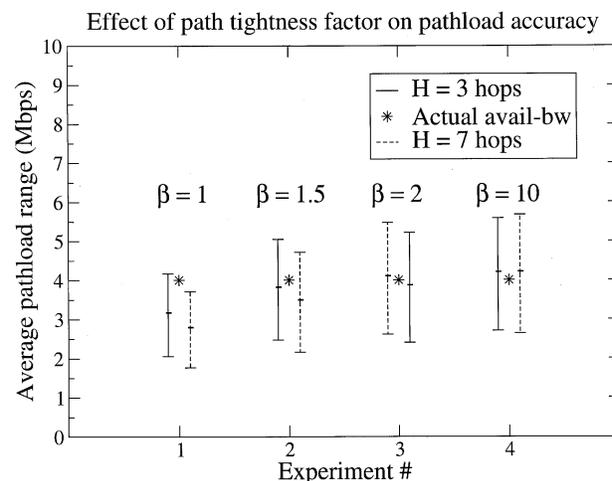


Fig. 6. Simulation results for different nontight link loads.


 Fig. 7. Simulation results for different path tightness factors β .

of *pathload* streams, but they do not affect the OWD trend that is formed when the stream goes through the tight link.

Let us now examine whether the accuracy of *pathload* depends on the path tightness factor β . Fig. 7 shows 50-sample average *pathload* ranges for four different values of β and for two different path lengths H . As previously, $C_t = 10$ Mb/s, $u_t = 60\%$, and so the average avail-bw is $A_t = 4$ Mb/s. Note that when the path tightness factor is $\beta = 1$, all H links have the same avail-bw $A_{nt} = A_t = 4$ Mb/s, meaning that they are all tight links. The main observation in Fig. 7 is that *pathload* succeeds in estimating a range that includes the actual avail-bw when there is only one tight link in the path, but it underestimates the avail-bw when there are multiple tight links.

To understand the nature of this problem, note that an underestimation occurs when R^{\max} is set to a fleet rate R , even though R is less than the avail-bw A_t . Recall that *pathload* sets the state variable R^{\max} to a rate R when more than $f = 70\%$ of a fleet's streams have an increasing delay trend. A stream of rate R , however, can get an increasing delay trend at any link of the path in which the avail-bw during the stream is less than R . Additionally, after a stream gets an increasing delay trend it is unlikely that it will loose that trend later in the path. Consider a path with H_t tight links, all of them having an average avail-bw

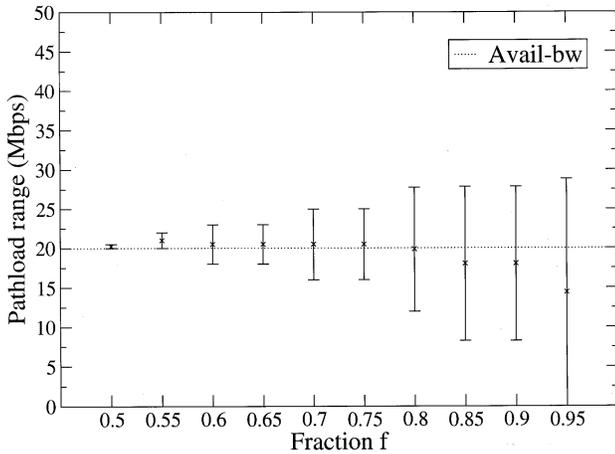


Fig. 8. Simulation results for different values of fraction f .

A_t . Suppose that p is the probability that a stream of rate R will get an increasing delay trend at a tight link, even though $R < A_t$. Assuming that the avail-bw variations at different links are independent, the probability that the stream will have an increasing delay trend after H_t tight links is $1 - (1 - p)^{H_t}$, which increases very quickly with H_t . This explains why the underestimation error in Fig. 7 appears when β is close to one (i.e., $H_t > 1$), and why it is more significant with $H_t = 7$ rather than with three hops.

Finally, we examine the effect of f and of the PCT/PDT thresholds on the *pathload* results.

Fig. 8 shows the effect of f on the *pathload* estimates. The reported *pathload* range, here, is a result of a single *pathload* run. In these simulations, $C_t = 50$ Mb/s, $u_t = u_{nt} = 60\%$, and so the average avail-bw in the path is $A_t = 20$ Mb/s. Note that as f increases, the width of the grey region, and hence the range of the estimated avail-bw, increases as well. The reason is that, for a given R and A , a higher f means that a larger fraction of streams must be of type I when $R > A$ (or of type N when $R < A$) in order to correctly characterize the entire fleet as increasing (or nonincreasing when $R < A$).

Fig. 9 shows the effect of the PDT threshold on the *pathload* estimates. The simulation parameters are as in Fig. 8, but here we use only the PDT metric to detect increasing delay trend. Note that *pathload* underestimates the avail-bw when the PDT threshold is too small (≈ 0), and it overestimates the avail-bw when the PDT threshold is too large (≈ 1). To understand why, recall that a stream is characterized as type I if S_{PDT} is larger than the PDT threshold. A small PDT threshold means that a stream can be marked as type I even if $R < A$. Similarly, with a large PDT threshold, a stream can be marked as type N even if $R > A$. The PCT threshold has a similar effect on the accuracy of *pathload*; we do not include those results due to space constraints.

B. Experimental Results

We have also verified *pathload* experimentally, comparing its output with the avail-bw shown in the MRTG [25] graph of the path's tight link. Even though this verification methodology is not very accurate, it was the only way in which we could evaluate *pathload* in real and wide-area Internet paths. For addi-

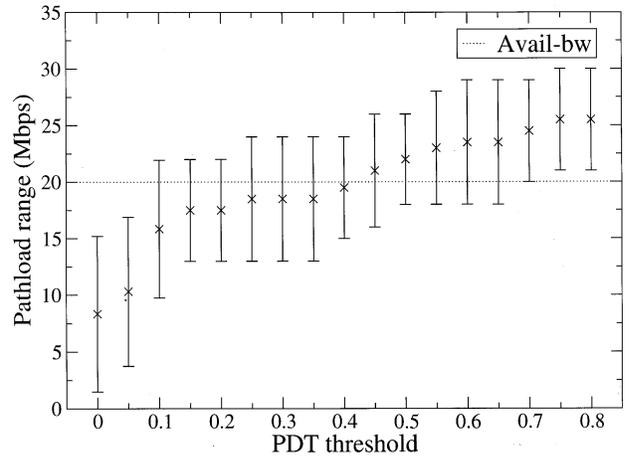


Fig. 9. Simulation results for different values of the PDT threshold.

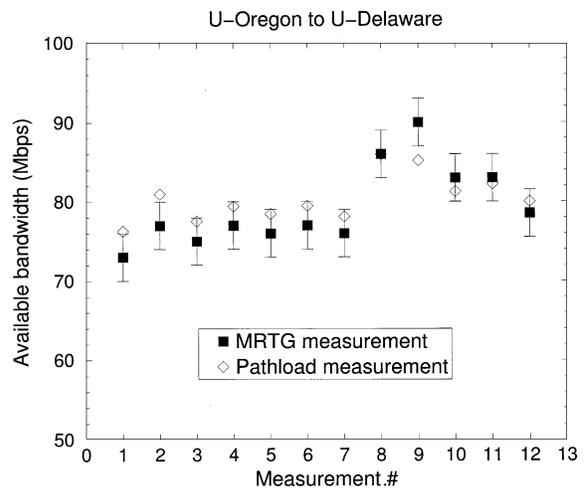


Fig. 10. Verification experiment.

tional experimental results, and information about the use of MRTG in the verification of *pathload*, see [12].

In the experiments of Fig. 10, the resolution parameters were $\omega = 1$ Mb/s and $\chi = 1.5$ Mb/s, while f and the PCT and PDT thresholds were 0.7, 0.6, and 0.5, respectively.

An MRTG reading is a 5-min average avail-bw. *Pathload*, however, takes about 10–30 s to produce an estimate. To compare these short-term *pathload* estimates with the MRTG average, we run *pathload* consecutively for 5 min. Suppose that in a 5-min (300-s) interval we run *pathload* W times, and that run i lasted for q_i seconds, reporting an avail-bw range $[R_i^{\min}, R_i^{\max}]$ ($i = 1, \dots, W$). The 5-min average avail-bw \hat{R} that we report here for *pathload* is the following weighted average of $(R_i^{\min} + R_i^{\max})/2$:

$$\hat{R} = \sum_{i=1}^W \frac{q_i}{300} \frac{R_i^{\min} + R_i^{\max}}{2}. \quad (11)$$

Fig. 10 shows the MRTG and *pathload* results for twelve independent runs in a path from a Univ-Oregon host to a Univ-Delaware host.⁴ An interesting point about this path is that the tight link is different than the narrow link. The former is a

⁴More information about the location of the measurements hosts and the underlying routes is given in [12].

155-Mb/s POS OC-3 link, while the latter is a 100-Mb/s Fast Ethernet interface. The MRTG readings are given as 6-Mb/s ranges, due to the limited resolution of the graphs. Note that the *pathload* estimate falls within the MRTG range in ten out of the twelve runs, while the deviations are marginal in the two other runs.

VI. AVAILABLE BANDWIDTH DYNAMICS

In this section, we use *pathload* to evaluate the variability of the avail-bw in different timescales and operating conditions. Given that our experiments are limited to a few paths, we do not attempt to make quantitative statements about the avail-bw variability in the Internet. Instead, our objective is to show the relative effect of certain operational factors on the variability of the avail-bw.

In the following experiments, $\omega = 1$ Mb/s and $\chi = 1.5$ Mb/s. Note that because $\omega < \chi$, *pathload* terminates due to the ω constraint only if there is no grey region; otherwise, it exits due to the χ constraint. So, the final range $[R^{\min}, R^{\max}]$ that *pathload* reports is either at most 1 Mb/s (ω) wide, indicating that there is no grey region, or it overestimates the width of the grey region by at most 2χ . Thus, $[R^{\min}, R^{\max}]$ is within 2χ (or ω) of the range in which the avail-bw varied during that *pathload* run. To compare the variability of the avail-bw across different operating conditions and paths, we define the following relative variation metric:

$$\rho = \frac{R^{\max} - R^{\min}}{(R^{\max} + R^{\min})/2}. \tag{12}$$

In the following graphs, we plot the $\{5, 15, \dots, 95\}$ percentiles of ρ based on 110 *pathload* runs for each experiment.

A. Variability and Load Conditions

Fig. 11 shows the cumulative distribution function (CDF) of ρ for a path with $C = 12.4$ Mb/s in three different utilization ranges of the tight link: $u = 20\%–30\%$, $40\%–50\%$, and $75\%–85\%$. Notice that the variability of the avail-bw increases significantly as the utilization u of the tight link increases (i.e., as the avail-bw A decreases). This observation is not a surprise. In Markovian queues, say, in $M|M|1$, the variance of the queueing delay is inversely proportional to the square of the avail-bw. The fact that increasing load causes higher variability was also observed for self-similar traffic in [19]. Returning to Fig. 11, the 75 percentile shows that when the avail-bw is around $A = 9$ Mb/s ($u = 20\%–30\%$), three quarters of the measurements have a relative variation $\rho \leq 0.25$. In heavy-load conditions ($u = 75\%–85\%$) on the other hand, when $A = 2–3$ Mb/s, the same fraction of measurements give almost five times higher relative variation ($\rho \leq 1.3$).

We observed a similar trend in all the paths that we experimented with. For users, this suggests that a lightly loaded network will not only provide more avail-bw, but also a more predictable and smooth throughput. This latter attribute is even more important for some applications, such as streaming audio/video.

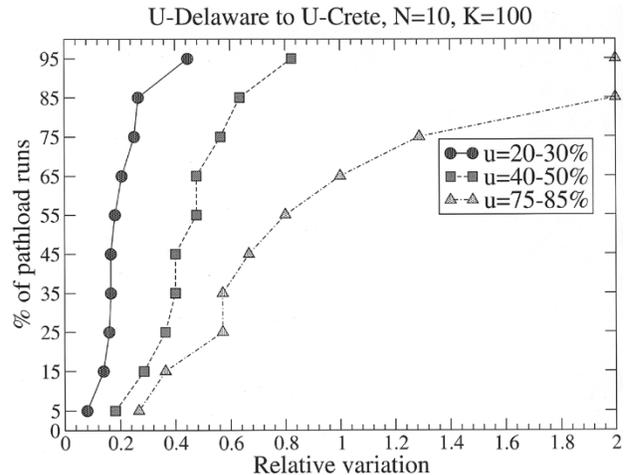


Fig. 11. Variability of avail-bw in different load conditions.

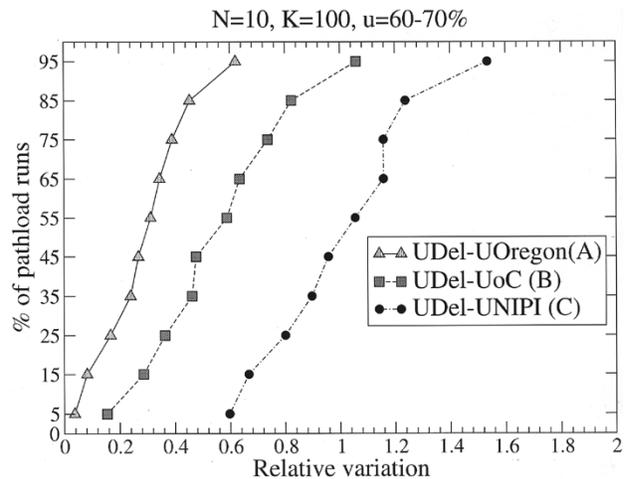


Fig. 12. Variability of avail-bw in different paths.

B. Variability and Statistical Multiplexing

In this experiment, we run *pathload* in three different paths, (A), (B), and (C), when the tight link utilization was roughly the same (around 65%) in all paths. The capacity of the tight link is 155 Mb/s (A), 12.4 Mb/s (B), and 6.1 Mb/s (C). The tight link in (A) connects the Oregon GigaPoP to the Abilene network, the tight link in (B) connects a large university in Greece (Univ-Crete) to a national network (GRnet), while the tight link in (C) connects a smaller university (Univ-Pireaus) to the same national network. Based on these differences, it is reasonable to assume that the degree of statistical multiplexing, i.e., the number of flows that simultaneously use the tight link, is highest in path (A), and higher in (B) than in (C). Fig. 12 shows the CDF of ρ in each path. If our assumption about the number of simultaneous flows in the tight link of these paths is correct, we observe that the variability of the avail-bw decreases significantly as the degree of statistical multiplexing increases. Specifically, looking at the 75 percentile, the relative variation is $\rho \leq 0.4$ in path (A), and it increases by almost a factor of two ($\rho \leq 0.74$) in path (B), and by almost a factor of three ($\rho \leq 1.18$) in path (C). It should be noted, however, that there may be other differences between the three paths that cause the observed variability

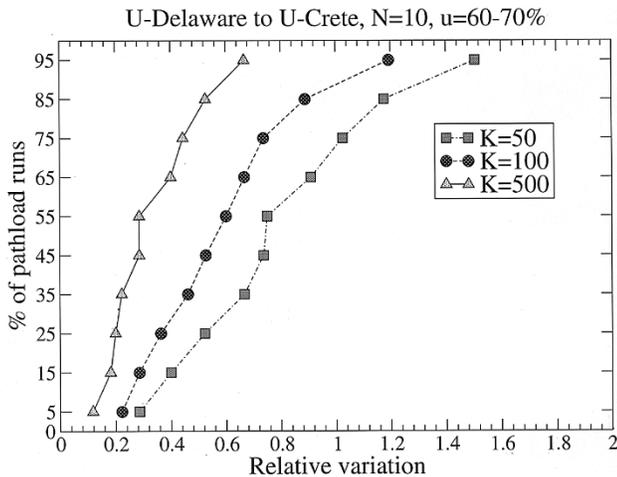


Fig. 13. Effect of K on the variability of the avail-bw.

differences; the degree of statistical multiplexing is simply one plausible explanation.

For users, the previous measurements suggest that if they can choose between two networks that operate at about the same utilization, the network with the wider pipes, and thus with a higher degree of statistical multiplexing, will offer them a more predictable throughput. For network providers, on the other hand, it is better to aggregate traffic in a higher-capacity trunk than to split traffic in multiple parallel links of lower capacity, if they want to reduce the avail-bw variability.

C. Effect of the Stream Length

Since $V = KT$, the stream duration V increases proportionally to the stream length K . With a longer stream, we examine whether $R > A$ over wider timescales. As mentioned in the introduction, however, the variability of the avail-bw A decreases as the averaging timescale increases. So, the variability in the relation between R and A , and thus the variability of the *pathload* measurements, is expected to decrease as the stream duration increases.

Fig. 13 shows the CDF of ρ for three different values of K in a path with $C = 12.4$ Mb/s. During the measurements, A was approximately 4.5 Mb/s. The stream duration V for $R = A$, $L = 200$ B, and $T = 356 \mu\text{s}$ is 18 ms for $K = 50$, 36 ms for $K = 100$, and 180 ms for $K = 500$. The major observation here is that the variability of the avail-bw decreases significantly as the stream duration increases, as expected. Specifically, when $V = 180$ ms, 75% of the measurements produced a range that is less than 2.0 Mb/s wide ($\rho \leq 0.45$). When $V = 18$ ms, on the other hand, the corresponding maximum avail-bw range increases to 4.7 Mb/s ($\rho \leq 1.05$).

D. Effect of the Fleet Length

Suppose that we measure the avail-bw $A^\tau(t)$ in a time interval $(t_0, t_0 + \Theta)$, with a certain averaging timescale τ ($\tau < \Theta$). These measurements will produce a range of avail-bw values, say from a minimum A_{\min}^τ to a maximum A_{\max}^τ . If we keep τ fixed and increase the measurement period Θ , the range $[A_{\min}^\tau, A_{\max}^\tau]$ becomes wider because it tracks the boundaries

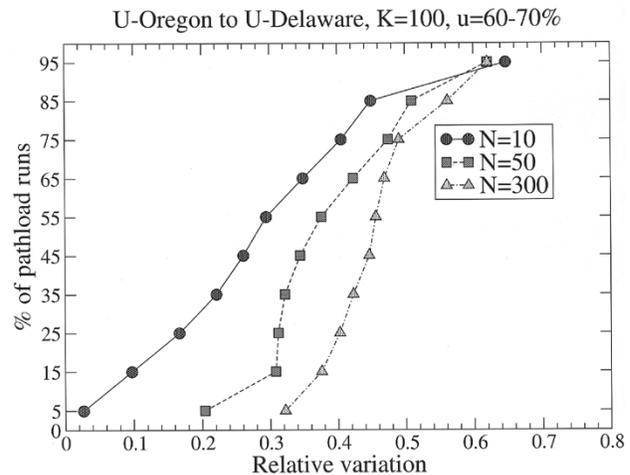


Fig. 14. Effect of N on the variability of the avail-bw.

of the avail-bw process during a longer time period. An additional effect is that, as Θ increases, the variation of the width $A_{\max}^\tau - A_{\min}^\tau$ decreases. The reason is that the boundaries A_{\min}^τ and A_{\max}^τ tend to their expected values (assuming a stationary process), as the duration of the measurement increases.

The measurement period Θ is related to the number of streams N in a fleet, and to the fleet duration $U = N(V + \Delta)$. As we increase N , keeping V fixed, we expand the time window in which we examine the relation between R and A , and thus we increase the likelihood that the rate R will be in the grey region of the avail-bw ($R \bowtie A$). So, the grey region at the end of the *pathload* run will be wider, causing a larger relative variation ρ . This effect is shown in Fig. 14 for three values of N . Observe that as the fleet duration increases, the variability in the measured avail-bw increases. Also, as the fleet duration increases, the variation across different pathload runs decreases, causing a steeper CDF.

VII. TCP AND AVAILABLE BANDWIDTH

We next focus on the relationship between the avail-bw in a path and the throughput of a persistent (or bulk) TCP connection with arbitrarily large advertised window. There are two questions that we attempt to explore. First, can a bulk TCP connection measure the avail-bw in a path, and how accurate is such an avail-bw measurement approach? Second, what happens then to the rest of the traffic in the path, i.e., how intrusive is a TCP-based avail-bw measurement?

It is well-known that the throughput of a TCP connection can be limited by a number of factors, including the receiver's advertised window, total transfer size, RTT, buffer size at the path routers, probability of random losses, and avail-bw in the forward and reverse paths. In the following, we use the term bulk transfer capacity (BTC) connection (in relation to [22]) to indicate a TCP connection that is only limited by the network, and not by end-host constraints.

Let us first describe the results of an experiment that measured the throughput of a BTC connection from Univ-Ioannina (Greece) to Univ-Delaware. The TCP sender was a Sun OS 5.7 box, while the TCP receiver ran FreeBSD 4.3. The tight link in

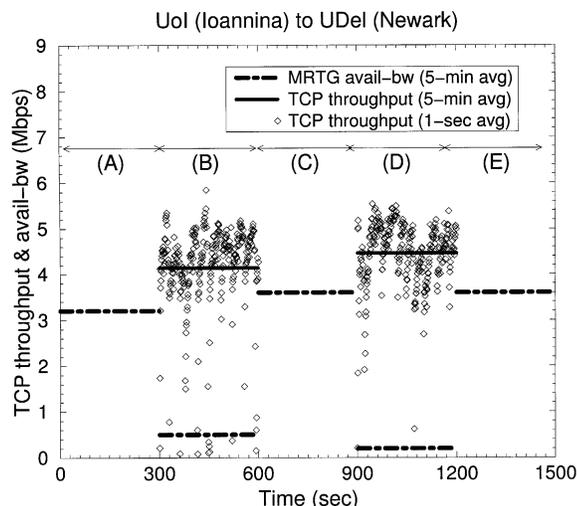


Fig. 15. Available bandwidth and BTC throughput.

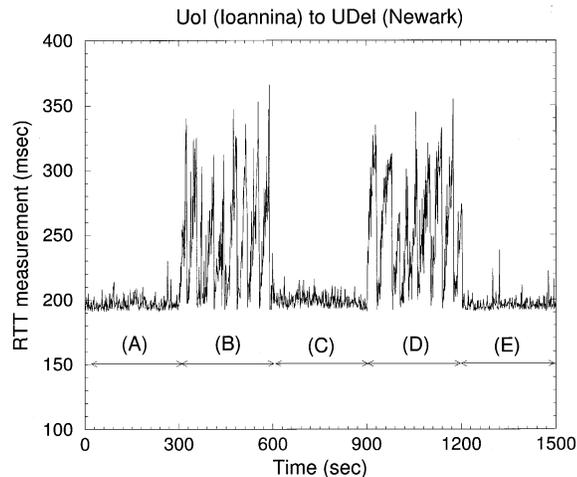


Fig. 16. RTT measurements during the experiment of Fig. 15.

the path had a capacity of 8.2 Mb/s. Consider a 25-min measurement interval, partitioned in five consecutive 5-min intervals (A)–(E). During (B) and (D), we performed a BTC connection and measured its throughput in both 1-s intervals and in the entire 5-min interval. During (A), (C), and (E), we did not perform a BTC connection. Throughout the 25-min interval, we also monitored the avail-bw in the path using MRTG data for each of the 5-min intervals. In parallel, we used *ping* to measure the RTT in the path at every second. Fig. 15 shows the throughput of the two BTC connections, as well as the 5-min average avail-bw in the path, while Fig. 16 shows the corresponding RTT measurements.

We make three important observations from these figures. First, the avail-bw during (B) and (D) is less than 0.5 Mb/s, and so for most practical purposes, the BTC connection manages to saturate the path. Also note, however, that the BTC throughput shows high variability when measured in 1-s intervals, often being as low as a few hundreds of kilobits per second. Consequently, even though a bulk TCP connection that lasts for several minutes should be able to saturate a path when not limited by

the end hosts,⁵ shorter TCP connections should expect a significant variability in their throughput.

Second, there is a significant increase in the RTT measurements during (B) and (D), from a “quiescent point” of 200 ms to a high variability range between 200 and 370 ms. The increased RTT measurements can be explained as follows: the BTC connection increases its congestion window until a loss occurs. A loss, however, does not occur until the queue of the tight link overflows.⁶ Thus, the queue size at the tight link increases significantly during the BTC connection, causing the large RTT measurements shown. To quantify the queue size increase, note that the maximum RTT’s climb up to 370 ms, or 170 ms more than their quiescent point. The tight link has a capacity of 8.2 Mb/s, and so its queue size becomes occasionally at least 170 kB larger during the BTC connection. The RTT jitter is also significantly higher during (B) and (D), as the queue size of the tight link varies between high and low occupancy due to the “sawtooth” behavior of the BTC congestion window.

Third, the BTC connection gets an average throughput during (B) and (D), that is about 20%–30% more than the avail-bw in the surrounding intervals (A), (C), and (E). This indicates that a BTC connection can get more bandwidth than what was previously available in the path, grabbing part of the throughput of other TCP connections. To see how this happens, note that the presence of the BTC connection during (B) and (D) increases the RTT of all other TCP connections that go through the tight link, because of a longer queue at that link. Additionally, the BTC connection causes buffer overflows and, thus, potential losses to other TCP flows.⁷ The increased RTTs and losses reduce the throughput of other TCP flows, allowing the BTC connection to get a larger share of the tight link than what was previously available.

To summarize, a BTC connection measures more than the avail-bw in the path, because it shares some of the previously utilized bandwidth of other TCP connections, and it causes significant increases in the delays and jitter at the tight link of its path. This latter issue is crucial for real-time and streaming applications that may be active during the BTC connection.

VIII. IS PATHLOAD INTRUSIVE?

An important question is whether *pathload* has an intrusive behavior, i.e., whether it causes significant decreases in the avail-bw, and increased delays or losses.

Figs. 17 and 18 show the results of a 25-min experiment, performed similarly as the experiment of Section VII. Specifically, *pathload* runs during the 5-min intervals (B) and (D). We monitor the 5-min average avail-bw in the path using MRTG (Fig. 17), and also perform RTT measurements in every 100 ms (Fig. 18). The RTT measurement period here is ten times smaller than in Section VII, because we want to examine whether *pathload* causes increased delays or losses even in smaller timescales than one second.

⁵This will not be the case, however, in underbuffered links, or paths with random losses [18].

⁶Assuming drop-tail queueing, which is the common practice today.

⁷We did not observe, however, losses of *ping* packets.

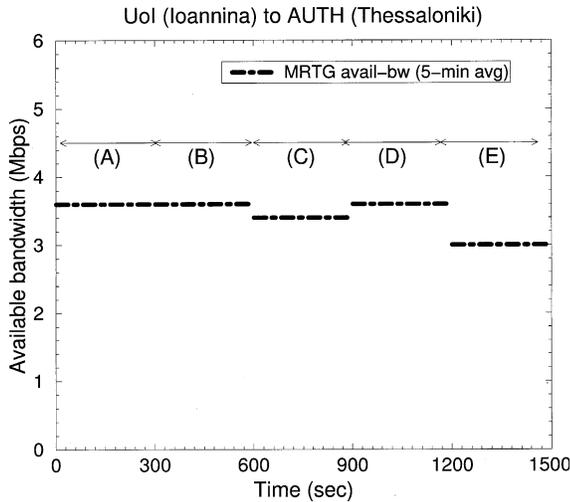


Fig. 17. Available bandwidth measurements.

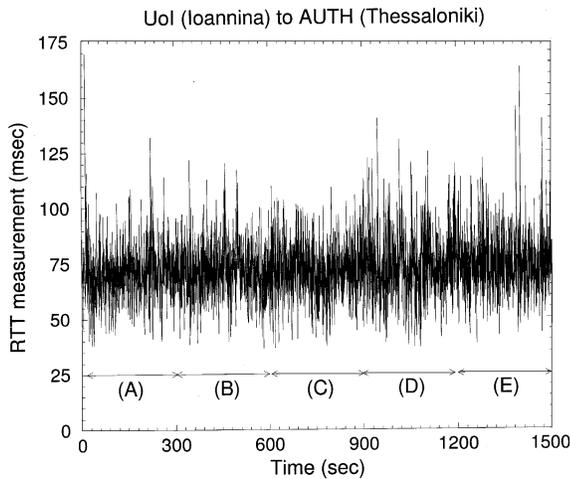


Fig. 18. RTT measurements during the experiment of Fig. 17.

The results of the experiment are summarized as follows. First, the avail-bw during (B) and (D) does not show a measurable decrease compared to the intervals (A), (C), and (E). Second, the RTT measurements do not show any measurable increase when *pathload* runs. So, *pathload* does not seem to cause a persistent queue size increase, despite the fact that it often sends streams of higher rate than the avail-bw. The reason is that each stream is only $K = 100$ packets, and a stream is never sent before the previous has been acknowledged. We also note that none of the *pathload* streams encountered any losses in this experiment. None of the *ping* packets was lost, either.

IX. CONCLUSION

We described an original end-to-end available bandwidth measurement methodology, called SLoPS. The key idea in SLoPS is that the one-way delays of a periodic stream show increasing trend if the stream rate is greater than the avail-bw. Such an end-to-end avail-bw measurement methodology can have numerous applications, such as tuning TCP's *ssthresh* parameter, overlay networks and end-system multicast, rate adaptation in streaming applications, end-to-end admission

control, server selection and anycasting, and verification of service level agreements (SLAs). We have implemented SLoPS in a tool called *pathload*, and showed through simulations and Internet experiments that *pathload* is nonintrusive and that it measures avail-bw accurately under various load conditions and typical path configurations. We finally examined the variability of avail-bw in different paths and load conditions, as well as the relationship between TCP throughput and avail-bw.

APPENDIX PROOF OF PROPOSITION 1

At the First Link

Case 1: $R_0 > A_1$: Suppose that t^k is the arrival time of packet k in the queue. Over the interval $[t^k, t^k + T)$, with $T = L/R_0$, the link is constantly backlogged because the arriving rate is higher than the capacity ($R_0 + u_1 C_1 = C_1 + (R_0 - A_1) > C_1$). Over the same interval, the link receives $L + u_1 C_1 T$ bytes and services $C_1 T$ bytes. Thus

$$\Delta d_1^k = (L + u_1 C_1 T) - C_1 T = (R_0 - A_1) T > 0 \quad (13)$$

and so

$$\Delta d_1^k = \frac{R_0 - A_1}{C_1} T > 0. \quad (14)$$

Packet $k + 1$ departs the first link Λ time units after packet k , where

$$\Lambda = (t^{k+1} + d_1^{k+1}) - (t^k + d_1^k) = T + \frac{R_0 - A_1}{C_1} T \quad (15)$$

that is independent of k . So, the packets of the stream depart the first link with a constant rate R_1 , where

$$R_1 = \frac{L}{\Lambda} = R_0 \frac{C_1}{C_1 + (R_0 - A_1)}. \quad (16)$$

We refer to rate R_{i-1} as the *entry-rate* in link i , and to R_i as the *exit-rate* from link i . Given that $R_0 > A_1$ and that $C_1 \geq A_1$, it is easy to show that the exit-rate from link 1 is larger or equal than A_1 ⁸ and lower than the entry-rate

$$A_1 \leq R_1 < R_0. \quad (17)$$

Case 2: $R_0 \leq A_1$: In this case, the arrival rate at the link in interval $[t^k, t^k + T)$ is $R_0 + u_1 C_1 \leq C_1$. So, packet k is serviced before packet $k + 1$ arrives at the queue. Thus

$$\Delta d_1^k = 0 \quad \Delta d_1^k = 0 \quad R_1 = R_0. \quad (18)$$

Induction to Subsequent Links: The results that were previously derived for the first link can be proved inductively for each link in the path. So, we have the following relationship between the entry and exit rates of link i :

$$R_i = \begin{cases} R_{i-1} \frac{C_i}{C_i + (R_{i-1} - A_i)}, & \text{if } R_{i-1} > A_i \\ R_{i-1}, & \text{otherwise} \end{cases} \quad (19)$$

with

$$A_i \leq R_i < R_{i-1}, \quad \text{when } R_{i-1} > A_i. \quad (20)$$

Consequently, the exit-rate from link i is

$$R_i \geq \min\{R_{i-1}, A_i\}. \quad (21)$$

⁸ $R_1 = A_1$ when $A_1 = C_1$.

Also, the queueing delay difference between successive packets at link i is

$$\Delta d_i^k = \begin{cases} \frac{R_{i-1} - A_i}{C_i} T > 0, & \text{if } R_{i-1} > A_i \\ 0, & \text{otherwise.} \end{cases} \quad (22)$$

OWD Variations: If $R_0 > A$, we can apply (20) recursively for $i = 1, \dots, (t-1)$ to show that the stream will arrive at the tight link with a rate $R_{t-1} \geq A_{t-1} > A_t$. Thus, based on (22), $\Delta d_t^k > 0$, and so the OWD difference between successive packets will be positive, $\Delta d^k > 0$.

On the other hand, if $R_0 \leq A$, then $R_0 \leq A_i$ for every link i (from the definition of A). So, applying (21) recursively from the first link to the last, we see that $R_i < A_i$ for $i = 1, \dots, H$. Thus, (22) shows that the delay difference in each link i is $\Delta d_i^k = 0$, and so the OWD differences are $\Delta d^k = 0$.

ACKNOWLEDGMENT

The authors would like to thank the following colleagues for providing them with computer accounts at their sites: C. Douligeris (Univ-Pireaus), L. Georgiadis (AUTH), E. Markatos (Univ-Crete), D. Meyer (Univ-Oregon), M. Paterakis (Technical Univ-Crete), I. Stavarakis (Univ-Athens), and L. Tassioulas (Univ-Ioannina). They would also like to thank the anonymous referees for their many constructive comments, and M. Seaver for her suggestions on the terminology.

REFERENCES

- [1] M. Allman, "Measuring end-to-end bulk transfer capacity," in *Proc. ACM SIGCOMM Internet Measurement Workshop*, Nov. 2001, pp. 139–143.
- [2] M. Allman and V. Paxson, "On estimating end-to-end network path properties," in *Proc. ACM SIGCOMM*, Sept. 1999, pp. 263–274.
- [3] A. A. Awadallah and C. Rai, "TCP-BFA: Buffer Fill Avoidance," in *Proc. IFIP High Performance Networking Conf.*, Sept. 1998, pp. 575–594.
- [4] H. Balakrishnan, S. Seshan, M. Stemm, and R. H. Katz, "Analyzing stability in wide-area network performance," in *Proc. ACM SIGMETRICS*, June 1997, pp. 2–12.
- [5] L. S. Brakmo, S. W. O'Malley, and L. L. Peterson, "TCP Vegas: New techniques for congestion detection and avoidance," in *Proc. ACM SIGCOMM*, Aug. 1994, pp. 24–35.
- [6] R. L. Carter and M. E. Crovella, "Measuring bottleneck link speed in packet-switched networks," *Perform. Eval.*, vol. 27, no. 28, pp. 297–318, 1996.
- [7] C. Dovrolis, P. Ramanathan, and D. Moore, "What do packet dispersion techniques measure?," in *Proc. IEEE INFOCOM*, Apr. 2001, pp. 905–914.
- [8] A. B. Downey, "Using Pathchar to estimate Internet link characteristics," in *Proc. ACM SIGCOMM*, Sept. 1999, pp. 222–223.
- [9] V. Jacobson, "Congestion avoidance and control," in *Proc. ACM SIGCOMM*, Sept. 1988, pp. 314–329.
- [10] —, (1997, Apr.) Pathchar: A tool to infer characteristics of Internet paths. [Online]. Available: ftp://ftp.ee.lbl.gov/pathchar/
- [11] M. Jain and C. Dovrolis, "End-to-end available bandwidth: Measurement methodology, dynamics, and relation with TCP throughput," in *Proc. ACM SIGCOMM*, Aug. 2002, pp. 295–308.
- [12] —, "Pathload: A measurement tool for end-to-end available bandwidth," in *Proc. Passive and Active Measurements (PAM) Workshop*, Mar. 2002, pp. 14–25.
- [13] R. Jain, "A delay-based approach for congestion avoidance in interconnected heterogeneous computer networks," *ACM Comput. Commun. Rev.*, vol. 19, no. 5, pp. 56–71, Oct. 1989.
- [14] G. Jin, G. Yang, B. Crowley, and D. Agarwal, "Network Characterization Service (NCS)," in *Proc. 10th IEEE Symp. High Performance Distributed Computing*, Aug. 2001, pp. 289–299.

- [15] S. Keshav, "A control-theoretic approach to flow control," in *Proc. ACM SIGCOMM*, Sept. 1991, pp. 3–15.
- [16] K. Lai and M. Baker, "Measuring bandwidth," in *Proc. IEEE INFOCOM*, Apr. 1999, pp. 235–245.
- [17] —, "Measuring link bandwidths using a deterministic model of packet delay," in *Proc. ACM SIGCOMM*, Sept. 2000, pp. 283–294.
- [18] T. V. Lakshman and U. Madhow, "The performance of TCP/IP for networks with high bandwidth delay products and random losses," *IEEE/ACM Trans. Networking*, vol. 5, pp. 336–350, June 1997.
- [19] W. E. Leland, M. S. Taqqu, W. Willinger, and D. V. Wilson, "On the self-similar nature of ethernet traffic (Extended Version)," *IEEE/ACM Trans. Networking*, vol. 2, pp. 1–15, Feb. 1994.
- [20] B. A. Mah. (1999, Feb.) Pchar: A tool for measuring Internet path characteristics. [Online]. Available: http://www.employees.org/~bmah/Software/pchar/
- [21] M. Mathis. (1999, Feb.) TReno bulk transfer capacity. IETF, Internet Draft. [Online]. Available: http://www.advanced.org/IPPM/docs/draft-ietf-ippm-treno-btc-03.txt
- [22] M. Mathis and M. Allman. (2001, July) A framework for defining empirical bulk transfer capacity metrics. IETF, RFC 3148. [Online]. Available: http://www1.ietf.org/rfc/rfc3148.txt
- [23] B. Melander, M. Bjorkman, and P. Gunningberg, "A new end-to-end probing and analysis method for estimating bandwidth bottlenecks," in *Proc. IEEE Globecom*, 2000, pp. 415–420.
- [24] D. Mitra and J. B. Seery, "Dynamic adaptive windows for high-speed data networks: Theory and simulations," in *Proc. ACM SIGCOMM*, Aug. 1990, pp. 30–40.
- [25] T. Oetiker. MRTG: Multi Router Traffic Grapher. Swiss Federal Inst. Technology, Zurich. [Online]. Available: http://ee-staff.ethz.ch/~oetiker/webtools/mrtg/mrtg.html
- [26] A. Pasztor and D. Veitch, "The packet size dependence of packet pair like methods," in *Proc. IEEE/IFIP Int. Workshop Quality of Service (IWQoS)*, 2002, pp. 204–213.
- [27] V. Paxson, "On calibrating measurements of packet transit times," in *Proc. ACM SIGMETRICS*, June 1998, pp. 11–21.
- [28] —, "End-to-end Internet packet dynamics," *IEEE/ACM Trans. Networking*, vol. 7, pp. 277–292, June 1999.
- [29] V. Ribeiro, M. Coates, R. Riedi, S. Sarvotham, B. Hendricks, and R. Baraniuk, "Multifractal cross-traffic estimation," in *Proc. ITC Specialist Seminar IP Traffic Measurement, Modeling, and Management*, Sept. 2000.
- [30] Y. Zhang, N. Duffield, V. Paxson, and S. Shenker, "On the constancy of Internet path properties," in *Proc. ACM SIGCOMM Internet Measurement Workshop*, Nov. 2001, pp. 197–211.



vices.

Manish Jain received the B.Eng. degree in computer engineering from the Shri Govindram Sekseria Institute of Technology and Science (SGSITS), Indore, India, in 1999 and the M.S. degree in computer science from the University of Delaware, Newark, in 2002. He is currently working toward the Ph.D. degree at the College of Computing, Georgia Institute of Technology, Atlanta.

His research interests include bandwidth estimation methodologies and applications, TCP in high bandwidth networks, and network applications and ser-



tools, service differentiation, and router architectures.

Constantinos Dovrolis (M'93) received the computer engineering degree from the Technical University of Crete, Crete, Greece, in 1995, the M.S. degree from the University of Rochester, Rochester, NY, in 1996, and the Ph.D. degree from the University of Wisconsin–Madison in 2000.

He is currently an Assistant Professor with the College of Computing, Georgia Institute of Technology, Atlanta. His research interests include methodologies and applications of network measurements, bandwidth estimation algorithms and