Pathload: a measurement tool for end-to-end available bandwidth

Manish Jain and Constantinos Dovrolis
Computer and Information Sciences
University of Delaware
{jain, dovrolis}@cis.udel.edu

Abstract

The available bandwidth of a network path $\mathcal{P}$ is the maximum throughput that $\mathcal{P}$ can provide to a flow, without reducing the throughput of the cross traffic in $\mathcal{P}$. We have developed an end-to-end active measurement tool, called *pathload*, that estimates the available bandwidth of a network path. The basic idea in *pathload* is that the one-way delays of a periodic packet stream show increasing trend, when the stream rate is larger than the available bandwidth. In this paper, we describe *pathload* in detail, and show some experimental results that illustrate the tool's accuracy.

I. Introduction

Consider two network hosts $\mathcal{SND}$ and $\mathcal{RCV}$. The path $\mathcal{P}$ is the sequence of store-and-forward links that transfer packets from $\mathcal{SND}$ to $\mathcal{RCV}$. We assume that $\mathcal{P}$ is fixed and unique, i.e., no routing changes or multipath forwarding occur during the measurement process. Two throughput-related metrics that are commonly associated with $\mathcal{P}$ are the *capacity* C and the *available bandwidth* (or *avail-bw* for short) A.

If H is the number of hops (links) in $\mathcal{P}$, and C_i is the transmission rate or *capacity* of link i, then the capacity of the path is

$$C = \min_{i=1\ldots H} C_i \tag{1}$$

So, *the end-to-end capacity is defined as the maximum rate that the path can provide to a flow, when there is no other traffic in* $\mathcal{P}$. Suppose, now, that link i transmitted $C_i u_i$ bits during a time interval $\mathcal{T}$. The term u_i is the *utilization* of link i during $\mathcal{T}$, with $0 \leq u_i \leq 1$. Intuitively, the avail-bw A_i of link i in $\mathcal{T}$ can be defined as the fraction of the link's capacity that has not been utilized during that interval, i.e.,

$$A_i \equiv C_i\,(1-u_i) \tag{2}$$

Extending this concept to the entire path, the end-to-end avail-bw A during $\mathcal{T}$ is the minimum avail-bw among all links in $\mathcal{P}$,

$$A \equiv \min_{i=1\ldots H} \{C_i\,(1-u_i)\} \tag{3}$$

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 cross traffic in* $\mathcal{P}$.

The link with the minimum transmission rate determines the capacity of the path, while the link with the minimum non-utilized capacity limits the avail-bw. To avoid the term *bottleneck link*, which has been widely used for both metrics, we refer to the capacity limiting link as the *narrow link*, and to the avail-bw limiting link as the *tight link*.

We have developed an active measurement tool, called *pathload*, that estimates the end-to-end avail-bw A. The basic idea in *pathload* is that the one-way delays of a periodic packet stream show increasing trend when the stream rate is larger than the avail-bw. The measurement algorithm is iterative and it requires the cooperation of both $\mathcal{SND}$ and $\mathcal{RCV}$. The tool has been verified experimentally, by comparing its results with SNMP utilization data from the path routers.

In this paper, we first briefly explain the measurement methodology, called *Self-Loading Periodic Streams* (SLoPS), that *pathload* is based on. Then, we give a detailed description of *pathload.*

This work was supported by the SciDAC program of the US Department of Energy (award # DE-FC02-01ER25467).

Finally, we show experimental results that verify the tool and illustrate its accuracy. An extended version of this paper presents a mathematical model for SLoPS, examines the variability of avail-bw in different timescales and operating conditions, and illustrates that *pathload* does not add significant load to the network [1].

II. Related work

There are several bandwidth estimation tools, but most of them focus on capacity, rather than avail-bw. Specifically, *pathchar* [2], *clink* [3], *pchar* [4], and the *tailgating* technique of [5] measure *per-hop capacity*. Also, *bprobe* [6], *nettimer* [7], *pathrate* [8], and the *PBM* methodology [9] measure *end-to-end capacity*.

Another related throughput metric is the *Bulk-Transfer-Capacity (BTC)* [10]. The BTC of a path in a certain time period is the throughput of a bulk TCP transfer, when the transfer is only limited by the network resources and not by limitations at the end-systems. The BTC can be measured with *Treno* or *cap* [11]. 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 depends on TCP's congestion control, and it is the maximum throughput that a *single and persistent* TCP connection can get. The relation between BTC and avail-bw is investigated in [1].

The first tool that attempted to measure avail-bw was *cprobe* [6]. *cprobe* estimated the avail-bw based on the dispersion of long packet trains at the receiver. A similar approach was taken in *pipechar* [12]. The underlying assumption in these works is that the dispersion of long packet trains is inversely proportional to the avail-bw A. Recently, however, [8] 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 that is referred to as *Asymptotic Dispersion Rate* (ADR).

A different avail-bw measurement technique, called *Delphi*, was proposed in [13]. Delphi assumes that the path can be well modeled by a single queue, and so it is not applicable when the tight and narrow links are different, or when there are significant queueing delays in several links of the path. Another technique (called *TOPP*) for measuring avail-bw was studied in [14]. TOPP provides wrong estimates when the path includes several queueing points, and its results depend on the order of the links in the path.

III. Basic idea

Suppose that $\mathcal{SND}$ transmits a *periodic packet stream* to $\mathcal{RCV}$. The stream consists of K packets, where K is the *length* of the stream. The *size* of each packet is L bits, while the *packet transmission period* is T seconds. The *transmission rate* of the stream is $R = L/T$ bits per second.

$\mathcal{SND}$ timestamps each packet i prior to its transmission with a timestamp t_i. Let a_i be the arrival time of the i'th packet at $\mathcal{RCV}$. $\mathcal{RCV}$ computes the *relative One-Way Delay (OWD)* of each packet as $D_i = a_i - t_i$. Note that D_i is the (absolute) one-way delay from $\mathcal{SND}$ to $\mathcal{RCV}$ plus/minus a certain offset Θ, where Θ is the clock offset between the two end-hosts. As will become clear next, the measurement methodology does not need synchronized clocks, because we are only interested in the relative magnitude of OWDs. Upon the receipt of the entire stream, $\mathcal{RCV}$ inspects the sequence of relative OWDs to check whether the transmission rate R is larger than the avail-bw A. The way we examine the relation between R and A is the key idea in the measurement methodology, and it is described next.

When the stream rate R is larger than the avail-bw A, the stream creates a short-term overload in the tight link of the path. During that overload period, the tight link receives more traffic than what it can transmit, and so the queue of the tight link gradually builds up. So, the queueing delay of packet i at the tight link is expected to be larger than the corresponding queueing delay of packet j with $j < i$. Consequently, *when $R > A$, the relative OWDs $\{D_1, D_2, \ldots, D_K\}$ of the stream packets are expected to have an increasing trend.* We refer to this effect as *self-loading of the periodic stream.* On the other hand, if the stream rate R is less than the avail-bw A, the stream will not cause an overload at the tight link, and thus the backlog of that link will not keep increasing with every new stream packet. So, *when $R < A$, the relative OWDs*

$\{D_1, D_2, \ldots, D_K\}$ *of the stream packets are expected to have a non-increasing trend.* A more precise statement and proof of the previous properties, for a fluid model of the cross traffic in the path, is given in [1].

$\mathcal{RCV}$ can infer whether the stream rate R is larger than the avail-bw A based on the self-loading effect of periodic streams. However, to actually estimate the avail-bw in the path, the two end-points have to cooperate so that the stream rate R converges *iteratively* to A. In the n'th step of this iterative procedure, $\mathcal{RCV}$ checks whether the transmission rate $R(n)$ of the n'th stream is larger than A, based on the presence of an increasing trend in the OWDs of stream n. If $R(n) > A$, $\mathcal{SND}$ sends an additional periodic stream with rate $R(n+1) < R(n)$. If $R(n) < A$, the rate of the next periodic stream is $R(n+1) > R(n)$.

The previous paragraphs described the basic idea of our measurement methodology, called *Self-Loading Periodic Streams (SLoPS)*. A complete estimation algorithm, however, has to address several hard issues. One of them is that the avail-bw may vary around R during the stream, making it difficult to judge whether the OWDs show increasing trend or not. To gain some insight into this issue, Figures 1-3 show the OWD variations of three periodic streams that crossed a 12-hop path from Univ-Delaware to Univ-Oregon. All three streams have K=100 packets and T=100μs. The 5-minute average avail-bw in the path during these measurements was about 74Mbps, according to MRTG data from the path routers (see Appendix). In Figure 1, the stream rate is R=96Mbps, i.e., higher than the long-term avail-bw. Notice that *overall, the OWDs of this stream have a clearly increasing trend.* On the other hand, the stream of Figure 2 has a rate R=37Mbps, 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 this stream.* The third stream, in Figure 3, has a rate R=82Mbps. 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 60-th packet. In that second half of the stream there is a clear increasing trend, showing that the avail-bw decreases to less than R. This last stream illustrates that the avail-bw may vary around rate R during a stream. When this is the case, there is no strict ordering between R and A. We refer to this third possibility as 'grey-region', and denote it by $R \bowtie A$.

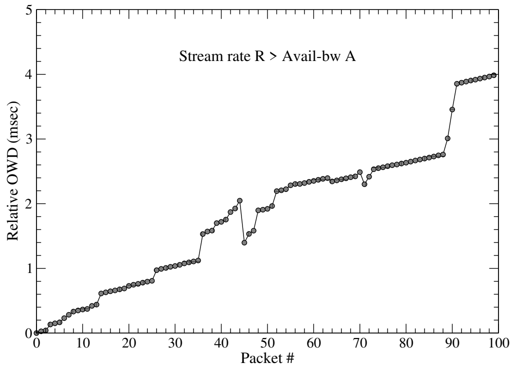


Fig. 1. OWD variations when $R > A$.

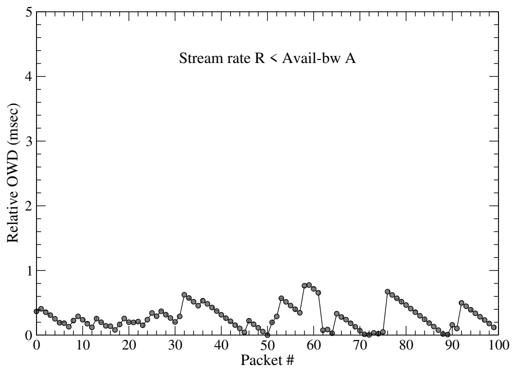


Fig. 2. OWD variations when $R < A$.

IV. Description of pathload

Pathload consists of two components: process $\mathcal{SND}$ running at the sender and process $\mathcal{RCV}$ running at the receiver. The tool uses UDP for the periodic packet streams. Additionally, a TCP connection between the two end-points serves as a 'control channel'. The control channel transfers messages regarding the characteristics of each stream, the abortion or end of the measurement process, etc. In the following, we describe *pathload* in detail.

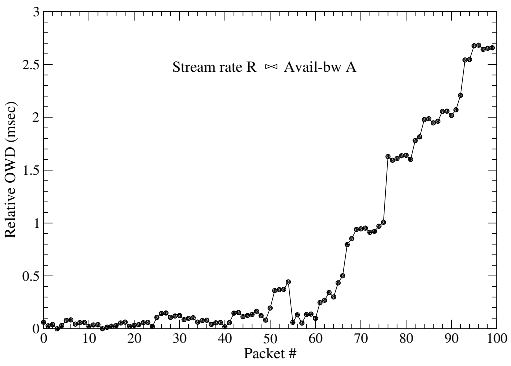


Fig. 3. OWD variations when $R \bowtie A$.

A. Selection of T and L

The transmission period T and the packet size L are two important parameters in *pathload*. First, the transmission rate R of a stream is

$$R = \frac{L}{T} \tag{4}$$

Given the stream rate R, *pathload* selects values for L and T to satisfy (4).

There are some practical constraints in the selection of L and T, however. Specifically, L cannot be less than a certain number of bytes, and it should not be more than the path's MTU (to avoid fragmentation). Also, if L is too small, the possibility of zero-padding in certain layer-2 links may cause a significant change in the layer-2 packet size, and thus in the stream rate, at those links. In the current version of *pathload*, we make sure that L is at least 96 bytes, lower than the path's MTU, and a multiple of 48 bytes (to avoid zero-padding at AAL5-ATM links).

On the other hand, the transmission period T should be as small as possible. The reason is that as T increases, so does the duration of each stream. Ideally, the transmission of each stream should complete before the *pathload* processes $\mathcal{SND}$ or $\mathcal{RCV}$ get interrupted by a context switch at the end-hosts. Additionally, a lower value of T leads to a shorter duration for the entire measurement process. The minimum possible value of T depends on the hardware and operating system of the measurement hosts. Using commodity workstations, we found out that the minimum transmission period T_{min} for back-to-back minimum-sized packets is 15-30μs. Being slightly conservative, we set T=100μs.

In summary, given a target stream rate R, *pathload* chooses the minimum possible period T=100μs, and then the corresponding value of L from (4). If the resulting packet size is less than the minimum allowed value (96 bytes), the packet size L is set to that minimum value and then T is computed from (4).

B. Selection of stream length K

There is a trade-off in the selection of the number of packets K in a stream. First, if K is too large, the stream may overflow the queue of the tight link when $R > A$, causing losses in both the stream and the cross traffic packets. Such losses can cause the cross traffic to back off, leading to a reduction of the avail-bw. On the other hand, if K is too small, the stream will not provide $\mathcal{RCV}$ with enough samples to infer in a robust manner whether there is an increasing trend in the measured OWDs. *Pathrate* uses K=100 packets, because this stream length rarely causes packet losses, while it provides an adequate number of OWD measurements to detect an increasing trend.

C. A fleet of streams

Pathload does not determine whether $R > A$ based on a single stream. Instead, it sends a fleet of N streams. Each stream consists of K packets of size L bits, transmitted periodically in every T seconds. All streams in a fleet have the same rate $R = L/T$. Each stream is sent only when the previous stream has been acknowledged. This introduces an idle interval of one round-trip time Δ between streams. The objective of this idle period is to let the path 'drain' the last stream before sending a next one.

There are two main reasons that we use N streams of K packets each, instead of a single fleet of $N \times K$ packets. First, having N streams allows us to examine N consecutive times whether $R > A$ or not. This is because $\mathcal{RCV}$ checks the measured OWDs for an increasing trend independently in each stream. Second, the use of multiple streams, separated by a 'silence' period Δ, allows the queues in the network to drain our measurement traffic and recover from the short-term overload that each stream causes. In *pathload*, the default value for N is 12 streams.

N, K and T determine the duration U of a fleet, where

$$U = N \times (K \times T + \Delta) \tag{5}$$

The effect of the stream duration $V{=}K \times T$ and of the fleet duration U on the variability of avail-bw measurements is investigated in [1].

D. Detection of increasing trend

A critical algorithm in *pathload* is the detection of increasing trend in the OWDs of a stream. As a pre-processing step, we partition the K OWD measurements $\{D_1, D_2, \ldots, D_K\}$ in $\Gamma = \sqrt{K}$ groups of Γ consecutive OWD measurements. Then, out of the Γ delays in each group i, we compute the median OWD $\hat{D}_i$ of the group. This 'partition-and-medianize' technique makes the following algorithms more robust, as we remove OWD outliers from the estimation process.

We use two statistics to check if a stream shows increasing trend. The *Pairwise Comparison Test* (PCT) metric of a stream is

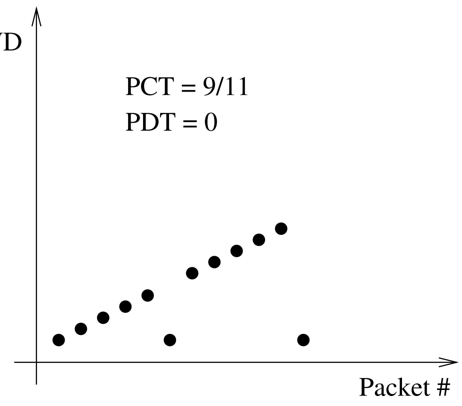
$$S_{PCT} = \frac{\sum_{k=2}^{\Gamma} I(\hat{D}_k > \hat{D}_{k-1})}{\Gamma - 1} \tag{6}$$

where $I(X)$ is one if X holds, and zero otherwise. PCT measures the fraction of consecutive OWD pairs that are increasing, and so $0 \leq S_{PCT} \leq 1$. 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. In *pathload*, the PCT metric reports 'increasing trend' if $S_{PCT} > 0.66$, 'non-increasing trend' if $S_{PCT} < 0.54$, and 'ambiguous trend' otherwise.

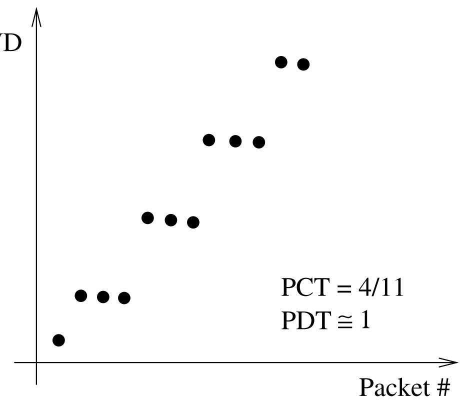
The *Pairwise Difference Test* (PDT) metric of a stream is

$$S_{PDT} = \frac{\hat{D}_{\Gamma} - \hat{D}_1}{\sum_{k=2}^{\Gamma} |\hat{D}_k - \hat{D}_{k-1}|} \tag{7}$$

PDT quantifies how strong is the start-to-end OWD variation, relative to the OWD absolute variations during the stream. Note that $-1 \leq S_{PDT} \leq 1$. If the OWDs are independent, the expected value of S_{PDT} is zero. If there is a strong increasing trend, S_{PDT} approaches one. In *pathload*, the PDT metric reports 'increasing trend' if $S_{PDT} > 0.55$, 'non-increasing trend' if $S_{PDT} < 0.45$, and 'ambiguous trend' otherwise.



(a) High PCT - Low PDT



(b) Low PCT - High PDT

Fig. 4. Comparison of PCT and PDT for a stream with increasing OWD trend.

There are cases in which one of the two metrics is better than the other in detecting an increasing trend. For instance, Figure 4 shows two 12-packet streams with an increasing OWD trend. In Figure 4-a, PCT would detect the increasing trend but PDT would not, while the opposite would be the case in Figure 4-b.

In *pathload*, we characterize the trend in a stream as follows. When one of the PCT and

PDT metrics reports 'increasing trend', while the other is either 'increasing' or 'ambiguous', the stream is characterized as *type-I* (for 'increasing'). Similarly, when one metric reports 'non-increasing' trend, while the other is either 'non-increasing' or 'ambiguous', the stream is characterized as *type-N* (for 'non-increasing'). If both metrics report 'ambiguous', or when one is 'increasing' and the other is 'non-increasing', the stream is discarded.

E. Comparison of R and A after a fleet

After all N streams of a fleet are received, $\mathcal{RCV}$ determines whether $R > A$. If a large fraction f of the N streams in a fleet are of type-I, the entire fleet shows 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 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. We say, then, that the fleet rate R is in the *'grey-region'* of the avail-bw, and write $R \bowtie A$. In *pathload*, f is set to 70%.

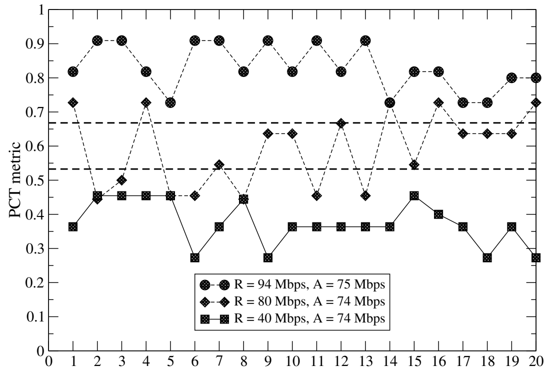


Fig. 5. PCTs for three fleets (20 streams each).

The previous three cases are illustrated in Figures 5 and 6. Figure 5 shows the PCTs for three fleets (with N=20 streams), while Figure 6 shows the PDTs for the same fleets. In the first fleet (R=94Mbps, A=75Mbps), all streams are characterized as type-I. In the second fleet (R=80Mbps, A=74Mbps), 11 streams are type-I, 6 are type-N, and 3 are discarded. So, in this case, the rate R is in the grey-region of the avail-bw ($R \bowtie A$). In the third fleet (R=40Mbps, A=74Mbps), all streams are characterized as type-N.

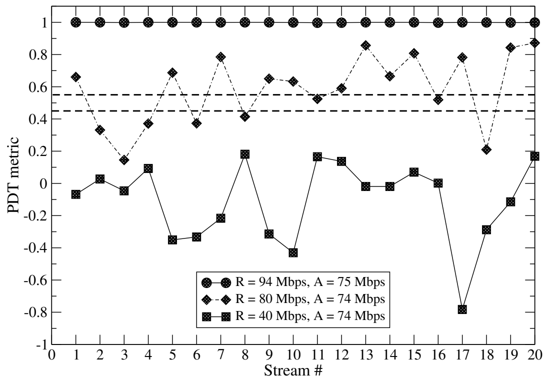


Fig. 6. PDTs for three fleets (20 streams each).

F. Adjustment of rate R

After a fleet n is over, *pathload* uses a rate adjustment algorithm to determine the rate $R(n+1)$ of the next fleet. This algorithm is shown in Figure 7. Some key elements in the algorithm are the state variables R^{min} and R^{max}, as well as G^{min} and G^{max}.

R^{min} is the highest rate that has been shown to be less than the avail-bw up to a certain point, while R^{max} is the lowest rate that has been shown to be higher than the avail-bw up to that point. In other words, $[R^{min}, R^{max}]$ is the narrowest bounding range for the avail-bw after each fleet.

Similarly, G^{min} is the lowest rate that has been shown to be in the grey-region, and G^{max} is the highest rate that has been shown to be in the grey-region after each fleet. So, $[G^{min}, G^{max}]$ is the widest inclusive range for the grey-region after each fleet.

Pathload converges to an avail-bw estimate when the range between the minimum and maximum avail-bw bounds is less than a user-specified resolution ω, i.e., when $R^{max} - R^{min} \leq \omega$. The tool also exits when $R^{max} - G^{max} \leq \chi$ and

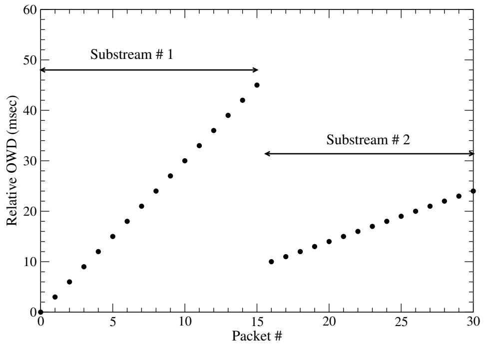


Fig. 8. A context switch at the sender.

$G^{min} - R^{min} \leq \chi$, i.e., when both avail-bw boundaries are within χ from the corresponding grey-region boundaries. In either case, *pathload* reports the range R^{min} to R^{max} as the final estimate. Note that, if $\omega < \chi$ and a grey region has been detected, the tool exits due to the latter condition. The reported range, in that case, has the width of the grey region, over-estimated by at most 2χ.

The initialization of the rate adjustment algorithm is described next. *Pathload* starts with an 'exponential phase' that is used to initialize R^{min} and R^{max}. The initial fleet rate $R(0)$ is either provided by the user, or a default value of 1Mbps is used. If $R(0) < A$, the rate is increased by a factor of two after each fleet, until $R(n) > A$. At that point, $R^{\max}{=}R(n)$ and $R^{min}{=}R(n-1)$. Similarly, if $R(0) > A$, the rate is decreased by a factor of two after each fleet, until $R(n) < A$. At that point, $R^{\min}{=}R(n)$ and $R^{max}{=}R(n-1)$.

G. How long does a pathload measurement take?

It is difficult to predict how long a *pathload* measurement will take, due the iterative nature of the algorithm. The following examples give a fair idea of the typical measurement latency in paths with $A \approx$ 10-100Mbps.

For the default tool parameters K=100 packets, T=100μs, and N=12 streams, and for an RTT Δ=90ms, the duration U of a fleet is 1.2sec. For a path with avail-bw A $\approx$ 10Mbps, *pathload* typically needs 10 fleets, or 12 seconds, to report an avail-bw estimate. Similarly, when the avail-bw A $\approx$ 75 Mbps, *pathload* needs 15-18 iterations, and the measurement latency increases to 18-22sec.

H. Clock skew compensation

The relative OWDs can be distorted by possible skew between the sender and receiver clocks [15]. We implemented the clock skew compensation algorithm of [15], but then realized that it is not really required in *pathload.* The reason is that, typically, the clock skew is 10-100 microseconds per second. Given that K=100 and T=100μs, a *pathload* stream lasts for only 10 milliseconds, and so the clock skew during a stream is typically less than one microsecond. This is insignificant compared to the OWD variations due to queueing.

I. Response to packet losses

After each stream is received, $\mathcal{RCV}$ measures the loss rate that the stream experienced. If a stream encountered moderate losses (<3%), the stream is marked as lossy. If more than a certain fraction of streams have been marked as lossy, then $\mathcal{RCV}$ informs $\mathcal{SND}$ to abort the remaining streams of the fleet. Also, if a stream encountered excessive losses (> 10%), the fleet is immediately aborted. The rate R at which the fleet was aborted becomes the new upper bound R^{max} of the rate adjustment algorithm.

J. Detection of a sender context switch

Pathload checks whether a context switch occurred at $\mathcal{SND}$ while a stream was being sent. Suppose that t_i is the transmission time of packet i from $\mathcal{SND}$. t_i is carried in packet i. $\mathcal{RCV}$ compares the sending times of consecutive packets to see whether $t_{i+1} - t_i > T + W$, where W is maximum allowed deviation from the target period T. If $t_{i+1} - t_i > T + W$, *pathload* takes the 'pessimistic' approach that $\mathcal{SND}$ was switched out after sending the i^{th} packet of a stream. Then, $\mathcal{RCV}$ splits the received stream into two substreams, one between packets 1 and i, and another between packets $i+1$ and K. If a substream includes less than $K/2$ packets, it is discarded from the OWD analysis. In the current version of *pathload*, W is set to 10ms.

Rate adjustment algorithm:

{

/* Initially: $G^{min} = G^{max} = 0$ */

if ($R(n) < A$) /* Non-increasing trend */
  $R^{min} = R(n)$;
  if ($G^{min} > 0$)
    $R(n+1) = (G^{min} + R^{min})/2$;
  else
    $R(n+1) = (R^{max} + R^{min})/2$;
else
if ($R(n) > A$) /* Increasing trend */
  $R^{max} = R(n)$;
  if ($G^{max} > 0$)
    $R(n+1) = (R^{max} + G^{max})/2$;
  else
    $R(n+1) = (R^{max} + R^{min})/2$;
else
  /* Grey-region */
  if ($G^{min} == G^{max} == 0$)
    $G^{min} = G^{max} = R(n)$;
  if ($G^{max} \leq R(n)$)
    $G^{max} = R(n)$;
    $R(n+1) = (R^{max} + G^{max})/2$;
  else if ($G^{min} > R(n)$)
    $G^{min} = R(n)$;
    $R(n+1) = (R^{min} + G^{min})/2$;

/* Termination conditions */
if ($R^{max} - R^{min} \leq \omega$ || ($G^{min} - R^{min} \leq \chi$ && $R^{max} - G^{max} \leq \chi$))
  return (R^{min}, R^{max});

}

Fig. 7. The rate adjustment algorithm.

Figure 8 shows an illustration of a 30-packet stream in which a context switch occurred at $\mathcal{SND}$. Packets 1-15 are sent before $\mathcal{SND}$ is switched out, while the remaining 15 packets are sent after it runs again. Both substreams show a increasing trend, indicating that a queue was building up at the tight link. In this example, we assume that the avail-bw changed between the two substreams, causing different increasing slopes in the OWDs of the two substreams.

K. Detection of a receiver context switch

Pathload also checks whether a context switch occurred at $\mathcal{RCV}$ while a stream was being received. Suppose that a_i is the arrival time of packet i at the $\mathcal{RCV}$ *pathload* process. If $\mathcal{RCV}$ is switched out while receiving a stream, some of the stream packets will be accumulated in a kernel buffer at the receiving host. When $\mathcal{RCV}$ runs again, those packets are transferred from kernel to user space with a spacing of Q μs, where Q is the latency of the *recvfrom* system call. Typically, Q is a few microseconds, and it can be measured at $\mathcal{RCV}$ before the measurements start. So,

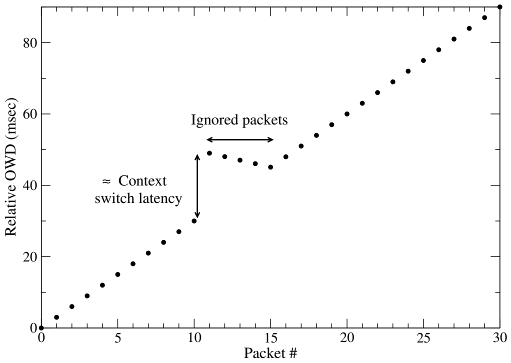


Fig. 9. A context switch at the receiver.

$\mathcal{RCV}$ can detect a local context switch comparing the arrival times of consecutive packets. If $a_{i+1} - a_i \approx Q$, packets i and $i+1$ are marked as backlogged at the kernel, and they are discarded from the OWD analysis.

Figure 9 shows an illustration of a 30-packet stream in which a context switch occurred at the receiver. Based on the arrival timestamps a_i, $\mathcal{RCV}$ determines that the context switch affected packets 11-14, and discards them from the OWD analysis. Packet 15 and onwards arrive at the receiver after $\mathcal{RCV}$ started running again. Hence, their OWD measurements are not affected by the previous context switch. Notice that we do not need to analyze packets 15-30 as a separate substream.

V. Verification experiments

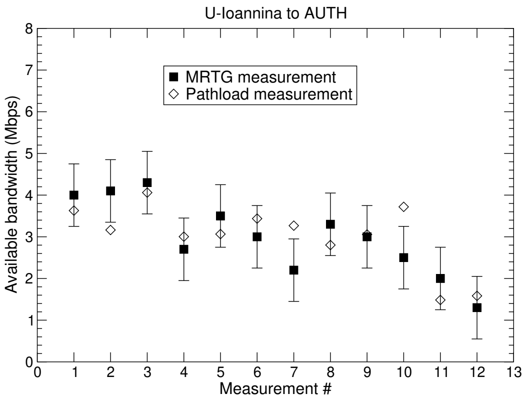


Fig. 10. Path #1

In this section, we present some experimental results that verify and illustrate the accuracy of *pathload*. The names and locations of the five hosts that we used in these experiments are given in Table V. Three hosts are located at Greek universities, connected through the academic & research network GRNET [16]. The U-Delaware host (*strauss*) and the U-Oregon host (*wayback*) are connected through the Abilene network, while the Abilene is connected to GRNET through the Dante (now GEANT) network in Europe. The underlying routes in these paths are almost always the same.

The main reason behind choosing these hosts for verification experiments is that we have access to MRTG graphs for most links in the path (especially for the most heavily utilized links), as well as information about their capacity. For more information about MRTG, please see the Appendix.

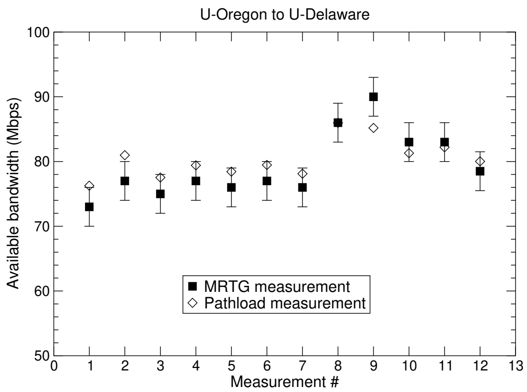


Fig. 11. Path #2

Note that an MRTG reading is an avail-bw measurement over a 5-min interval. *Pathload*, however, takes about 10-30 seconds to produce an estimate. To compare *pathload* with MRTG, we run *pathload* consecutively for 5 minutes. Suppose that in a 5-min interval we run *pathload* W times, and that run i lasted for q_i seconds, reporting an avail-bw range $[R_i^{min}, R_i^{max}]$ $(i = 1, \ldots W)$. The 5-min average avail-bw $\hat{R}$ that we eventually report for *pathload* is the following weighted average of the *central point* $(R_i^{min} + R_i^{max})/2$ in each of the W measured

Host	Location
strauss	Univ.Delaware, Newark DE, USA
wayback	Univ.Oregon, Eugene Or, USA
agwnia	Univ.Crete, Heraclion, Greece
vergina	Aristotle Univ. (AUTH), Thessaloniki, Greece
scylla	Univ.Ioannina, Ioannina, Greece

TABLE I
MEASUREMENT HOSTS AND THEIR LOCATIONS.

ranges:

$$\hat{R} = \sum_{i=1}^{W} \frac{q_i}{300} \frac{R_i^{min} + R_i^{max}}{2} \tag{8}$$

The *pathload* measurements shown next were collected during January 2002, both on weekdays and weekends. All graphs show end-to-end avail-bw measurements using *pathload*, as well as the corresponding MRTG avail-bw readings for the tight link of the path. The MRTG readings are given as ranges, due to the limited resolution of those graphs. Unless noted otherwise, this resolution was 1.5Mbps.

In the following experiments, the estimation resolution parameters of *pathrate* were chosen as ω=1Mbps and χ=1.5Mbps.

(C=8.2 Mbps). The *pathload* measurements are within the MRTG range in 9 out of 12 runs. The errors in the remaining 3 runs are less than 0.5Mbps.

Figure 11 refers to the path from U-Oregon to U-Delaware. An interesting point about this path is that the tight link is different than the narrow link. The former is an IP-over-SONET 155Mbps OC3 link that connects the Oregon GigaPOP to the Abilene, while the latter is a Fast Ethernet interface (100Mbps). The best resolution we could get from the MRTG readings in this case is 6Mbps. The *pathload* measurements are within the MRTG range in 10 out of 12 runs, with an error of less than 2Mbps in the remaining runs.

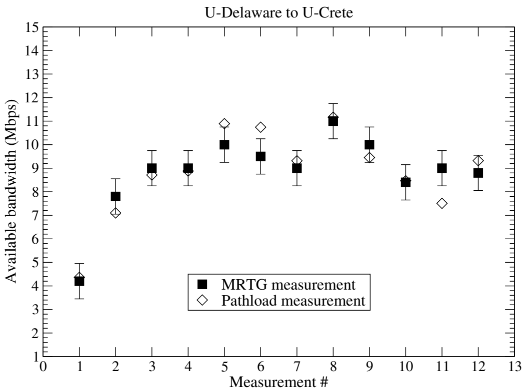


Fig. 12. Path #3

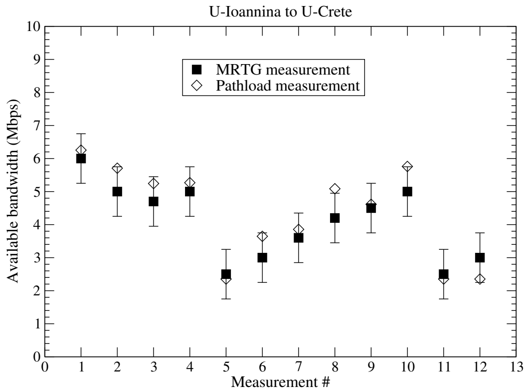


Fig. 13. Path #4

Figure 10 compares the MRTG and *pathload* measurements for twelve 5-min intervals at the path from U-Ioannina to AUTH in Greece. The tight (and narrow) link in this path is the access link that connects Univ-Ioannina to GRNET

Figure 12 refers to the path from U-Delaware to U-Crete. The tight & narrow link is the access link that connects U-Crete to GRNET (C=12.4Mbps). The *pathload* measurements are within the MRTG range in 9 out of 12 cases, with marginal errors in the remaining runs.

Figure 13 refers to the path from U-Ioannina and U-Crete. The tight & narrow link is the U-Ioannina access link. The *pathload* measurements are within the MRTG ranges in 11 out of 12 runs.

Finally, Figure 14 refers to the path from U-Ioannina to U-Oregon. Again, the tight link is the U-Ioannina access link. The *pathload* measurements are within the MRTG ranges in 8 out of 12 runs, with an an error of less than 0.5Mbps in the remaining runs.

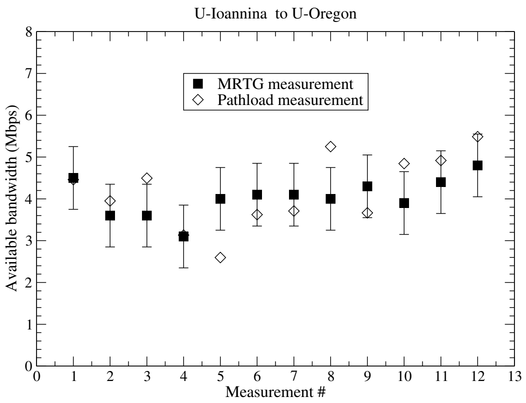


Fig. 14. Path #5

VI. Summary

Available bandwidth measurements can be useful in transport protocols, dynamic server and proxy selection, adaptive reconfiguration of overlay networks, and in rate-adaptive streaming applications.

We described in detail an active measurement tool, called *pathload*, that estimates the end-to-end avail-bw in a network path. Our verification experiments show that the *pathload* measurements are comparable to the MRTG avail-bw readings at the tight link of the path. We plan to release the source code for *pathload* in Spring 2002.

In the follow-up work [1], we further show that *pathload can measure the avail-bw in a path in a non-intrusive manner*, i.e., without affecting the throughput of other connections, without generating significant traffic load, and without causing a persistent increase in the queueing delays or losses at the path.

Appendix: Verification using MRTG

MRTG is a widely used tool that displays the utilized bandwidth of a link, based on information that comes directly from the router interface [17]. Specifically, MRTG periodically reads the number of bytes sent and received from the router's Management Information Base using SNMP. The default measurement period is 5 minutes, and thus the MRTG bandwidth readings should be interpreted as 5-min averages. If the capacity of the link is also known, the avail-bw in the same time interval is the capacity minus the utilized bandwidth. Figure 15 shows a

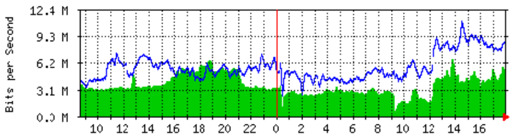


Fig. 15. Example of an MRTG graph for a 14Mbps link. The x-axis is measured in hours.

snapshot of an MRTG graph for a 14Mbps duplex link. The two curves (shaded vs line) refer to the two directions of the link (IN vs OUT).

MRTG offers a primitive way to verify *pathload* measurements, based on 5-minute avail-bw measurements for individual links. In the paths that we experiment with, we have access to MRTG graphs for most links in the path (especially for the most heavily utilized links), as well as information about their capacity. The tight link remains the same for the duration of many hours in these paths, and so the measurement of end-to-end avail-bw is based on a single MRTG graph. Even though this verification approach is not too accurate, it gives us avail-bw measurements with a resolution of 1-6Mbps.

The throughput measurements shown at an MRTG graph refer to layer-2 traffic, whereas *pathload* measures layer-3 (IP) throughput. In the verification experiments, we reduce the MRTG avail-bw measurements by a certain factor to compensate for the layer-2 header overhead. We compute this factor separately for each path, depending on the layer-2 technology that is used at the tight link of the path.

Acknowledgments

We are grateful to the following colleagues for providing us 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. Stavrakakis (Univ-Athens), and L. Tassiulas (Univ-Ioannina). This work would not be feasible without their help.

References

[1] M. Jain and C. Dovrolis, "End-to-end available bandwidth: measurement methodology, dynamics, and relation with TCP throughput," Tech. Rep., University of Delaware, Feb. 2002.

[2] V. Jacobson, "pathchar: A Tool to Infer Characteristics of Internet Paths," ftp://ftp.ee.lbl.gov/pathchar/, Apr. 1997.

[3] A.B. Downey, "Using Pathchar to Estimate Internet Link Characteristics," in *Proceedings of ACM SIGCOMM*, Sept. 1999.

[4] B. A. Mah, "pchar: a Tool for Measuring Internet Path Characteristics," http://www.employees.org/~bmah/Software/pchar/, June 2000.

[5] K. Lai and M.Baker, "Measuring Link Bandwidths Using a Deterministic Model of Packet Delay," in *Proceedings of ACM SIGCOMM*, Sept. 2000.

[6] R. L. Carter and M. E. Crovella, "Measuring Bottleneck Link Speed in Packet-Switched Networks," *Performance Evaluation*, vol. 27,28, pp. 297–318, 1996.

[7] K. Lai and M.Baker, "Measuring Bandwidth," in *Proceedings of IEEE INFOCOM*, Apr. 1999.

[8] C. Dovrolis, P. Ramanathan, and D. Moore, "What do Packet Dispersion Techniques Measure?," in *Proceedings of IEEE INFOCOM*, Apr. 2001.

[9] V. Paxson, "End-to-End Internet Packet Dynamics," *IEEE/ACM Transaction on Networking*, vol. 7, no. 3, pp. 277–292, June 1999.

[10] M. Mathis and M. Allman, *A Framework for Defining Empirical Bulk Transfer Capacity Metrics*, July 2001, RFC 3148.

[11] M. Allman, "Measuring End-to-End Bulk Transfer Capacity," in *Proceedings of ACM SIGCOMM Internet Measurement Workshop*, Nov. 2001.

[12] G. Jin, G. Yang, B. Crowley, and D. Agarwal, "Network Characterization Service (NCS)," in *Proceedings of 10th IEEE Symposium on High Performance Distributed Computing*, Aug. 2001.

[13] V. Robeiro, M. Coates, R. Riedi, S. Sarvotham, B. Hendricks, and R.Baraniuk, "Multifractal Cross-Traffic Estimation," in *Proceedings ITC Specialist Seminar on IP Traffic Measurement, Modeling, and Management*, Sept. 2000.

[14] B. Melander, M. Bjorkman, and P. Gunningberg, "A New End-to-End Probing and Analysis Method for Estimating Bandwidth Bottlenecks," in *Global Internet Symposium*, 2000.

[15] V. Paxson, "On Calibrating Measurements of Packet Transit Times," in *Proceedings of ACM SIGMETRICS*, June 1998.

[16] GRNET, "Greek Research and Technology Network," http://netmon.grnet.gr/traffic/.

[17] T. Oetiker, "MRTG: Multi Router Traffic Grapher," http://ee-staff.ethz.ch/~oetiker/webtools/mrtg/mrtg.html.