Measuring Link Bandwidths Using a Deterministic Model of Packet Delay

Kevin Lai
Stanford University
laik@cs.stanford.edu

Mary Baker
Stanford University
mgbaker@cs.stanford.edu

ABSTRACT

We describe a deterministic model of packet delay and use it to derive both the *packet pair* [2] property of FIFO-queueing networks and a new technique (*packet tailgating*) for actively measuring link bandwidths. Compared to previously known techniques, packet tailgating usually consumes less network bandwidth, does not rely on consistent behavior of routers handling ICMP packets, and does not rely on timely delivery of acknowledgments.

Preliminary empirical measurements in the Internet indicate that compared to current measurement tools, packet tailgating sends an order of magnitude fewer packets, while maintaining approximately the same accuracy. Unfortunately, for all currently available measurement tools, including our prototype implementation of packet tailgating, accuracy is low for paths longer than a few hops.

1. INTRODUCTION

As long as Internet bandwidth has increased, the amount of traffic sent over the Internet has grown to consume it. This means that despite the increasing link bandwidth in network backbones and into homes and offices, optimizing the use and allocation of bandwidth continues to be an interesting problem. Although many applications are more interested in available bandwidth than link bandwidth, knowing the link bandwidth along a path enables more accurate measurement of available bandwidth. In addition, several applications can directly use link bandwidth, including planning networks to minimize bottlenecks and analyzing network performance as a whole [14].

However, the Internet's current size, heterogeneity, and rate of change make determining link bandwidth a challenging research problem. This is true even though applications are usually only interested in the bandwidth along a particular path or even just the smallest bandwidth (the *bottleneck* bandwidth) along that path. A database to store bandwidth information would neither scale well nor cope with the rate at which routes change [13]. Routers currently do not report link bandwidths. Since routers gain much of their speed by being as simple as possible, slowing them to answer link bandwidth queries is probably not acceptable. The easiest approach to deploy, and consequently the one in which we are most interested, is for end hosts to infer link bandwidth by actively probing or passively listening to traffic. Hosts can share this information if the probing or listening is expensive [17].

The challenge of the inference approach is to measure link bandwidth as accurately, quickly, robustly, and unobtrusively as possible. By robust we mean it is usable in the variety of network environments that exists in the Internet: few or many hops from source to destination, empty or saturated links, one or several channels per link, different wired and wireless link technologies, different queueing disciplines, and different router implementations. By unobtrusive we mean it places minimimal additional load on the network, since it is desirable to prevent measurement traffic from delaying application data traffic.

As we describe in Section 2, existing solutions to infer all link bandwidths along a path are built from a deterministic model that considers only one measurement packet. As a result, these techniques rely on routers handling ICMP packets consistently, and timely delivery of acknowledgements. These techniques use significant amounts of network bandwidth to perform their measurements and can be slow enough to become impractical for some of the applications that most need them.

In this paper, we describe a new deterministic model of packet delays that unifies previous models. Using this model, we derive a novel technique, called *packet tailgating*, for measuring link bandwidths along a path through the Internet. Packet tailgating captures link-specific characteristics by causing queuing of packets at particular links. For each link, the technique sends a large packet with a time-to-live (TTL) set to expire at that link followed by a very small packet that will queue continuously behind the large packet until where the large packet expires.

Packet tailgating consumes less network bandwidth than previous techniques, does not rely on consistent router behavior for handling ICMP packets, does not rely on timely delivery of acknowledgements, can theoretically detect multichannel links and can be run on multicast trees. Using our

SIGCOMM '00, Stockholm, Sweden.

Table 1: Variable Definitions.

n links	hop length of the path
d_l sec.	latency of link l
d^l sec.	sum of latencies up and including link l
b_l bits/sec.	bandwidth of link l
s^k bits	size of packet k
t_l^k sec.	time when packet k fully arrives at link l
q_l^k sec.	amount of time packet k is queued at link l
l_{bn} link number	the bottleneck link

prototype tailgating implementation on a path through the Internet, we demonstrate that packet tailgating sends an order of magnitude fewer packets than previous techniques while maintaining similar accuracy. Unfortunately, all currently available bandwidth measurement tools, including our prototype implementation of packet tailgating, have low accuracy on paths longer than a few hops.

In addition, we use our multi-packet delay model to derive the *packet pair* [2] property of FIFO-queueing networks which has previously been used to measure bottleneck link bandwidth. Packet pair has previously been derived for fair-queueing networks [9], but not for FIFO-queuing networks.

The rest of the paper is organized as follows. In Section 2 we review related work on packet delay models used for measuring link bandwidths along a path and for measuring bottleneck bandwidth. In Section 3 we present a multi-packet deterministic model of packet delay and show how we can use it to derive previous packet delay models and the new tailgating technique. In Section 4, we describe how we use the technique to measure link bandwidths and analyze the technique's strengths and weaknesses. In Section 5, we compare preliminary measurements of our packet tailgating implementation with those of previous link bandwidth techniques. In Section 6, we conclude. We include a derivation of the packet pair property in the appendix.

2. PREVIOUS WORK

In this section, we describe previous work using deterministic models to measure link bandwidth. Deterministic models are typically easier to work with mathematically than stochastic models, enabling us to find an analytical solution rather than a numerical one. Unfortunately, a deterministic model implies modeling events with absolute certainty, and many things cannot be known with enough certainty to make this practical. Although the Internet qualifies as a system where many things cannot be known with certainty, some aspects of Internet behavior can be reasonably described using a deterministic model. In particular, previous work on bandwidth measurement has used a deterministic model along with filtering to fit empirical data to that model.

2.1 The One-Packet Model

Bellovin [1], Jacobson [8], and Downey [4] measure link bandwidths using what we call the *one-packet* model for packet delay. This model uses the following equation (variables defined in Table 1):

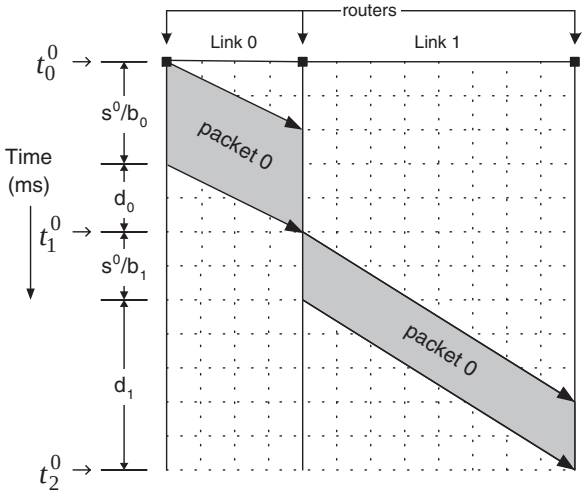

Figure 1: This figure shows the amount of time a packet spends on links 0 and 1. In this example, s^0 = 6000 bits, b_0 = 2Mb/s, d_0 = 2ms, b_1 = 3 Mb/s, d_1 = 5ms. This packet travels across these two links in $t_2^0 = \sum_{i=0}^{1} \left(\frac{s^0}{b_i} + d_i\right) = 12ms$.

$$t_l^0 = t_0^0 + \sum_{i=0}^{l-1} \left(\frac{s^0}{b_i} + d_i\right) \quad (1)$$

This equation predicts the time needed for one packet to travel across the $l-1$ links before the lth node. Each link contributes some amount of transmission delay $\left(\frac{s^0}{b_i}\right)$ and latency (d_i). The transmission delay is due to the time that a router takes to copy a packet around in buffers and serialize a packet onto a link. The latency is due to the time for a signal to travel at the speed of light, the time for a router to look up routes in a routing table, and other fixed per-packet delays that a router incurs before it can forward a packet. An example of using the one-packet model to obtain packet delay is shown in Figure 1.

The one-packet model assumes that the transmission delay is linear with respect to packet size, routers are store-and-forward, links are single-channel, and no other traffic in the path causes the measurement packet to queue. The assumption that transmission delay is linear with respect to packet size may not be true if, for example, a router manages its buffers in such a way that a 128 byte packet is copied more than proportionally faster than a 129 byte packet. However, this effect is usually small enough to be ignored. The assumption that routers are store-and-forward (they receive the last bit of the packet before forwarding the first bit) is almost always true in the Internet.

A more limiting assumption is that links are composed of only one channel. This assumption is a result of the model considering only one packet. In practice, some links stripe traffic packet by packet across multiple channels to make them appear as one link. For example, BRI ISDN links

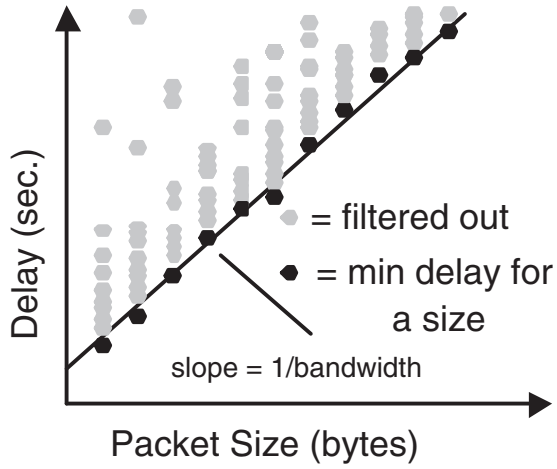

Figure 2: This figure illustrates how the one-packet model and linear regression are used to determine bandwidth. The graph shows several hypothetical measurements of the round trip delay of packets of different sizes traveling along the same path. The gray samples experienced queueing. The black samples did not experience queueing. The line is the linear regression of the black samples. The inverse of the slope is the bandwidth of the path.

are usually composed of two 64Kb/s channels in parallel. Equation 1 will detect this as a single 64Kb/s link [4].

The final and most problematic assumption is that other traffic does not cause the measurement packet to queue. The assumption arises from the deterministic nature of the model: we do not know when the other packets were sent and what size they are, so we cannot model them deterministically. In the Internet, this assumption is almost always false. In the next section, we describe how one-packet techniques work around this.

We describe a multi-packet model in Section 3 that is able to detect multi-channel links and to use intra-flow packets (i.e., from the same source and to the same destination) for better measurement. However, the model shares the assumptions that transmission delay is linear with packet size and that no extra-flow packets in the path will cause the measurement packets to queue.

2.2 One-Packet Techniques

Bellovin and Jacobson use the one-packet delay model to develop a technique for measuring link bandwidths. Although Equation 1 specifies the one-way delay, Bellovin and Jacobson instead use the round-trip delay to successive routers along a path. The round-trip delay can be modeled as the sum of the one-way delay for the initial packet and that of its acknowledgement.

Bellovin and Jacobson resolve the problematic assumption about no queueing by observing that queueing caused by additional traffic can only increase delays. Therefore, the minimum of several observed delays of a particular packet size fits the model. Their technique is to send several packets for each of several different packet sizes, plot the delays of these packets versus their sizes, and then use linear regression (Figure 2) to obtain the slope of the graph. The inverse of the slope is the bandwidth.

In practice, the problems with this technique are that linear regression is expensive, routers are not built to send acks in a timely manner, some nodes are "invisible", and the reverse path adds noise.

The first problem is that the linear regression described above must be done for every link measured. Many packets may be required to filter out the effect of other traffic and calculate a regression with high confidence. Jacobson [7] provides `pathchar` as an implementation of the algorithms just described. Using its default settings, it will send 10MB of data in the course of measuring a 10 hop Ethernet path [10].

Downey [4] uses statistical methods to reduce measurement traffic. Once he detects the convergence of a link bandwidth estimate, then he avoids sending further packets to measure this link. Methods such as this are complementary to our packet tailgating technique.

The second problem is that the one-packet technique requires getting timely acknowledgements from routers. Bellovin uses Internet Control Message Protocol (ICMP) Echo and Echo Reply packets sent to the routers, while Jacobson and Downey use UDP packets, successively incrementing the IP Time-To-Live (TTL) field to receive ICMP time exceeded responses from the routers.

These approaches have the advantage that no special software needs to be deployed on routers to gather timing information, but unfortunately they may not work in all parts of the Internet. Because of malevolent use of ICMP packets, some routers and hosts either rate-limit them or filter them out [16], thus slowing down or precluding measurement.

Another problem is that bridges, host operating systems (OSs), and network interface cards (NICs) are usually store-and-forward nodes but do not decrement the IP TTL and are not individually addressable in IP. Consequently, the links corresponding to these "invisible" nodes cannot be detected or measured using the IP TTL decrement method cited above. There is a node between the source application and the source operating system because the sending OS usually must copy packets from the application's address space to the kernel's. In addition, the source OS's network card driver usually must copy the sent packet from kernel address space across the system bus to the NIC. Finally, if the destination is a PC, the packet usually must be copied from the destination's NIC to the destination's kernel address space. The application–kernel, kernel–NIC, and NIC–kernel copies usually must be individually complete before the packet can be forwarded any further in the pipeline. These invisible nodes cause error in the measurement of the next link.

The final problem is that relying on acknowledgements and

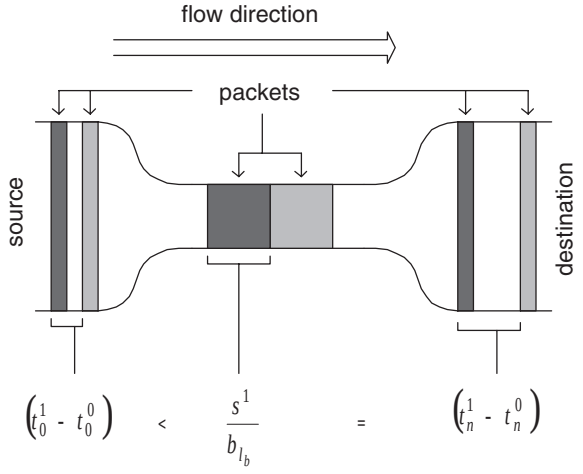

Figure 3: This figure shows two packets of the same size traveling from the source to the destination. The wide part of the pipe represents a high bandwidth link while the narrow part represents a low bandwidth link. The spacing between the packets caused by queueing at the bottleneck link remains constant downstream because there is no additional downstream queueing.

round-trip delays means that there is twice the possibility that queueing could corrupt a sample when compared to a technique that relies only on one-way delay. This is because queueing in the reverse path can delay the acknowledgement, even if there is no queueing in the forward path. As a result, many packets may be required to filter out the effect of other traffic and calculate a regression with high confidence. To use one-way delay, one-packet-based techniques to use one-way delay would need new software at every router on a path, which would not be practical.

The packet tailgating technique described in Section 4 can overcome most of these limitations. It erforms linear regression only once instead of once for each link. Because it does not rely on timely delivery of acknowledgements from routers, it is robust against routers that generate ICMP packets inconsistently. Finally, it can increase accuracy and reduce the number of packets sent by measuring one-way delay instead of round trip delay, without requiring new software at routers. Packet tailgating still suffers from the invisible node problem; we partially address it in our implementation described in Section 4.1.

2.3 The Packet Pair Model

Some applications are only interested in the smallest bandwidth link along a path (the *bottleneck* link) rather than the bandwidths of all links in a path. An advantage of measuring bottleneck bandwidth is that it can be done with as few as two packets instead of thousands. Bolot [2], Carter and Crovella [3], Paxson [14], and Lai and Baker [10] use the *packet pair* model and technique to measure the bottleneck bandwidth.

In contrast to the one-packet model described in Section 2.1, packet pair in FIFO-queueing networks uses a two-packet model. This model predicts the difference in arrival times of two packets of the same size traveling from the same source to the same destination:

$$t_n^1 - t_n^0 = \max\left(\frac{s_1}{b_{l_{bn}}}, t_0^1 - t_0^0\right) \quad (2)$$

The variables in this equation are defined in Table 1. The intuitive rationale for this equation (we give an analytical one in Appendix A) is that if two packets are sent close enough together in time to cause the packets to queue together at the bottleneck link ($\frac{s_1}{b_{l_{bn}}} > t_0^1 - t_0^0$), then the packets will arrive at the destination with the same spacing ($t_n^1 - t_n^0$) as when they exited the bottleneck link ($\frac{s_1}{b_{l_{bn}}}$). The spacing will remain the same because the packets are the same size and no link downstream of the bottleneck link has a lower bandwidth than the bottleneck link (as shown in Figure 3, which is a variation of a figure from [6]).

This algorithm makes several assumptions that may not hold in practice. First, the packet pair model assumes that the two packets queue together at the bottleneck link and at no later link. This could by violated by other packets queueing between the two packets at the bottleneck link, or packets queueing in front of the first, the second or both packets downstream of the bottleneck link. If any of these events occur, then Equation 2 does not hold. In practice, previous work mitigates this limitation by filtering out samples that suffer undesirable queueing.

In addition, this model assumes that the two packets are sent close enough in time that they queue together at the bottleneck link. Carter actively sends traffic for measurement purposes to guarantee that this is the case. Paxson and Lai avoid this cost by using existing TCP traffic.

Another assumption is that the bottleneck router uses FIFO-queueing. If the router uses weighted fair queueing, then packet pair measures the available bandwidth of the bottleneck link [9].

Finally, the packet pair model, like the one-packet model, assumes that transmission delay is linear with respect to packet size and that routers are store-and-forward.

In contrast to the one-packet model, an extension to packet pair avoids the assumption that links are single-channel by considering *bunches* of more than two packets [14]. This extension is also able to consider other packets in the same flow. Unfortunately, it does not consider the per-link latencies taken into account in the one-packet model. The multi-packet model we present in the following section considers both.

3. A MULTI-PACKET MODEL

In this section we describe a deterministic multi-packet model that is more powerful than the models described in the previous section and show how it can be used to measure link bandwidths along a path. The new model takes into account all the packets in a single flow as well as latency. We developed the multi-packet model as a result of attempting to unify the one-packet and packet pair models. In fact, as we show below, the multi-packet model can derive both

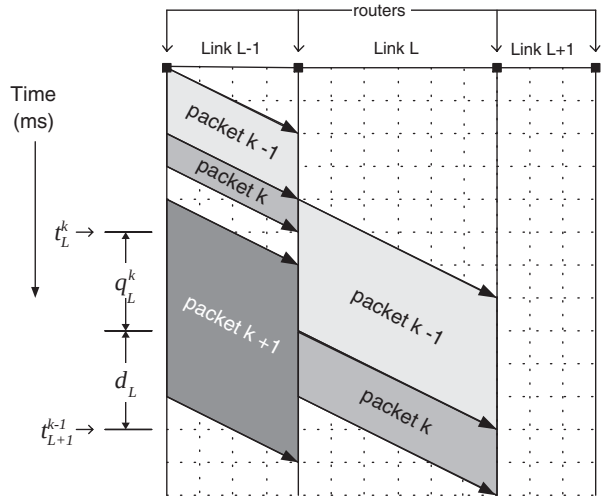

Figure 4: This figure shows the amount of time several packets from a flow spend on links $l-1$ and l. In this example, s^{k-1} = 4000 bits, s^k = 2000 bits, s^{k+1} = 12000 bits, b_{l-1} = 2Mb/s, d_{l-1} = 2ms, b_l = 1 Mb/s, d_l = 3ms. Packet k is queued at link l for $q_l^k = \max(0, 11-3-5)$ = 3ms. Packet $k+1$ arrives 1ms after packet k leaves because something delayed it earlier in the path.

the one-packet and packet pair models as well as the new tailgating technique.

3.1 Multi-Packet Delay Equation

The multi-packet model consists of a delay equation derived from two other equations: an arrival time equation and a queueing delay equation. The following arrival time equation is a slight variation on the one-packet equation (1) (variables defined in Table 1):

$$t_l^k = t_0^k + \sum_{i=0}^{l-1} \left(\frac{s^k}{b_i} + d_i + q_i^k \right) \quad (3)$$

This equation predicts that packet k arrives at link l at its transmission time (t_0^k) plus the sum over all the previous links of the latencies (d_i), transmission delays $\left(\frac{s^k}{b_i}\right)$, and queueing delays ($q_i^k$) of those links. Equation 3 differs from the one-packet equation (1) in that it considers $k-1$ packets in the same flow and the queueing delays of those packets.

We model the queueing delay due to other packets in the same flow using the following equation:

$$q_l^k = \max\left(0, t_{l+1}^{k-1} - d_l - t_l^k\right) \quad (4)$$

This equation predicts that packet k is queued at the router just before link l from the time it arrives at that router (t_l^k) until it can begin transmitting, which is the time when the previous packet ($k-1$) arrives at the next router (t_{l+1}^{k-1}) minus the latency of this link (d_l). We assume that the first packet is never queued ($q_0^0 = \cdots = q_{n-1}^0 = 0$). An example of using Equation 4 to compute queueing delay is shown in Figure 4. This equation is equivalent to those described by Paxson [15] and Stoica [18]. Notice that packet $k+1$ in the figure is not queued at all because it arrives at link l after packet k has been transmitted. Queueing delay cannot be negative, so the max() function in the queueing equation causes it to be 0 in this case.

We combine Equations 3 and 4 to form the multi-packet delay equation:

$$t_l^k = t_0^k + \sum_{i=0}^{l-1} \left(\frac{s^k}{b_i} + d_i + \max\left(0, t_{i+1}^{k-1} - d_i - t_i^k\right) \right) \quad (5)$$

The multi-packet equation is as least as powerful as both the one-packet and packet pair models because we can derive both of those models from Equation 5. We reduce the multi-packet equation to the one-packet equation (1) by taking $k = 0$. To derive the packet pair equation from Equation 5 we reformulate the packet pair equation as the following property:

THEOREM 3.1 (PACKET PAIR PROPERTY). *Let $b_{min(l)} \leq b_i, (\forall i,\ 0 \leq i \leq l)$, then if we send two packets of the same size ($s^0 = s^1$) at the same time ($t_0^0 = t_0^1$) and there is no cross traffic, they will arrive with a difference in time equal to the size of the second packet divided by the smallest bandwidth on the path ($t_n^1 - t_n^0 = \frac{s^1}{b_{min(n-1)}}$).*

Using Equation 5, we derive the Packet Pair Property in Appendix A.

As well as combining the advantages of previous models, the multi-packet model combines the assumptions of previous models. The model assumes that packets from other flows do not cause queueing in the modeled flow. As with previous models, this assumption can be worked around by using the minimum of several observed delays of particular packet size. Like previous models, it assumes that transmission delay is linear with respect to packet size and that routers are store-and-forward.

3.2 Link Bandwidth Measurement

In addition to deriving previous models, we use the multi-packet model to develop a new technique for link bandwidth measurement. To do so we assume that we can send one packet with no queueing and a second packet that queues behind the first packet at a specific link, but not at any later link. We describe how we ensure this assumption in practice in Section 4.

The basic approach we take is to start with the multi-packet delay equation and try to solve for the bandwidth b_{l_q} of the link l_q at which queueing occurs. We rewrite Equation (5) to state the time packet k takes to arrive at the destination link n (with variables defined in Table 1):

$$t_n^k = t_0^k + \sum_{i=0}^{n-1} \left(\frac{s^k}{b_i} + d_i + \max\left(0, t_{i+1}^{k-1} - d_i - t_i^k\right) \right)$$

Assuming no queuing except at the queuing link, we can split this delay into the time to travel to the queueing link, the time spent at the queueing link, and the time spent after the queueing link:

$$t_n^k = \left[t_0^k + \sum_{i=0}^{l_q-1} \left(\frac{s^k}{b_i} + d_i \right) \right] + \left[\frac{s^k}{b_{l_q}} + t_{l_q+1}^{k-1} - t_{l_q}^k \right] + \left[\sum_{i=l_q+1}^{n-1} \left(\frac{s^k}{b_i} + d_i \right) \right]$$

We substitute using (5) and simplify:

$$= t_{l_q}^k + \frac{s^k}{b_{l_q}} + t_{l_q+1}^{k-1} - t_{l_q}^k + \sum_{i=l_q+1}^{n-1} \left(\frac{s^k}{b_i} + d_i \right)$$

$$= \frac{s^k}{b_{l_q}} + t_{l_q+1}^{k-1} + \sum_{i=l_q+1}^{n-1} \left(\frac{s^k}{b_i} + d_i \right)$$

We substitute using (5), include the assumption that the first packet experiences no queueing, and simplify:

$$= \frac{s^k}{b_{l_q}} + \sum_{i=0}^{l_q} \left(\frac{s^{k-1}}{b_i} + d_i \right) + t_0^{k-1} + \sum_{i=l_q+1}^{n-1} \left(\frac{s^k}{b_i} + d_i \right)$$

$$= \frac{s^{k-1}}{b_{l_q}} + \sum_{i=0}^{l_q-1} \left(\frac{s^{k-1}}{b_i} \right) + \sum_{i=l_q}^{n-1} \left(\frac{s^k}{b_i} \right) + t_0^{k-1} + \sum_{i=0}^{n-1} (d_i)$$

Before continuing with the derivation, we define the following variables for more compact notation:

$$d^l = \sum_{i=0}^{l} d_i \quad \frac{1}{b^l} = \sum_{i=0}^{l} \left(\frac{1}{b_i} \right)$$

Using these definitions, we continue simplifying the equation:

$$= \frac{s^{k-1}}{b_{l_q}} + s^{k-1} \sum_{i=0}^{l_q-1} \left(\frac{1}{b_i} \right) + s^k \sum_{i=l_q}^{n-1} \left(\frac{1}{b_i} \right) + t_0^{k-1} + d^{n-1}$$

$$= \frac{s^{k-1}}{b_{l_q}} + \frac{s^{k-1}}{b^{l_q-1}} + s^k \left(\sum_{i=0}^{n-1} \left(\frac{1}{b_i} \right) - \sum_{i=0}^{l_q-1} \left(\frac{1}{b_i} \right) \right) + t_0^{k-1} + d^{n-1}$$

$$= \frac{s^{k-1}}{b_{l_q}} + \frac{s^{k-1}}{b^{l_q-1}} + s^k \left(\frac{1}{b^{n-1}} - \frac{1}{b^{l_q-1}} \right) + t_0^{k-1} + d^{n-1}$$

Solving for b_{l_q} and collecting terms,

$$b_{l_q} = \frac{s^{k-1}}{(t_n^k + \frac{s^k - s^{k-1}}{b^{l_q-1}} - \frac{s^k}{b^{n-1}} - t_0^{k-1} - d^{n-1})} \qquad (6)$$

This shows that we can compute the bandwidth of a link at which queuing occurs (b_{l_q}) from the sizes of the two packets (s^{k-1}, s^k), the arrival time of the second packet (t_n^k), the transmission time of the first packet (t_0^{k-1}), the bandwidth of all earlier links (b^{l_q-1}) and (b^{n-1}), and the delay of all earlier links (d^{n-1}). To use this equation in practice, we have to solve various problems including calculating the inter-packet transmission time, dealing with clock skew, and using round trip measurements. We describe our solutions to these and other implementation problems in the next section.

4. PACKET TAILGATING TECHNIQUE

In this section, we describe the tailgating technique and our prototype implementation, (`nettimer`). We also analyze the advantages and disadvantages of the technique.

The technique is divided into two phases: the sigma phase, which measures the characteristics of the entire path, and the tailgating phase, which measures the characteristics of each link individually.

The purpose of the sigma phase is to measure the non-link-specific quantities described in the previous section: b^{n-1} and d^{n-1}. We do this by sending single packets of different sizes and using linear regression on the minimum delay for each size. We send packets until the confidence of the linear regression exceeds 99%. This is the same technique (described in Section 2) that Jacobson and Downey use. As mentioned before, this step requires many packets. While previous work does this step for every router along a path, we only do it once from the source to the destination to calculate b^{n-1} and d^{n-1} for (6).

The purpose of the tailgating phase is to compute the remainder of the variables in (6). As mentioned before, the tailgating technique assumes that we can send one packet without queueing and a second packet that queues after the first packet at link l_q, but at no later link. We do this by sending the largest possible non-fragmented packet with an IP Time-to-Live (TTL) field of l_q immediately followed by the smallest possible packet. The smaller packet almost always (the exception is discussed in Section 4.2) has a lower transmission delay than the larger packet's transmission delay on the next link. This causes the smaller packet (the tailgater) to queue continuously after the larger packet (the tailgated). The TTL for the tailgated packet will cause it to be dropped at link l_q, so the tailgater can then continue without queueing to the destination.

As with the sigma phase, we take the minimum of several delay samples of the tailgater packet. We start this process with a TTL of 1 and continue until we reach the destination. We measure closer links first to compute b^{l_q-1} for later links.

In our `nettimer` implementation, we randomly probe different links until the error for all of them drops below 2%. This value was chosen to balance the time to finish with the accuracy of the results. We calculate the error for a particular link by using the bootstrap method [5]. We randomly re-sample with replacement from the set of delays for a link until we have a new set of samples of 25% of the size of the original. Using this new set, we compute a new minimum delay. We repeat this process 20 times and then compute the variance of all the new delays. The error is this variance divided by the actual minimum delay of the set. These values are selected so that the CPU overhead of the computation is unnoticable at typical network latencies.

4.1 Tailgating Complexities

In addition to the basic technique described above, we have to deal with several additional complexities: 1) calculating the maximum inter-packet transmission time, 2) clock skew, 3) using round trip measurements, 4) causing acknowledgements to be sent, 5) detecting that packets were dropped, and 6) dealing with invisible nodes.

4.1.1 Inter-packet Transmission Time

As discussed in 3.2, the tailgated and tailgater packets must be sent such that queueing takes place at the link to be measured. We could guarantee this by sending the packets at the same instant in time, but this is impossible. Here we derive the maximum time we have between each packet transmission so that (6) remains valid. There may be scheduling variations in the source host's operating system such that the deadline sometimes cannot be met, so we want to determine when the deadline is to filter out those measurements.

We want packet k to queue at link l_q:

$$q_{l_q}^k > 0$$

Using (4),

$$\max\left(0, t_{l_q+1}^{k-1} - d_{l_q} - t_{l_q}^k\right) > 0$$
$$t_{l_q}^k < t_{l_q+1}^{k-1} - d_{l_q}$$

Using (3) and the assumption that the tailgated and tailgater packets experience no queueing before l_q,

$$t_0^k + \sum_{i=0}^{l_q-1}\left(\frac{s^k}{b_i} + d_i\right) < t_0^{k-1} + \sum_{i=0}^{l_q}\left(\frac{s^{k-1}}{b_i} + d_i\right) - d_{l_q}$$

Simplifying,

$$t_0^k - t_0^{k-1} < \frac{s^{k-1}}{b_{l_q}} + \frac{s^{k-1} - s^k}{b^{l_q-1}}$$

This means that we might have trouble sending the tailgater packet quickly enough if closer links have high bandwidth. For example, if the tailgated packet is 1500 bytes and the first link has a bandwidth of 100Mb/s, then we must send the tailgater packet within 120 microseconds, which is possible on most machines. However, if the first link is 1Gb/s, then we only have 12 microseconds to send the tailgater. In this case, we can use the regression technique on closer links until we are at a link sufficiently far away that we can meet the transmission deadline.

4.1.2 Clock Skew

Equation 6 uses timing at both the sender t_0^{k-1} and the receiver t_n^k, which suggests that clock skew could cause error in the calculation. However, we show here that any clock skew is canceled in the calculation.

We model clock skew as time measurements taken at the receiver to be offset by ϵ:

$$\hat{t}_n^k = t_n^k + \epsilon \quad \hat{d}^{n-1} = d^{n-1} + \epsilon$$

Where $\hat{t}$, $\hat{d}$, and $\hat{b}$ are the clock skewed versions of packet arrival time, cumulative link delay, and link bandwidth. The clock skewed version of (6) becomes

$$\hat{b}_{l_q} = \frac{s^{k-1}}{(\hat{t}_n^k + \frac{s^k - s^{k-1}}{b^{l_q-1}} - \frac{s^k}{b^{n-1}} - t_0^{k-1} - \hat{d}^{n-1})}$$

Substituting and simplifying,

$$\begin{aligned}\hat{b}_{l_q} &= \frac{s^{k-1}}{(t_n^k + \epsilon + \frac{s^k - s^{k-1}}{b^{l_q-1}} - \frac{s^k}{b^{n-1}} - t_0^{k-1} - d^{n-1} - \epsilon)} \\ &= \frac{s^{k-1}}{(t_n^k + \frac{s^k - s^{k-1}}{b^{l_q-1}} - \frac{s^k}{b^{n-1}} - t_0^{k-1} - d^{n-1})} \\ &= b_{l_q}\end{aligned}$$

So clock skew does not affect the calculation.

4.1.3 Using Round Trip Measurements

Although (6) uses timings at both the sender and the receiver, we can transform it to use only timings at the sender. The idea is to cause the receiver to send an acknowledgement for each tailgater packet that arrives. If we can correlate the tailgater and acknowledgement packets (described in the next section), then we can determine the round-trip properties of the path and use these to calculate the link bandwidths of the forward path.

We define S, T, Q, B, and D to be the packet size, packet arrival time, packet queueing delay, link bandwidth, and link delay of the flow in the reverse of the direction we are interested in. We start with measurements of the arrival times of packets at link 0, the round trip delay (d^{rt}), and the round trip bandwidth (b^{rt}) and derive a form of (6) containing only these variables.

We know that that the round trip delay is equal to the sum of the delay in the forward direction and the delay in the reverse direction and similarly for bandwidth:

$$d^{rt} = d^{n-1} + D^{n-1} \qquad \frac{1}{b^{rt}} = \frac{1}{b^{n-1}} + \frac{1}{B^{n-1}}$$
$$d^{n-1} = d^{rt} - D^{n-1} \qquad \frac{1}{b^{n-1}} = \frac{1}{b^{rt}} - \frac{1}{B^{n-1}}$$

Furthermore, using (3) and assuming that the acknowledgement is sent immediately when the tailgater arrives and experiences no queueing:

$$T_0^k = t_n^k + \sum_{i=0}^{n-1}\left(\frac{s^k}{b_i} + d_i\right)$$
$$t_n^k = T_0^k - \sum_{i=0}^{n-1}\left(\frac{s^k}{b_i} + d_i\right)$$

Substituting into (6) and simplifying,

$$b_{l_q} = \frac{s^{k-1}}{T_0^k - \frac{s^k - S^k}{B^{n-1}} + \frac{s^k - s^{k-1}}{b^{l_q-1}} - \frac{s^k}{b^{rt}} - t_0^{k-1} - d^{rt}}$$

We know all the terms of this equation except the inverse sum of the inverse reverse link bandwidths B^{n-1}. At this point, we assume that the bandwidths are symmetric ($B^{n-1} = \frac{b^{rt}}{2}$). To measure asymmetric links, we would have to be able to measure one way delay.

4.1.4 Causing Acknowledgements to be Sent

To get round trip measurements, we must cause the receiver to reply consistently to arriving packets. The `nettimer` implementation sends TCP FIN packets in both the sigma and tailgate phase. Hosts are required to respond to TCP FIN packets with a TCP RST (reset) packet. `Nettimer` uses this to measure the round-trip-delay. This is better than using ICMP packets because some routers and hosts block or rate-limit ICMP packets. Similarly, TCP is better than UDP because UDP packets sent to closed ports induce ICMP port unreachable messages which are rate-limited on some systems (e.g. Linux and Solaris).

4.1.5 Dropped Packets

A dropped tailgater is not a problem because it simply means one fewer delay samples. However, a source can only admit a tailgater timing sample if it also received the ICMP TTL-exceeded message for the corresponding tailgated packet. This is because the tailgated packet could have been dropped (perhaps because of congestion) before link l_q, and the tailgater packet could have traveled through the network unimpeded, thus giving an abnormally low minimum delay.

4.1.6 Dealing with Invisible Nodes

Finally, to combat the problem of invisible nodes mentioned in Section 2, `nettimer` gets timing information directly from the kernel using libpcap [12], rather than from the application level. This removes the application–kernel node from the measured path. In addition, we tried to extend the tailgating technique to measure the kernel–NIC node by causing select packets to be dropped in the NIC, but we could not find a way to do this across many different NIC drivers. As with the one-packet-based tools, bridges remain invisible nodes for `nettimer`.

4.2 Tailgating Analysis

In this section we list the advantages and disadvantages of the packet tailgating technique. The advantages of the technique are its speed, unobtrusiveness and robustness compared to the other link bandwidth techniques described in Section 2. The disadvantages of the technique are its need to send packets back-to-back on the first link, its inability to measure a very fast link after a very slow link, the fact that queuing anywhere along the path disrupts the measurement of all links on the path, and the accumulation of errors in the calculation.

Packet tailgating is potentially faster and less obtrusive than the previously discussed techniques because it performs the expensive linear regression step only once for the entire path instead of once for every link. The tailgating step has to be done for every link, but this only requires finding the minimum delay for a pair of packets compared to finding the minimum delay of 16 to 64 different packet sizes. We test this hypothesis using `nettimer` in the next section.

Packet tailgating is potentially more robust because it can detect multichannel links, does not rely on timely delivery of ICMP packets, and can be run without acknowledgements.

Packet tailgating can measure multi-channel links because, like the packet bunch extension to packet pair, it can send multiple packets to fill the channels. To do this, we send the tailgated packet followed by any number of tailgaters. If we send $c + 1$ packets where c is the number of channels, then we can cause the last tailgater to queue behind the tailgated packet. We can use this queueing to measure the bandwidth of one of the channels (we assume that all the channels have the same bandwidth). We determine the number of channels by sending variable numbers of tailgaters until we observe that sending $c+1$ packets results in significantly higher delay than sending c.

Unlike the techniques described in Section 2.2, packet tailgating does not use ICMP time-exceeded packets from intermediate nodes for measurement. Our `nettimer` implementation uses these packets for identifying routers and for determining which tailgated packets were prematurely dropped, but it does not rely on their timely delivery. This enables it to run faster in environments where these packets are rate-limited and to run more accurately in environments where they are delivered inconsistently.

In fact, packet tailgating can work without acknowledgments at all from the destination, although our current implementation does not have this feature. By deploying software at the destination host, we can measure the one-way delay of packets. The tailgating source can continue to send later tailgater packets without knowing the delay of earlier tailgater packets. If the earlier delays can be occasionally transmitted back to the source, then the source can adaptively decide when to finish the two stages, but this is an optimization. Otherwise, the tailgater source can just send a fixed number of packets in each stage. Eventually the source and destination must communicate so that the source can specify which tailgated packets were prematurely dropped.

Measuring without acknowledgements avoids queuing in the return path, enabling packet tailgating to be twice as accurate as single-packet techniques. In addition, measuring without acknowledgements avoids ack-implosion on multicast trees, enabling packet tailgating to measure the bandwidth of several links simultaneously on a multicast tree. Single packet techniques used on a multicast tree would cause a flood of acknowledgements to flow back to the source. The queueing of these acks would likely destroy their usefulness for round trip delay measurements.

Packet tailgating also has several limitations compared to previous techniques. The first is that the source must be able to send packets quickly on the first link. This limitation is quantified in the previous section.

The second limitation is that tailgating cannot measure a very fast link after a very slow link. The problem is that the tailgated packet may have finished transmitting on the faster link before the tailgater packet has finished transmitting on the slower link. The typical size of the tailgated and tailgater packets are 1500 and 40 bytes (with 1500 bytes being the largest unfragmented packet many paths will carry and 40 bytes being the smallest TCP packet due to IP header size). Therefore, this is only a problem when the ratio of bandwidths of the faster link to the slower link exceeds 1500/40 = 37.5.

Table 2: Program Versions: This table lists the versions of the programs we used.

Program	Version	Date
`pathchar`	alpha	April 21, 1997
`clink`	1.0	August 14, 1998
`pchar`	1.1.1	January 24, 2000
`nettimer`	1.0.6	May 30, 2000

The solution to this problem is to use the one-packet technique to measure the problematic link and use tailgating everywhere else. We split the path by performing the sigma phase three times: once to the node just before the very fast link, once to the node just after the very fast link, and once to the destination node. We know we should do this when the measured ratio of bandwidths of two adjacent links approaches 37.5. This solution increases the number of packets sent beyond what regular tailgating sends but should still send fewer packets than using a one-packet method over the entire path.

Another limitation is that queueing anywhere along the path disrupts the measurement of all the links. In contrast, for one-packet techniques, only queueing at earlier links affects the measurement of a link. This could be a significant disadvantage for tailgating if there is a very congested link far downstream along a path. It would prevent accurate and fast tailgating measurement of all the earlier links. The solution is similar to that for the previous limitation. We perform the sigma phase to the node just before the congested link and to the destination node. We then perform the tailgating phase to the pre-congestion node to measure all the links before the congested one.

The final limitation is that errors may accumulate during the calculation such that links that are very far away are unmeasurable. The one-packet techniques only propagate errors forward one link [8] so they are fairly robust with respect to errors. In our measurements in the next section, the accumulation of error is not noticeable in paths up to length 11. However, quantifying the exact error is part of our future work.

5. MEASUREMENTS

The goal of this experiment is to determine whether packet tailgating has accuracy comparable to previously known techniques with a significant reduction in the number of packets sent and received. For these measurements, we use a prototype implementation of `nettimer`. The results in this section are preliminary and are intended as a proof of concept and not an evaluation of the full potential of the technique.

We compare the results of the latest publicly available versions (listed in Table 2) of `pathchar` [7], `clink` [4], `pchar` [11], and `nettimer`. `pathchar`, `clink`, and `pchar` implement the one-packet technique described earlier. Although 1.0 is the latest publicly available version of Downey's `clink` program, it does not incorporate the adaptive algorithms described in [4]. However, those techniques are complementary with packet tailgating.

Table 3: Short Path: This table lists the results of running the link bandwidth programs on the short path. TTL is the distance of the link from the source. C is the number of channels in the link. BW/C is the actual physical bandwidth per channel. Columns 3-6 are bandwidths given in Mb/s.

TTL	C	BW/C	`pathchar`	`clink`	`pchar`	`nettimer`
1	1	10	7.9	7.8	7.9	6.7
2	1	100	34	35	34	62
3	1	100	31	32	32	21
4	1	100	9.6	7.9	7.9	38

Table 4: Network Load: This table lists the total number of packets transferred during the short and long path probes.

Program	Short Path Packets	Long Path Packets
`pathchar`	11562	31782
`clink`	6002	16400
`pchar`	11732	32417
`nettimer`	982	6663

Table 5: Long Path: This table lists the results of running the link bandwidth programs on the long path. TTL is the distance of the link from the source. C is the number of channels in the link. BW/C is the actual physical bandwidth per channel. Columns 3-6 are bandwidths given in Mb/s.

TTL	C	BW/C	`pathchar`	`clink`	`pchar`	`nettimer`
1	1	10	8.0	7.9	7.9	7.6
2	1	100	35	32	34	36
3	1	100	77	100	88	100
4	1	622	345	223	222	244
5	1	622	145	173	330	239
6	1	622	289	508	183	286
7	1	622	207	168	224	277
8	1	155	55	60	49	45
9	2	100	36	35	39	55
10	2	100	73	44	66	34
11	1	100	36	52	43	66

We ran each program using default settings from `tnt.stanford.edu` to `statistics.stanford.edu` (4 hops) and to `www.berkeley.edu` (11 hops). The short path has a RTT of 1.0ms and the long path has an RTT of 4.2ms. The actual number of channels of the links and their bandwidths are listed in the 2nd and 3rd columns of Table 3 and Table 5. We chose these paths because we know that both endpoints are well connected and that their administrators would be likely to tell us the link bandwidths of these links. In addition, both the one-packet and tailgating techniques can measure these paths because the destinations do not rate-limit or filter ICMP packets and there is no very slow link followed by a very fast link.

The measurement source node, `tnt.stanford.edu`, is an Intel Pentium II 266MHz with 256MB of main memory. It is running Redhat 6.1 with a GNU/Linux kernel 2.2.12-20. The results were gathered from 1:21 to 2:13 AM PST on May 30. We chose this time because there would be relatively little traffic in the network. We used `tcpdump` to determine the number of packets sent and received.

The short path results are summarized in Table 3 and Table 4. All of the programs have approximately the same accuracy, but `nettimer` uses an order of magnitude fewer packets than the others. In bandwidth measurements, `pathchar`, `clink`, and `pchar` are mostly in agreement, which is not surprising considering their measurement technique is similar. In particular, all of these programs under-estimate the last link on this path. This is because the destination host responds to probes with an ICMP port unreachable message which is up to 584 bytes. This is within the RFC standard because it sometimes allows the sender to make a more accurate correlation between the ICMP packet and the UDP packet that caused it. However, `pathchar`, `clink`, and `pchar` apparently do not take into account how much longer the large ICMP packet takes to travel back to the source, causing an under estimation of the bandwidth.

The long path results are summarized in Table 5 and Table 4. These results show that there is not significant accumulation of error in the tailgating implementation compared to the others. However, the results of all the programs deviate by as much as 68% from the nominal values, even though there was relatively little traffic during this time.

We believe that the four most likely contributors to error for all four programs are 1) unreachable nominal values, 2) errors in implementation, 3) noise, 4) timing resolution, and 5) multi-channel links.

Although some of the links have a nominal value of 622Mb/s, in practice a router may not actually be able to forward packets this quickly. We are attempting to investigate this issue through simulation.

Another possible source of error is the implementation. The problem with not accounting for large ICMP packets is an example of such an error. If three independent implementations of the same algorithm all have the same error, then there are likely even more errors in our singular implementation.

The problem with noise is that we are trying to measure differences in packet delays that are several orders of magnitude smaller than delays caused by other traffic. In our experiment, to measure the 622Mb/s links, we have to detect differences in delays of $\frac{1500\text{bytes}}{622\text{Mb/s}} = 19\mu s$. Just one packet queued at the wrong time could result in a delay of $\frac{1500\text{bytes}}{10\text{Mb/s}} = 1.2ms$. This is almost 100 times larger than the delay we are trying to measure. One solution is to deploy `nettimer` software at the destination instead of relying on TCP RST packets. This would eliminate the interference of noise along the reverse path. We are implementing this in our next version of `nettimer`.

The timing resolution of the host machine and operating system can also cause a large error in the bandwidth estimate. We measured the timing resolution of the source machine by sending small packets to the loopback interface and measuring the smallest non-zero inter-packet delay. This gives an upper bound on timing resolution. The timing resolution of `tnt.stanford.edu` is no worse than $20\mu s$. Nonetheless, this could cause us to measure a 622Mb/s link as a 414Mb/s link or a 1333Mb/s link. The GNU/Linux 2.3 kernel has a more efficient method of conveying packet timings to applications, and we hope to test this shortly.

Finally, while we are adapting `nettimer` to detect multi-channel links, the one-packet techniques will have difficulty measuring the multi-channel links 9 and 10 because of the inability of their fundamental model to consider such links.

6. CONCLUSION

As the Internet grows larger and more heterogenous, it becomes more important, but more difficult, for us to understand its most fundamental aspects. The ability to measure metrics such as link bandwidth is essential, but the power of measurement models and techniques must keep pace with the size and complexity of the Internet.

In this paper we present a new deterministic model of packet delay that unifies previous models. From this model, we derive a new technique, called packet tailgating, to measure link bandwidths along a path through the Internet. Preliminary measurements from our prototype implementation show that it places an order of magnitude less load on the network than previous measurement techniques while maintaining similar accuracy. The technique can theoretically measure multi-channel links, can be run on multicast trees, does not rely on consistent behavior of routers handling ICMP packets, and does not rely on timely delivery of acknowledgments. Ultimately, the tailgating technique depends only on the existence of store-and-forward routers, the IP TTL mechanism, and packet queueing.

We are adding several features to `nettimer`. First, we are enhancing the implementation to measure multi-channel links. Second, we are reducing its vulnerability to noise by using one-way packet delay measurements rather than acknowledgements, and by using finer-grained timing information. Finally, we are exploring its use on a variety of different types of links, including wireless links and DSL. The source code for `nettimer` is available via our web site: `http://www.stanford.edu/~laik/projects/nettimer/`.

7. ACKNOWLEDGMENTS

We thank Ed Swierk, Petros Maniatis, and Mema Roussopoulos as well as the anonymous SIGCOMM reviewers for their many helpful comments on this paper. Our thanks to Walter Willinger for his comments on the multi-packet model. Our thanks to Van Jacobson, Allen Downey, and Bruce Mah for providing `pathchar`, `clink`, and `pchar`, respectively. Special thanks to Steve Tingley, Charlie Orgish, Jay Kohn, and Ron Roberts of Stanford University and the anonymous system administrators at U.C. Berkeley for tracking down the nominal link bandwidths of our test paths. Finally, thanks to Vern Paxson for inspiring this work.

This work was supported by a gift from NTT Mobile Communications Network, Inc. (NTT DoCoMo). Additionally, Kevin Lai was supported in part by a USENIX Scholar Fellowship.

8. REFERENCES

[1] S. M. Bellovin. A Best-Case Network Performance Model. http://www.research.att.com/ smb/papers/netmeas.ps, 1992.

[2] J.-C. Bolot. End-to-End Packet Delay and Loss Behavior in the Internet. In *Proceedings of ACM SIGCOMM*, 1993.

[3] R. L. Carter and M. E. Crovella. Measuring Bottleneck Link Speed in Packet-Switched Networks. Technical Report BU-CS-96-006, Boston University, 1996.

[4] A. B. Downey. Using pathchar to Estimate Internet Link Characteristics. In *Proceedings of ACM SIGCOMM*, 1999.

[5] B. Efron and R. J. Tibshirani. *An Introductin to the Bootstrap*. Chapman and Hall, 1993.

[6] V. Jacobson. Congestion Avoidance and Control. In *Proceedings of ACM SIGCOMM*, 1988.

[7] V. Jacobson. pathchar. ftp://ftp.ee.lbl.gov/pathchar/, 1997.

[8] V. Jacobson. pathchar – a tool to infer characteristics of Internet paths. Presented at the Mathmatical Sciences Research Institute, 1997.

[9] S. Keshav. A Control-Theoretic Approach to Flow Control. In *Proceedings of ACM SIGCOMM*, 1991.

[10] K. Lai and M. Baker. Measuring Bandwidth. In *Proceedings of IEEE INFOCOM*, 1999.

[11] B. A. Mah. pchar. http://www.ca.sandia.gov/ bmah/Software/pchar/, 2000.

[12] S. McCanne and V. Jacobson. The BSD Packet Filter: A New Architecture for User-level Packet Capture. In *Proceedings of the 1993 Winter USENIX Technical Conference*, 1993.

[13] V. Paxson. End-to-End Routing Behavior in the Internet. In *Proceedings of ACM SIGCOMM*, 1996.

[14] V. Paxson. End-to-End Internet Packet Dynamics. In *Proceedings of ACM SIGCOMM*, 1997.

[15] V. Paxson. *Measurements and Analysis of End-to-End Internet Dynamics*. PhD thesis, University of California, Berkeley, April 1997.

[16] S. Savage. Sting: a TCP-based Network Measurement Tool. In *Proceedings of the USENIX Symposium on Internet Technologies and Systems*, 1999.

[17] S. Seshan, M. Stemm, and R. Katz. SPAND: Shared Passive Network Performance Discovery. In *Proceedings of the USENIX Symposium on Internet Technologies and Systems*, 1997.

[18] I. Stoica and H. Zhang. Providing guaranteed services without per flow management. In *Proceedings of ACM SIGCOMM*, 1999.

APPENDIX

A. PACKET PAIR DERIVATION

Before starting the derivation, we define and derive two lemmas to keep the main derivation clearer. The lemmas have no other conceptual significance.

LEMMA A.1. *Let* $s^0 = s^1$, $t_0^0 = t_0^1$, $0 \le j \le n$. *Then*

$$t_j^1 - t_j^0 = \sum_{i=0}^{j-1} \max\left(0, t_{i+1}^0 - d_i - t_i^1\right)$$

Proof.

$$\begin{aligned} & t_j^1 - t_j^0 = \\ & = \sum_{i=0}^{j-1} \left(\frac{s^1}{b_i} + d_i + \max\left(0, t_{i+1}^0 - d_i - t_i^1\right) \right) + \\ & \quad t_0^1 - \left(\sum_{i=0}^{j-1} \left(\frac{s^0}{b_i} + d_i \right) + t_0^0 \right) \\ & \quad \text{(From Equation 5)} \\ & = \sum_{i=0}^{j-1} \max\left(0, t_{i+1}^0 - d_i - t_i^1\right) \end{aligned}$$

□

LEMMA A.2. *Let* $s^0 = s^1$, $t_0^0 = t_0^1$, $0 \le j \le n$. *Then*

$$t_{j+1}^0 - d_j - t_j^1 = \frac{s^0}{b_j} - \sum_{i=0}^{j-1} \max\left(0, t_{i+1}^0 - d_i - t_i^1\right)$$

Proof.

$$\begin{aligned} & t_{j+1}^0 - d_j - t_j^1 = \\ & = \sum_{i=0}^{j} \left(\frac{s^0}{b_i} + d_i \right) + t_0^0 - d_j - \\ & \quad \left(\sum_{i=0}^{j-1} \left(\frac{s^1}{b_i} + d_i + \max\left(0, t_{i+1}^0 - d_i - t_i^1\right) \right) + t_0^1 \right) \\ & \quad \text{(From Equation 5)} \end{aligned}$$

$$= \sum_{i=0}^{j-1} \left(\frac{s^0}{b_i} + d_i \right) + \frac{s^0}{b_j} + d_j - d_j - \left(\sum_{i=0}^{j-1} \left(\frac{s^1}{b_i} + d_i \right) + \sum_{i=0}^{j-1} \max\left(0, t_{i+1}^0 - d_i - t_i^1\right) \right)$$

$$= \frac{s^0}{b_j} - \sum_{i=0}^{j-1} \max\left(0, t_{i+1}^0 - d_i - t_i^1\right)$$

□

We state the packet pair property and then derive it:

THEOREM A.1 (PACKET PAIR PROPERTY). *Let $b_{min(l)} \leq b_i, (\forall i,\ 0 \leq i \leq l)$, then if we send two packets of the same size ($s^0 = s^1$) at the same time ($t_0^0 = t_0^1$), they will arrive with a difference in time equal to the size of the second packet divided by the smallest bandwidth on the path ($t_n^1 - t_n^0 = \frac{s^1}{b_{min(n-1)}}$).*

Proof. We perform induction on n, the number of links. For $n = 1$, we substitute using Equation 5:

$$t_n^1 - t_n^0 = t_1^1 - t_1^0$$

$$= \sum_{i=0}^{0} \left(\frac{s^1}{b_i} + d_i + \max(0, t_{i+1}^0 - d_i - t_i^1) \right) + t_0^1 - \left(\sum_{i=0}^{0} \left(\frac{s^0}{b_i} + d_i \right) + t_0^0 \right)$$

We continue simplifying and substitute again using Equation 5:

$$= \frac{s^1}{b_0} + d_0 + \max(0, t_1^0 - d_0 - t_0^1) + t_0^1 - \left(\frac{s^0}{b_0} + d_0 + t_0^0 \right)$$

$$= \max(0, t_1^0 - d_0 - t_0^1)$$

$$= \max(0, \sum_{i=0}^{0} \left(\frac{s^0}{b_i} + d_i \right) + t_0^0 - d_0 - t_0^1)$$

$$= \max(0, \frac{s^1}{b_0} + d_0 - d_0)$$

$$= \frac{s^1}{b_0}$$

$$= \frac{s^1}{b_{min(n-1)}}$$

This proves the $n = 1$ case. For $n > 1$, we start by using Lemma A.1:

$$t_n^1 - t_n^0 =$$

$$= \sum_{i=0}^{j-1} \max(0, t_{i+1}^0 - d_i - t_i^1)$$

We simplify further and apply the inductive hypothesis:

$$= \sum_{i=0}^{j-2} \left(\max(0, t_{i+1}^0 - d_i - t_i^1) \right) + \max(0, t_n^0 - d_{n-1} - t_{n-1}^1)$$

$$= \frac{s^0}{b_{min(n-2)}} + \max(0, t_n^0 - d_{n-1} - t_{n-1}^1)$$

We apply Lemma A.2:

$$= \frac{s^0}{b_{min(n-2)}} + \max\left(0, \frac{s^0}{b_{n-1}} - \sum_{i=0}^{n-2} \max(0, t_{i+1}^0 - d_i - t_i^1)\right)$$

We apply Lemma A.1:

$$= \frac{s^0}{b_{min(n-2)}} + \max\left(0, \frac{s^0}{b_{n-1}} - t_{n-1}^1 - t_{n-1}^0)\right)$$

We apply the inductive hypothesis again and simplify:

$$= \frac{s^0}{b_{min(n-2)}} + \max\left(0, \frac{s^0}{b_{n-1}} - \frac{s^0}{b_{min(n-2)}}\right)$$

$$= \max\left(\frac{s^0}{b_{min(n-2)}}, \frac{s^0}{b_{n-1}}\right)$$

There are two possibilities at this point. One possibility:

$$\frac{s^0}{b_{min(n-2)}} \geq \frac{s^0}{b_{n-1}}$$

$$b_{n-1} \geq b_{min(n-2)} \tag{7}$$

$$t_n^1 - t_n^0 = \frac{s^0}{b_{min(n-2)}}$$

$$= \frac{s^1}{b_{min(n-1)}} \quad \text{(From (7) and } s^0 = s^1\text{)}$$

The other possibility:

$$\frac{s^0}{b_{min(n-2)}} < \frac{s^0}{b_{n-1}}$$

$$b_{n-1} < b_{min(n-2)} \tag{8}$$

$$t_n^1 - t_n^0 = \frac{s^0}{b_{n-1}}$$

$$= \frac{s^1}{b_{min(n-1)}} \quad \text{(From (8) and } s^0 = s^1\text{)}$$