Measuring Bottleneck Bandwidth of Targeted Path Segments*

Khaled Harfoush harfoush@cs.bu.edu
Azer Bestavros best@cs.bu.edu
John Byers byers@cs.bu.edu

Computer Science Department
Boston University
Boston, MA 02215

Technical Report
BUCS-2001-016

July 31, 2001

Abstract

Accurate measurement of network bandwidth is crucial for flexible Internet applications and protocols which actively manage and dynamically adapt to changing utilization of network resources. These applications must do so to perform tasks such as distributing and delivering high-bandwidth media, scheduling service requests and performing admission control. Extensive work has focused on two approaches to measuring bandwidth: measuring it hop-by-hop, and measuring it end-to-end along a path. Unfortunately, best-practice techniques for the former are inefficient and techniques for the latter are only able to observe bottlenecks visible at end-to-end scope. In this paper, we develop and simulate end-to-end probing methods which can measure bottleneck bandwidth along *arbitrary, targeted subpaths* of a path in the network, including subpaths *shared* by a set of flows. As another important contribution, we describe a number of practical applications which we foresee as standing to benefit from solutions to this problem, especially in emerging, flexible network architectures such as overlay networks, ad-hoc networks, peer-to-peer architectures and massively accessed content servers.

Keywords: End-to-end measurement; packet-pair; bottleneck bandwidth; overlay networks; content distribution.

*This work was partially supported by NSF research grants ANIR-9986397 and CAREER ANIR-0093296.

1 Introduction

Measurement of network bandwidth is crucial for many Internet applications and protocols, especially those involving the transfer of large amounts of data and those involving the delivery of content with real-time QoS constraints, such as streaming media. Examples of the importance of bandwidth estimation include request routing protocols in Content Distribution Networks (CDNs) [1] or in Peer-to-Peer (P2P) networks [27], network-aware cache/replica placement and maintenance policies [15, 25], flow scheduling and admission control policies at massively-accessed content servers [8], end-system multicast and overlay network reconfiguration protocols [7, 14, 2], among many others.

Bottleneck Bandwidth Measurement: We use the term *base bandwidth* of a link to refer to the transmission rate of that link. We use the term *bottleneck bandwidth* of a sequence of links (or path) to refer to the maximum transmission rate that could be achieved between two hosts at the endpoints of that path in the absence of any competing traffic. The bottleneck bandwidth of a path is limited by the minimum base bandwidth (i.e. slowest) of all links along that path. The presence of competing traffic along a path can also limit the transmission rate for that path. We use the term *available bandwidth* to refer to the portion of the bottleneck bandwidth that is not claimed by competing traffic.

There are many applications that require the measurement of bottleneck bandwidth or available bandwidth between two end-points. Due to the highly-variable nature of cross-traffic on the Internet, measurement of available bandwidth is likely to be effective only at short time-scales (e.g. few round-trip times). Thus, estimating available bandwidth (whether directly or indirectly) is only valuable for control (or optimization) of processes operating at fairly short time-scales (e.g. size of TCP congestion window), as opposed to those processes operating at longer time scales (e.g. server selection or admission control). For these latter processes, the use of bottleneck bandwidth may be more appropriate.

Current bandwidth measurement techniques can be classified into two broad categories: (1) Link-level bandwidth estimation techniches, and (2) End-to-End bandwidth estimation techniques. Link-level techniques incrementally infer the bandwidth of individual physical links along a path [12, 20, 9]. While such path characterization can be used to yield the bottleneck bandwidth shared between multiple paths, it is rather inefficient and unscalable, due to the need for large numbers of probes. End-to-end techniques infer the bandwidth of the bottleneck link along a path [16, 4, 6] through the use of end-to-end probing. These techniques, though efficient, do not have the capability to identify bottlenecks other than those observable at end-to-end scope. Thus, they cannot determine the bottleneck bandwidth over an arbitrary subpath, nor over a subpath *shared* by a given set of connections. Such inferences are crucial to effective construction of ad-hoc networks which arise in peer-to-peer communication settings, dynamic overlay networks, end-system multicast and content delivery, as we illustrate next.

Catalyst Applications: To exemplify the type of applications that can be leveraged by the identification of shared bottleneck bandwidth (or more generally, the bottleneck bandwidth of an arbitrary, targeted subpath), we consider the two scenarios illustrated in Figure 1. In the first

scenario, a client must select two out of three servers to use to download data in parallel. This scenario may arise when downloading content in parallel from a subset of mirror sites [5, 26], from a subset of multicast sessions [28], or from a subset of peer nodes in P2P environments. In the second scenario, an overlay network must be set up between a single source and two destinations. This scenario may arise in ad-hoc networks and end-system multicast systems [7, 14].

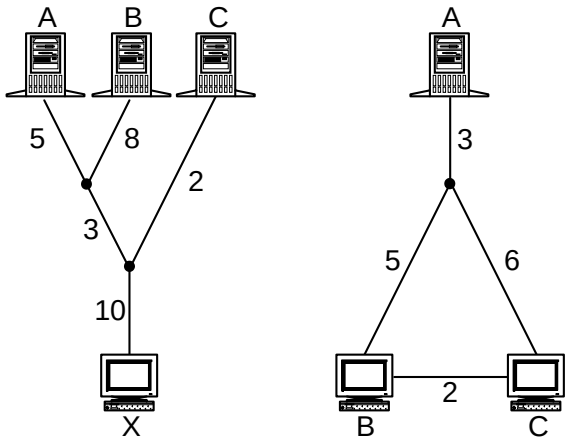

Figure 1: Leveraging shared bandwidth measurement for optimizing parallel downloads (left) and overlay network organization (right). Numeric labels represent bottleneck bandwidth of path segments in Mbps.

For the first scenario, illustrated in Figure 1 (left), the greedy approach of selecting the two servers whose paths to the client have the highest end-to-end bottleneck bandwidth—namely, servers A and B—is not optimal, since the aggregate bandwidth to the client would be limited by the shared 3Mbps bottleneck bandwidth from servers A and B to the client. To be able to select the pair of servers yielding the maximum *aggregate* bandwidth of 5Mbps—namely A and C or B and C—the client needs to measure the shared bottleneck bandwidth between pairs of servers. Similarily, in the second scenario illustrated in Figure 1 (right), the identification of the best set of routes for distributing content from source A to destinations B and C hinges on our ability to determine the bottleneck bandwidth of the shared portion of the AB and AC paths (as well as the end-to-end bottleneck bandwidth of path BC). Specifically, it is better to use the AB + BC links to provide 3Mbps to client B and 2Mbps to client C, rather than the AB + AC links for 1.5Mbps to each (assuming fair sharing).

Paper Scope, Contributions, and Organization: In this paper we propose an *efficient* end-to-end measurement technique that yields the bottleneck bandwidth of any *path segment* defined by routes between a set of end-points. By a path segment, we mean a sequence of network links between two *identifiable* nodes on that path. A node i on a path between a source s and a destination d is identifiable if it is possible to coerce a packet injected at the source s to exit the path at node i. One can coerce a packet to exit the path at node i by (1) simply targeting the packet to i (if the IP address of i is known), or (2) forcing the packet to stop at i through the use of TTL field (if the hopcount from s to i is known), or (3) by targeting the packet to a destination d', such that the paths from s

to d and from s to d' diverge at node i.

In prior work [3], we have shown that inference and (at least partial) labeling of a general metric-induced logical topology consisting of path segments between a sender and a set of receivers hinges on our ability to measure the value of the metric of interest for the shared portion of paths connecting the sender to any two receivers. Thus, in this paper we focus on this problem, for the metric of bottleneck bandwidth. Specifically, we present a novel probing technique that enables efficient end-to-end inference of shared bottleneck bandwidth and we present extensive simulation results that demonstrate the effectiveness of our proposed technique.

The remainder of this paper is organized as follows. In Section 2, we review existing literature and related work. In Section 3, we develop a basic probing toolkit, comprising existing methods and our new ideas. We then compose several of these tools together in Sections 4 and 5 to measure bottleneck bandwidth along arbitrary subpaths, and bottleneck bandwidth shared by a set of flows, respectively. In Section 6, we present results of extensive simulation experiments that we have conducted to show the effectiveness of our proposed techniques. We conclude with a summary in Section 7.

2 Related Work

As noted in the introduction, one way of classifying bandwidth estimation techniques is based on whether they are link-level [12, 20, 9] or end-to-end [16, 4, 6] techniques. Link-level techniques rely on incrementally probing routers along a path and timing their ICMP replies, whereas end-to-end techniques base their bandwidth estimation on end-host replies only. The techniques we present in this paper belong to this latter class, albeit at a granularity finer than that achievable using exisiting end-to-end techniques.

Another classification of bandwidth measurement techniques would be based on whether they measure the bottleneck bandwidth [12, 20, 9, 4, 16, 6, 18, 19] or the available bandwidth [6] of a path. The techniques we present in this paper are aimed at measuring bottleneck bandwidth.

In classifying bandwidth measurement techniques, one can also look at the probing methodology employed—namely, the number and sizes of packets in a probe. Probe structures considered in the literature include: (1) *single packet* probing [12, 20, 9], (2) *packet bunch* probing, employing a group of packets sent back-to-back [6], (3) *uniform packet-pair* probing, employing two back-to-back packets of the same size [4, 16, 6], and (4) *non-uniform packet-pair* probing, employing two back-to-back packets of different sizes [18, 19]. The probing techniques we will propose can be classified as packet-bunch probes with non-uniform packet sizes.

Finally, one can also classify bandwidth estimation techniques into *active* techniques and *passive* techniques. Active techniques, comprising most of the work in the literature, send probes for the sole purpose of bandwidth measurement. Passive techniques rely on data packets for probing as exemplified in Lai and Baker's *nettimer* tool [19], which uses a packet-pair technique at the transport level to passively estimate bottleneck link bandwidth. The techniques we propose for inferring bottleneck bandwidth are applied actively.

The probing constructions most closely related to ours are the "packet-pair" [17] and "tailgating" [18] constructions. We discuss the properties of these constructions and how they relate to ours in Section 3.

A key difference between these techniques and ours is that while packet-pairs aim at estimating end-to-end bottleneck bandwidth, and tailgating probes aim at estimating the base bandwidth of physical links; our constructions aim at estimating the bottleneck bandwidth along arbitrary subpaths.

3 Probing Toolkit

In this section, we start by describing basic constructs of our probing sequences and corresponding terminology. With each probing construct, we describe its properties and point to its usefulness as a building block for end-to-end measurement of subpath bottleneck bandwidth. Proofs of all lemmas presented in this section are available in the appendix.

3.1 Basic Definitions

For the purposes of this section, a *probe* is a sequence of one or more packets transmitted from a common origin. We say that any contiguous subsequence of packets within a probe are transmitted *back-to-back* if there is no time separation between transmission of the individual packets within the subsequence. As detailed in the related work section, back-to-back packets have been widely used in estimating the end-to-end bandwidth of a connection [4, 16, 6, 19, 18]. A *multi-destination* probe is one in which the constituent packets of the probe do not all target the same destination IP address. Multi-destination probes have begun to see wider use as emulations of notional multicast packets — many of the same end-to-end inferences that can be made with multicast packets can be made with multi-destination unicast probes (albeit with added complexity) [11, 10]. A *uniform* probe is one in which all of the constituent packets are of the same size; likewise, a *non-uniform* probe consists of packets of different sizes. Finally, we say that an individual packet is *hop-limited* if its TTL is set to an artificially small value so as not to reach the ostensible destination. Hop-limited packets can be used to trigger an ICMP response from an intermediate router and in other ways we describe momentarily.

Throughout the paper we use various probing techniques that rely on sending sequences of probes. The probing techniques differ in the number of packets constituting a probe, the size and the path traversed by each probe packet. They also differ in the host collecting the probing responses and the function used by this host to perform the required estimation.

Each packet p that is transmitted within a probe is parameterized by its size $s(p)$ in bytes and its final destination, $D(p)$. In the event that a packet is hop-limited, it has a third parameter, its maximum hop-count, $h(p)$. To denote a probe, we refer to each probe packet with a distinct lowercase letter, and represent the sequential order in which they are transmitted from the probing host by writing them from left to right. Finally, we denote interpacket spacing with square braces. As an example, $[pq][pq][r]$ would denote transmission of a pair of identical two-packet probes followed by a single packet probe which has different characteristics.

We use the term *interarrival time of packets p and q at a link* to denote the time elapsed between the arrival of the last byte of p and the arrival of the last byte of q at that link. Similarily, we use the term *interdeparture time* to denote the time elapsed between the transmission of the last byte of p and the transmission of the last byte of q. By these definitions, the interarrival time of packets p and q at a given link s the same as the interdeparture time of packets p and q at the preceding link on the path.

3.2 Existing Probing Methods and Properties

One of the essential techniques in our constructions is the use of "packet-pair" techniques, originally used by Keshav [17], and subsequently refined by Carter and Crovella [6], Paxson [22, 24, 23] and Lai and Baker [19], to determine bottleneck bandwidth on a network path.

Packet-pair techniques rely on the following property, which holds under an assumed network model which we specify momentarily.

Lemma 1 PACKET-PAIR PROPERTY. *Consider a path of n physical links $L_1, L_2, \ldots, L_n$ with base bandwidths $b_1, b_2, \ldots, b_n$ respectively. If a probe of the form [pp] is injected at L_1, with $D(p) = L_n$, then the interarrival time of the two constituent packets of this probe at L_n is $\frac{s(p)}{\min_k b_k}$ units of time.*

An important corollary to Lemma 1 is that the bottleneck bandwidth across a set of links ($\min_k b_k$) can be estimated through measurement of packet interarrival times and knowledge of packet sizes.

Another closely related technique also used in our constructions is the "packet-tailgating" probing technique. This technique was introduced by Lai and Baker in [18] and evaluated within their *nettimer* tool [19] to estimate the bottleneck bandwidth of all physical links along a path. The packet-tailgating technique hinges on the following property (implicit in [19]), which formulates the condition under which a non-uniform packet-pair remains back-to-back over a sequence of physical links.

Lemma 2 TAILGATING PROPERTY. *Consider a path of n physical links $L_1, L_2, \ldots, L_n$ with base bandwidths $b_1, b_2, \ldots, b_n$ respectively. If a probe of the form [pq] is injected at L_1, with $D(p) = D(q) = L_n$ and if $\forall k \leq n$, $\frac{s(p)}{s(q)} \geq \frac{b_{k+1}}{b_k}$, then [pq] will remain back-to-back along the entire path.*

The two basic properties spelled out in Lemmas 1 and 2, as well as the constructions and analyses we present later in this paper, are conditioned on a set of basic assumptions about the network. These assumptions (common to most probing studies—e.g., [4, 16, 6, 18, 19]) are:

1. Routers are store-and-forward and use FIFO queueing.
2. It is possible for a probing host to inject back-to-back packets into the network.
3. Host clock resolution is granular enough to enable accurate timing measurements.
4. Analytic derivations assume an environment free from cross-traffic.

Assumption 1 is needed to ensure that probe packet orderings are preserved. Assumptions 2 and 3 are easily enforcable using proper kernel capabilities. Assumption 4, while necessary for analysis, is typically relaxed in experimental (simulation or implementation) settings to establish the robustness of the constructions in realistic settings.

3.3 Paced Probes

We now describe the first of our constructions, a probe which is activated at a particular location *inside* the network. Such "remote" activation is made possible by use of large *pacer* packets, which lead the probe into the network, as detailed below.

Definition 1 *A* paced probe *is a probe X sent back-to-back behind a large* pacer packet *p of the form $[pX]$. The pacer packet has a destination $D(p)$ at an intermediate point in the network. It leads the paced probe (its* followers*) up through this link as part of their trip and all the followers queue behind (are* paced by*) p in the queue at router $D(p)$. At this point, p is dropped and the probe X is activated.*

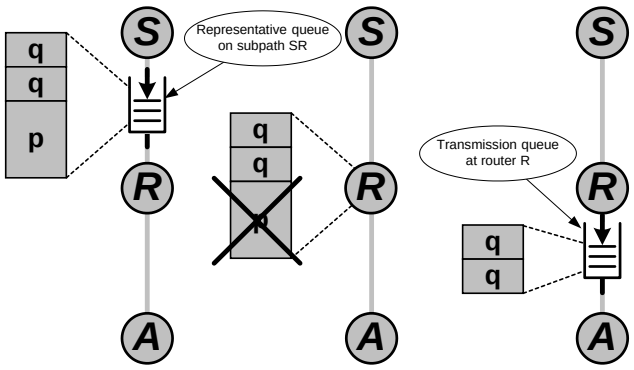

Figure 2: Pacer packet p precedes packet-pair $[qq]$ from S to router R (left), where p is dropped (middle), ensuring that the packet-pair are transmitted back-to-back from R towards A (right), enabling the measurement of the bottleneck bandwidth of subpath from R to A.

The primary intuition behind the use of pacer packets is that they allow *delayed release* of the probe which they precede. For example, in the picture depicted in Figure 2, a packet-pair (used to measure bottleneck bandwidth) destined for node A, is led out by a pacer packet destined for router R. Using our notation, we would denote such a probe by $[pqq]$, with $D(p) = R$ and $D(q) = A$ and where $s(p) >> s(q)$.[1] When the pacer drops out after node R, the packet-pair remains back-to-back, thus can be used to measure the bottleneck bandwidth on the remainder of the path. Guaranteeing that the probe remains back-to-back at the pacer packet's final destination is the key to the success of the technique, as expressed by the following lemma:

[1] Alternatively, we could make p a hop-limited probe, whereby $D(p) = R$, but $h(p)$ is set to the hop distance between S and A.

Lemma 3 *Let L be a sequence of n physical links L_1, L_2, $\ldots L_n$ with capacity bandwidths b_1, b_2, $\ldots b_n$ respectively. Also, let $[pq_1 \ldots q_m]$ be a probe consisting of a set of m paced packets which are injected back-to-back behind a pacer packet, where $D(p) = L_k$ and $D(q_i) = L_n$. A sufficient condition for all follower packets q_i to queue behind p at link L_k is*

$$\min_{1 \le w \le m} \left(\frac{s(p) + \sum_{j=1}^{w-1} s(q_j)}{\sum_{j=1}^{w} s(q_j)} \right) \ge \frac{b_k}{\min_{i \le k} b_i}$$

Using pacing, we are able to (effectively) inject a probe into an internal link with the guarantee that the constituent packets are not separated (e.g. no separation between back-to-back packet pairs). To some extent, pacing gives us the illusion of "bypassing" an uninteresting prefix of the path (namely $L_1, L_2, \ldots, L_{i-1}$) and "depositing" the probe at the first link of the subpath of interest (namely L_i).

3.4 Preserving Packet Interarrival Times Over a Subpath

The next challenge we tackle is to some extent complementary to pacing—namely how we can ensure that the traversal of links following the subpath of interest does not alter the measurements we intend to perform as a result of probing. The specific measurement that we must preserve is the interarrival/interdeparture time of packets at the last link on the subpath of interest (namely, L_i). The following Lemma establishes a necessary condition for the preservation of packet interarrival time over a sequence of links.

Lemma 4 *Consider a path of n physical links $L_1, L_2, \ldots, L_n$ with base bandwidths $b_1, b_2, \ldots, b_n$ respectively. If a probe of the form [p][p] is injected at L_1 with $D(p) = L_n$ and interarrival time of Δ, then Δ will be preserved over all links L_i if and only if $\frac{s(p)}{\Delta} \le \min_{1 \le k \le n} b_k$.*

Lemma 4 shows that in order to avoid skewing the interarrival time (Δ_i) through subpath L_{i+1}, $\ldots L_n$, the condition $\frac{s(p)}{\Delta_i} \le \min_{(i+1) \le k \le n} b_k$ need to be true.

4 Cartouche Probes

We are now ready to begin integrating the building blocks described in the preceding section to compose the bottleneck bandwidth probing technique which we call *Cartouche* probing. To recap our goal in this section, we are given a set of links $L_1, \ldots, L_n$ with base bandwidths $b_1, \ldots, b_n$ and we must estimate the bottleneck bandwidth of arbitrary subintervals of these links, i.e. estimate $\min_{i \le k \le j} b_k$, for arbitrary i and j such that $i \le j \le n$. We will use the conventional shorthand $b_{i,j}$ to denote the bottleneck bandwidth in the interval between links i and j inclusive.

4.1 Estimating Bandwidth Over a Prefix of the Path

We motivate the general technique for estimating the bottleneck bandwidth of a subpath by first

describing how to infer $b_{1,j}$, i.e., solving the easier problem of estimating bottleneck bandwidth over an arbitrary *prefix* of an entire path.

Since the packet-pair technique described earlier provides an estimate for $b_{1,n}$, it follows that if $b_{1,j} \leq b_{j+1,n}$, then $b_{1,j} = b_{1,n}$, giving us a solution. But when $b_{1,j} > b_{j+1,n}$, the packet-pair technique will end up estimating $b_{j+1,n}$. The underlying reason for this is that packet-pair techniques rely on the preservation of packet interarrival times induced at the bottleneck. So while the packet-pair property gives an interarrival time Δ_j at L_j of $\Delta_j = \frac{s(p)}{b_{1,j}}$, the interarrival time at L_n is $\Delta_n = \frac{s(p)}{b_{1,n}}$. This suggests a potential solution, namely preserving Δ_j unaltered to the end-host so that it can then actually infer $b_{1,j}$. This is precisely the functionality we considered in Section 3.4.

Lemma 4 gave us the condition we must satisfy to ensure the preservation of the interarrival time Δ_j between the two packets of a packet-pair at an internal link L_j. To guarantee this condition, we need to generalize the packet-pair construction (spelled out in Lemma 1) to yield an interarrival time that is large enough to satisfy the constraints set by Lemma 4. We do so next.

Lemma 5 *Consider a path of n physical links L_1, L_2, ... L_n with capacity bandwidths b_1, b_2, ... b_n respectively. If a probe of the form $[\{p\}^{(r+1)}]$ is injected at L_1 and destined towards L_n then the interarrival time (Δ_j) between the first and the last probe packets at the end of every physical link L_j, for $1 \leq j \leq n$, is $\frac{r \cdot s(p)}{\min_{1 \leq k \leq j} b_k}$.*

Based on the above lemma, one can generalize the packet-pair technique by using a probe structure consisting of a sequence of packets of the same size, whereby all packets except the first and the last are dropped at the end of L_j. By including enough packets in this sequence, the interarrival time between the first and last packets Δ_j at the end of L_j can be made large enough to be preserved as these two packets traverse links L_{j+1}, ... L_n. Indeed, Lemma 5 shows that if we use $r+1$ packets, then Δ_j would be $\frac{r \cdot s(p)}{b_{1,j}}$. To satisfy the packet interarrival preservation condition, it turns out that $r \geq \frac{b_{1,j}}{b_{j+1,n}} \leq r$. That is, we would need as many probe packets as the ratio between $b_{1,j}$ and $b_{j+1,n}$; which makes this approach impractical.

A better approach to propagating the interarrival times of probe packets at an internal link L_j through subsequent links $L_{j+1} \ldots L_n$ is to use small packets as "markers". Small packets have lower transmission delays and thus a lower probability of their interarrival times being altered. Using this improved idea, we are now ready to present a basic probing structure that incorporates all the features needed for an end-to-end inference of the bottleneck bandwidth of a path prefix.

Definition 2 *A* cartouche *probe $[pm\{pq\}^{r-1}pm]$ over the set of links $L_1, \ldots L_j, \ldots, A$ is a sequence of $r+1$ heterogenous packet-pairs in which $s(p) > s(m) = s(q)$, $D(p) = D(q) = L_j$, $D(m) = A$. We refer to the first packet (p) in each pair as the* magnifier *packet, the second packet (m or q) in each pair as the* marker *packet, link L_j as the* egress *link of the cartouche and r as the* cartouche dimension.

Figure 3 shows the composition and progression of an r-dimensional cartouche probe injected at link L_i towards end-host A with link L_j as its egress link.

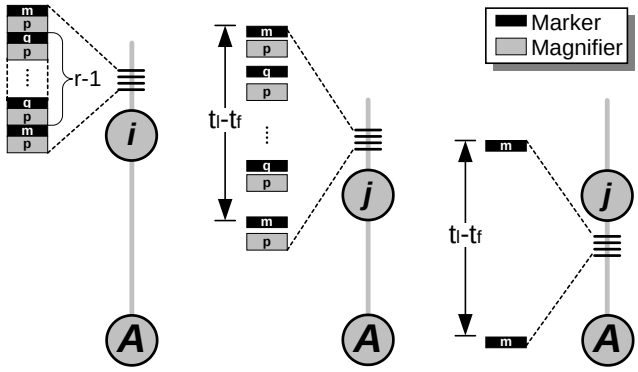

Figure 3: An r-dimensional cartouche consisting of back-to-back packet pairs of the form $[pm\{pq\}^{r-1}pm]$ injected at L_i (left). Cartouche constituents are spread out over time until they arrive at L_j (middle), where only the first and last markers continue on to destination A (right) with an interarrival time $t_\ell - t_f = \frac{r(s(p)+s(m))}{\min_{1\leq k\leq j} b_k}$ as detailed in Lemma 6.

Lemma 6 *Let L be a sequence of n physical links L_1, L_2, ...L_n with base bandwidths b_1, b_2 , ...b_n respectively. Given a cartouche of the form $[pm\{pq\}^{r-1}pm]$ over L with L_j as its egress link, let t_f and t_ℓ be the time that the final byte of the first and last marker packets are received at link L_j, respectively, then $t_\ell - t_f = \frac{r(s(p)+s(m))}{\min_{1\leq k\leq j} b_k}$.*

Lemma 6 provides the most important property of cartouche probe packets. It defines the interarrival time for the first and last marker packets over every physical link up to the cartouche egress link.

Corollary 1 *Let L be a sequence of n physical links L_1, L_2, ...L_n with base bandwidths b_1, b_2 , ...b_n, respectively. Given a cartouche of the form $[pm\{pq\}^{r-1}pm]$ over L with L_j as its egress link, let Δ_j be the interarrival time between the m markers at the end of L_j, then Δ_j will be preserved over L_{j+1}, ...L_n if and only if $\frac{b_{1,j}}{b_{j+1,n}} \leq \frac{r(s(p)+s(m))}{s(m)}$*

Corollary 1 follows directly from Lemma 6 and Lemma 3 to derive a sufficient condition for the preservation of markers interarrival times upon exit from the cartouche probe egress link L_j and throughout the sequence L_{j+1}, ...L_n. Note that with $s(p) = 1500$ bytes and $s(m) = 40$ bytes[2], preservation holds even when $\frac{b_{1,j}}{b_{j+1,n}} \leq 38.5r$; that is the interarrival time between the first and last marker packets holds even when $b_{j+1,n}$ is approximately $40r$ times smaller than $b_{1,j}$, where r is the cartouche dimension.

Lemma 6 and Corollary 1 are all that are needed to provide a solution to the problem of infering the bottleneck bandwidth of a path prefix. Specifically, this is done by: (1) dimensioning a cartouche probe to satisfy the conditions of Corollary 1, (2) setting the cartouche egress link to be L_j, (3)

[2]We motivate this particular choice later.

injecting the cartouche probe packets back-to-back at link L_1, and (4) using the interarrival time of the first and last marker packets at link L_n as an estimate of their interarrival time at link L_j and using the relationship given in Lemma 6 to estimate $b_{1,j}$.

4.2 Estimating Bandwidth over an Arbitrary Subpath

Now that we have presented our Cartouche probing technique to infer $b_{1,j}$, what do we need to do to infer $b_{i,j}$, for $1 \leq i \leq j \leq n$? Obviously, what we need to do is to have the cartouche probe "bypass" links $L_1, L_2, \ldots, L_{i-1}$ and be released at link L_i, the first link on the subpath of interest. This functionality is precisely what *paced* probes defined earlier are able to provide.

According to Lemma 3, a pacer probe packet could be used to guarantee cartouche pacing up through an arbitrary link L_i. Specifically, in order to guarantee that cartouche probe packets will be paced to L_i we need to know the base bandwidth of L_i (b_i) and the bottleneck bandwidth of the path L_1, L_2, $\ldots L_i(\min_{k \leq i} b_k)$. By fixing the pacer packet size to be the maximum IP packet size which will not be subject to IP fragmentation (in our experiments, we typically assume 1500 bytes, the Ethernet MTU), and the size of the cartouche marker packets to be the size of the smallest possible packet (we typically assume 40 bytes, the combined size of TCP/IP headers with no options)[3], then for any particular cartouche dimension we can solve the condition of Lemma 3 to obtain the largest possible packet size we can use for cartouche magnifier packets that guarantees queueing at L_i.

Thus, using a pacer packet preceding the cartouche, i.e. a probe of the form $[apm\{pq\}^{r-1}pm]$, where $D(a) = L_i$ and $s(a)$ is chosen sufficiently large to satisfy the condition in Lemma 3 will allow the measurement of $b_{i,j}$. The minimum pacer packet size which will satisfy the condition depends on the values of both $b_{1,i}$ and b_i. We have demonstrated how to infer $b_{1,i}$, but we are not aware of end-to-end methods which are available to infer b_i. One strategy is to resort to `pathchar` probing methods developed by Jacobson [12] and further refined and studied by Mah [20] and Downey [9]. While these methods employ network-internal ICMP probes and can be probe-intensive, we would use it only to obtain the base bandwidth b_i of a single physical link. More significantly, we do not in fact need to accurately infer b_i; instead we only need an *upper bound* on b_i. A valid upper bound provides us with a corresponding lower bound on the pacer packet size needed to satisfy the pacing condition and any pacer of equal or larger size would then suffice.

4.3 Recap

We now put together the pieces of our probing technique, showing how to guarantee that the technique generates the correct estimate $b_{i,j}$ over any subpath of L. We first infer the bottleneck bandwidth $b_{1,i}$ over the prefix of the path through link i as described in section 4.1, and infer an upper bound on b_i as described in section 4.2. We then build a Cartouche probe of the form $[apm\{pq\}^{r-1}pm]$, where $s(a)$ =1500 bytes is the size of the pacer packet, $s(m)$ =40 bytes is the size of the cartouche probe marker packet. Then by substituting $b_{1,i}$, b_i, $s(p)$ and $s(a)$ in the pacing equation (Lemma 3) to get the minimum size $s(p)$ for the magnifier packet. Using the obtained pacer, marker and magnifier

[3]Motivation for large pacer packets and small markers are provided later, as is a discussion of limitations posed by a fixed upper bound on the maximum pacer size.

packet sizes, send a paced cartouche probing sequence. The cartouche probes are paced up through link L_i with their egress link set to L_j. By measuring the end-host interarrival time Δ between the first and last marker packets (they are the only ones the end-host should receive), $b_{i,j} = \frac{r \cdot (s(p)+s(m))}{\Delta}$.

5 Shared Bottleneck Bandwidth

In section 4, we have shown how we can use paced cartouche probes to efficiently estimate the bottleneck bandwidth $b_{i,j}$ for the subpath $L_i, \ldots L_j$. That approach employed *hop-limited* packets, in which we set the TTL field to control the number of hops traversed by each probe packet. In this section, we extend our probing technique to enable the inference of the bottleneck bandwidth along the sequence of links shared by flows emanating from the same server and destined to two different clients (depicted in Figure 4).[4] Extending the technique to deal with shared links between more than two clients is straightforward (see for example [3]).

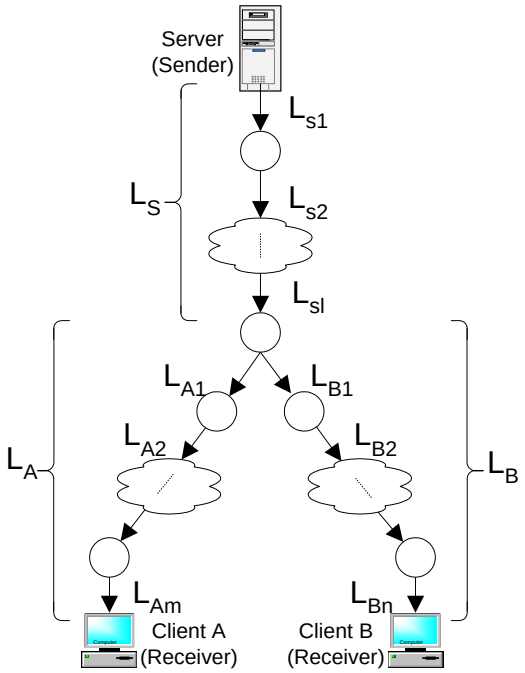

Figure 4: Notation used to describe the topology between a server and two clients.

Consider the set of links used to route traffic between a server and two different clients. Together these links form a tree T rooted at the server, with the clients at the leaves and routers at the internal nodes. The flows of packets sent from the server to each of the two clients share some (possibly none) of T's links and then continue on separate links en route to the different clients. This leads to the inverted Y topology of Figure 4.

[4]Using slightly different constructions, we can estimate the bottleneck bandwidth along the sequence of links *shared* by flows emanating from two different servers and destined to the same client.

As depicted in Figure 4, links on shared segments are denoted as L_S, while links on disjoint segments are labeled as L_A and L_B. We use b_S to denote the bottleneck bandwidth over links in L_S and employ similar notation for other sets of links. Our objective is to efficiently and accurately estimate b_S.

In order to infer b_S, we need to send a cartouche probe over L_S with the egress link as the last physical link in L_S using the same technique of section 4. To do so, we need to know the IP address of the last physical link in L_S, or alternatively, using hop-limited packets, we need to know the distance in hops (l) between S and the last physical link in L_S.[5] Neither of these alternatives is practical.

A more efficient way to coerce packets from a cartouche destined to one of the two clients to exit after the last physical link in L_S is to use multi-destination probes. Namely, instead of setting the TTL field values of probe packets to l, we set their *destination address* field to the *other* client. As an illustration, we send from source S a $[pm\{pq\}^{r-1}pm]$ cartouche probe sequence where $D(m) = D(n) = A$ and $D(p) = D(q) = B$. The two markers m destined to A travel together with the other probe packets destined to B only over the links in L_S, which is exactly the desired outcome. Client A then measures the interarrival time Δ between marker packets m and computes $\frac{r(s(p)+s(m))}{\Delta}$ to estimate b_S.

6 Experimental Results

In this section, we present results of extensive simulations that demonstrate the applicability of the proposed techniques under various parameter settings and network conditions.

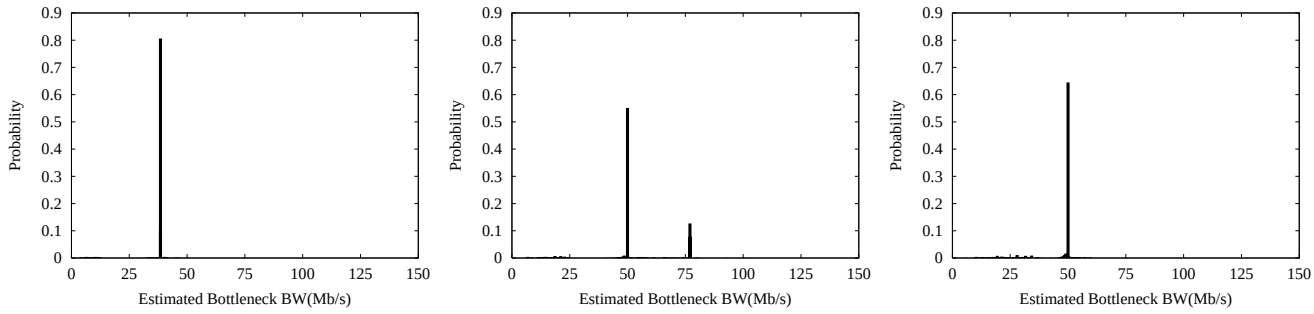

Figure 5: Distribution of estimated $b_{1,10}$ values after using a sequence of 1000 $[pm\{pq\}^{r-1}pm]$ cartouche probes of dimensionality r =1 (left) r =2 (middle) and r =3 (right). The setup includes 16 cross-traffic flows over each physical link in L, actual $b_{1,10}$ =50Mb/s and $b_{11,20}$ =1Mb/s.

Experimental Setup: We used the Network Simulator (ns) [21] to simulate a path L connecting two hosts A and B. L consists of 20 physical links L_1, L_2, ... L_{20}. Link bandwidth values b_1, b_2, ... b_{20} and link latencies d_1, d_2, ... d_{20} were hand-picked to illustrate various scenarios. Link cross-traffic

[5] One out-of-band method to determine the value of l is to run *traceroute* [13] on the path from S to A and again on the path from S to B, then compare the results to infer the number of matching hops. However, this is clumsy, inefficient and relies on network-internal probing.

was modeled by a set of aggregated Pareto ON/OFF UDP flows with 1.2 as the distribution shape parameter, 0.5 seconds as the average burst and idle times, and 32Kb/sec as mean flow rate. Also, packet sizes of the cross-traffic flows were drawn at random with an average size of 512 bytes and a maximum of 1500 bytes. By varying the number of cross-traffic flows over each link we define the level of congestion to be considered. Probe transmission, time measurements, logging and estimation functions were all performed at host A.

Estimation of Bottleneck Bandwidth of Path Prefix ($b_{1,i}$): As described in Section 4.1, our technique to measure path prefix bandwidth relies on sending a sequence of $[pm\{pq\}^{r-1}pm]$ cartouche probes from source A. We set $s(p) = 1500$ bytes, $s(m) = s(q) = 40$ bytes, $D(m) = B$ and $D(p) = D(q) = L_i$, then study different values for the cartouche dimension r. Our experiments also aim at studying the impact of cross traffic, varying the prefix length i, and the ratio between $b_{1,i}$ and $b_{i+1,20}$, all of which alter the interarrival of markers (and thus, $b_{1,i}$ estimation accuracy). In our simulations, we set $b_{i+1,20}$ =1Mb/s to make $b_{1,i}$ equal to the mentioned ratio. Host A monitors the interarrival time Δ of the responses to markers m, and uses $\frac{r(s(p)+s(m))}{\Delta}$ to estimate of $b_{1,i}$.

The presence of competing cross-traffic introduces jitter and perturbs the interarrival time of the markers m. In an effort to filter out such effects, we compile a histogram[6] of the frequency of each observed value for $b_{1,i}$ and pick the one with the largest frequency (the mode). Figure 5 shows the histograms we obtain trying to infer $b_{1,10}$ after using a sequence of 1000 $[pm\{pq\}^{r-1}pm]$ cartouche probes ($r = 1$ (left) $r = 2$ (middle) and $r = 3$ (right)). The setup includes 16 cross-traffic flows over each physical link in L, with actual $b_{1,10} = 50$ Mb/s and $b_{11,20} = 1$ Mb/s.

Note that the r=2 and r=3 cases lead to a correct $b_{1,10}$ estimate while the r=1 case does not. This could be explained using the marker interarrival preservation property presented in section 4.1. Using $s(p) = 1500$ bytes and $s(m) = 40$ bytes the condition $\frac{b_{1,i}}{b_{i+1,n}} \leq 38.5r$ must hold to deliver unperturbed marker interarrival times to the end-hosts. Since $\frac{b_{1,10}}{b_{11,20}} = 50$ the condition does not hold for $r = 1$. It does hold however for the $r = 2$ and $r = 3$ cases. This means that the maximum possible $b_{1,i}$ value we can estimate is $r * b_{i+1,n} * 38.5$ which is 38.5Mb/s and 77Mb/s and 115.5Mb/s for r =1,2 and 3 respectively. In fact this is the reason that the estimated $b_{1,10}$ value in the $r = 1$ case is 38.5Mb/s ($b_{11,20}$*38.5).

It might also be the case that the condition is satisfied but the presence of cross-traffic along $L_{11}, \ldots L_{20}$ leads to marker packets queueing in $L_{11}, \ldots L_{20}$ bottleneck queue for specific durations of time. The results as reflected through histogram bars corresponding to $b_{1,i}$ estimates of 38.5Mb/s and 77Mb/s and 115.5Mb/s for r =1,2 and 3 respectively. This is obvious in the $r = 2$ histogram of figure 5 where there exists noticeable mass at around 77Mb/s. No extra significant bars appear in the $r = 3$ histogram of figure 5, because of relatively large marker interarrival time induced by cartouche probes of dimension 3. This is consistent with our hypothesis that the larger the dimensionality of cartouche probing the more reliable our estimates are.

Figure 6 represents 3-dimensional grids plotting the estimated $b_{1,i}$ values (the bandwidth value corresponding to the highest histogram bar) when we vary the number of cross-traffic flows over each

[6] In all our experiments, we use a fixed bin width of 1Mbps for the histograms.

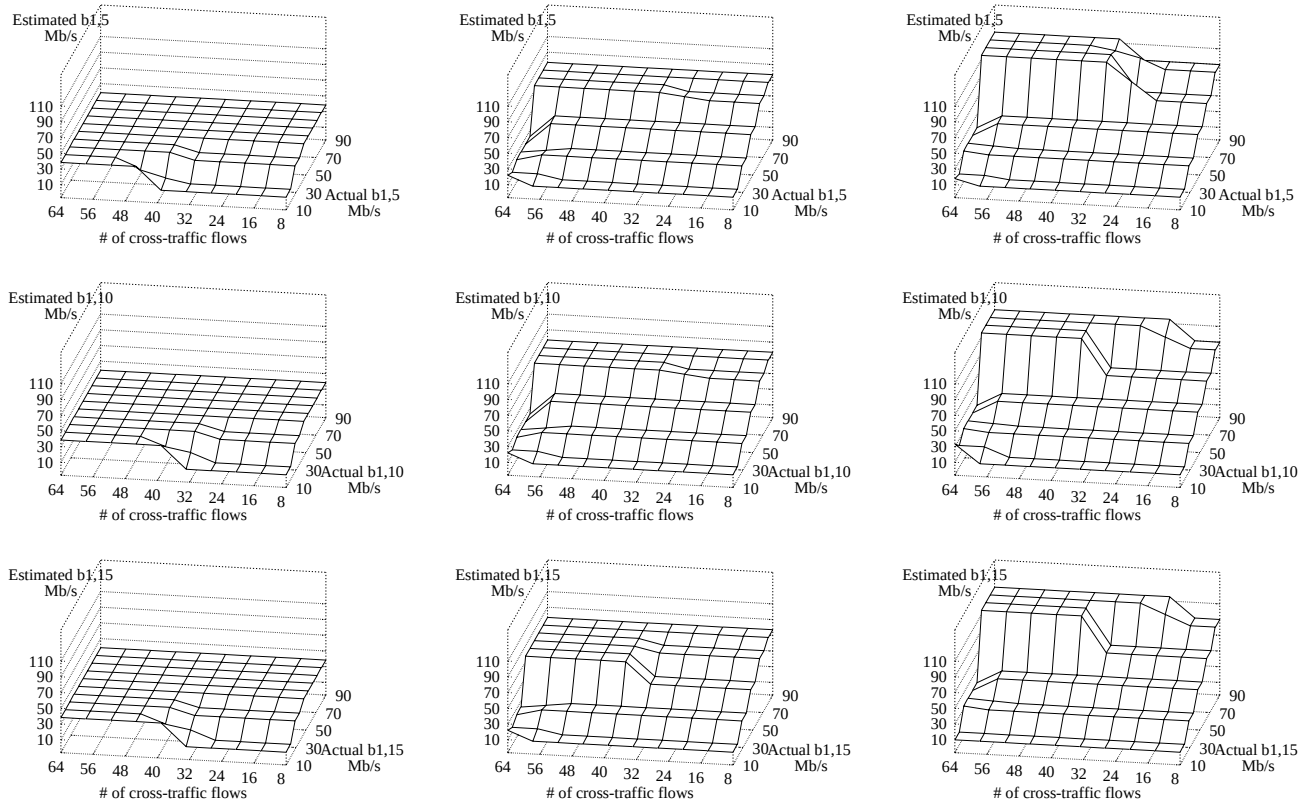

Figure 6: $b_{1,i}$: $i = 5$ (top), $i = 10$ (middle), $i = 15$ (bottom), and $r = 1$ (left), $r = 2$ (middle), $r = 3$ (right).

link and for different actual $b_{1,i}$ value (while always keeping $b_{i+1,20}$ =1Mb/s). These plots are given for various prefix lengths and various cartouche dimensions. By carefully inspecting the resulting grids, one can make the following three observations. First, the results confirm that the maximum possible $b_{1,i}$ value we can estimate using the experimental setup is 38.5Mb/s, 77Mb/s and 115.5Mb/s for r =1,2 and 3 respectively. Second, the closer the ratio between $b_{1,i}$ and $b_{i+1,20}$ to the value of r, the more susceptible the cartouche packets to interarrival time alterations due to cross-traffic. This advocates larger r values. Finally, we saw no significant impact due to prefix length (the value of i) in our experiments. We conclude that congestion along the path prefix presents the most significant issues, but can be compensated for using large values of r.

Other insights into the effects of congestion on $b_{1,i}$ estimation can be obtained by observing Figure 7. In this figure, for the $r = 3$ case, we plot the fraction of the probes yielding the mode bandwidth (i.e. the percentage of probes in the mode bin of the histogram). The plots are for a prefix length $i = 10$. Taking these percentage values as a measure of our *confidence* in the $b_{1,i}$ estimates, one can see that in some cases confidence decreases as congestion increases and in others confidence increases in tandem with congestion. The justification is that when the ratio between $b_{1,i}$ and $b_{i+1,20}$ is large compared to r we have a correct $b_{1,i}$ estimate with high confidence and the more we impose

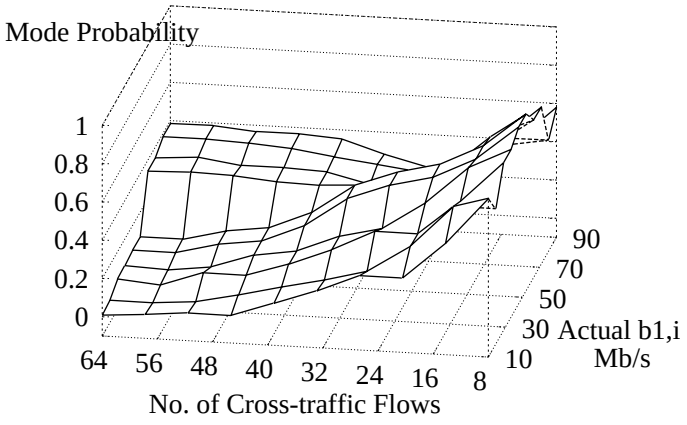

Figure 7: Mode probability of $b_{1,i}$ when using cartouche probes of dimension r =3 with a prefix length $i = 10$.

cross-traffic the less confident we get in our estimate. On the other hand, when the ratio is small compared to r we have a doubtful $b_{1,i}$ estimate to begin with and as cross-traffic increases we move to a wrong $b_{1,i}$ estimate, moreover confidence in the wrong $b_{1,i}$ estimate only builds as congestion increases!

Estimation of Subpath Bottleneck Bandwidth: In section 4, we have shown analytically that if pacing is guaranteed for cartouche probes over any physical link L_i, then we can extend our $b_{1,i}$ and b_i estimates to infer the bottleneck bandwidth $b_{i,j}$ over any sub-path $L_i, \ldots L_j$. In section 3.3, we have presented a sufficient condition for pacing a sequence of packets behind one *large* pacer packet. In this section, we study the limitations of using a single pacer packet on our inference methods.

Consider a scenario in which a single pacer packet precedes a cartouche probe of dimension $r = 1$ (i.e. it is of the form $[pmpm]$) and let g be a pacer packet. In order to guarantee that the cartouche probe is paced over a specific link L_i, then $s(g)$, $s(p)$ and $s(m)$ should satisfy the pacing condition of Lemma 3, warranting g being sufficiently large, while m and p are relatively small. Also, as demonstrated in section 4.1, a partially conflicting objective is that it is preferable to have p be large relative to m to avoid skewing the measured interarrival times of marker packets. In an effort to satisfy both requirements we fix $s(g) = 1500$ bytes, $s(m) = 40$ bytes and substitute in the pacing equation to get the largest possible value for $s(p)$. Figure 8 displays four $s(p)$ curves, as functions of $\frac{b_i}{b_{1,i}}$, resulting from substituting $s(g)$ and $s(m)$ values in the pacing equation inequalities. For a given $\frac{b_i}{b_{1,i}}$ value, the largest possible $s(p)$ value that yields the cartouche pacing is the minimum $s(p)$ value given by any of the four curves.

As evident from figure 8, $s(p)$ decreases dramatically as $\frac{b_i}{b_{1,i}}$ increases. In fact $s(p)$ becomes as low as 40 bytes when the ratio reaches 10. This makes the pacing of a cartouche probe using a *single*

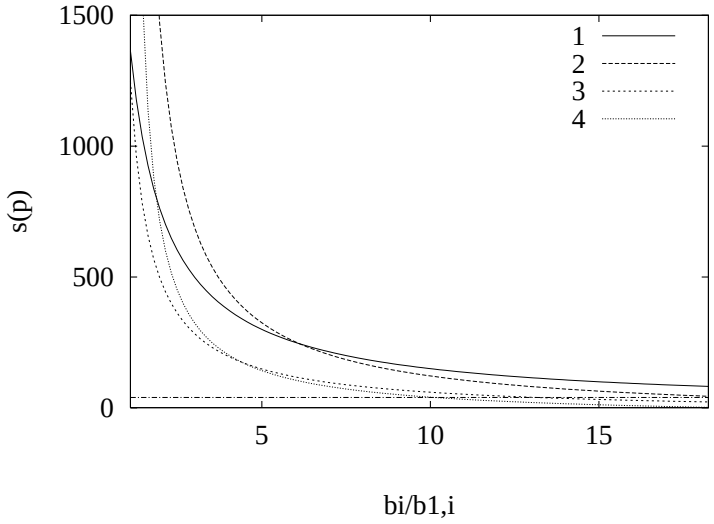

Figure 8: Pacing limitations associated a one packet pacer.

pacer packet not always practical, especially in light of the importance we attributed to cartouche dimensionality (r). Pacing an arbitrary number of packets behind a *single* large pacer packet or a *sequence* of pacer packets is the subject of future work.

7 Conclusions

We have described an end-to-end probing technique which is capable of inferring the bottleneck bandwidth along an arbitrary set of path segments in the network, or across the portion of a path shared by a set of connections. The constructions we advocate are built upon robust packet-pair techniques, and the inferences we draw are accurate under a variety of simulated network conditions and are robust to network effects such as the presence of bursty cross-traffic.

The end-to-end probing constructions we proposed in this paper are of a generic nature, and while we used them to tackle a specific problem (namely, measurement of bottleneck bandwidth), we believe that they will be equally instrumental in the measurement of other characteristic properties of arbitrary path segments (e.g. buffer sizes, queuing disciplines, etc.). We anticipate that lightweight mechanisms to facilitate end-to-end probing of metrics of interest, such as bottleneck bandwidth, will see increasing use as emerging network-aware applications optimize their performance via intelligent utilization of network resources. Our future work will address the challenges with Internet deployment of Cartouche probes, where we anticipate validation of the results to be among the most significant hurdles.

Appendix

A Basic Lemmas

In this section we prove some basic lemmas that we use later in proving the lemmas appearing earlier in the paper.

Lemma 7 *Let L be a sequence of n physical links L_1, L_2, $\ldots L_n$ with capacity bandwidths b_1, b_2, $\ldots b_n$ respectively. Also, let $[pq_1 \ldots q_m]$ be a probe consisting of a set of m paced packets which are injected back-to-back behind a pacer packet, where $D(p) = L_n$ and $D(q_w) = L_n$. If $\forall i \leq n$, $\min_{1 \leq w \leq m} \left(\frac{s(p)+\sum_{j=1}^{w-1} s(q_j)}{\sum_{j=1}^{w} s(q_j)} \right) \geq \frac{b_i}{b_{i-1}}$, then all follower packets q_w will queue behind p along the entire path.*

Proof: *(By induction over $L_1, L_2, \ldots, L_n$).*

Induction Basis: The packets of $[pq_1 \ldots q_m]$ queue back-to-back in L_1 queue by construction.

Induction Hypothesis: Assume that $\forall i \leq k-1$, $\min_{1 \leq w \leq m} \left(\frac{s(p)+\sum_{j=1}^{w-1} s(q_j)}{\sum_{j=1}^{w} s(q_j)} \right) \geq \frac{b_i}{b_{i-1}}$, and that all follower packets q_w queue behind p along the path $L_1, \ldots L_{k-1}$.

Induction Step: Using the induction hypothesis, all $[pq_1 \ldots q_m]$ packets queue back-to-back in L_{k-1} queue. In order for q_1 to be transmitted back-to-back after p over L_k, the transmission time of p over L_k must be at least as large as the interarrival time between p and q_1 over L_{k-1}. That is the condition $\frac{s(p)}{b_k} \geq \frac{s(q_1)}{b_{k-1}}$ (or equivalently $\frac{s(p)}{s(q_1)} \geq \frac{b_k}{b_{k-1}}$) must be satisfied. Similarly, in order for q_2 to be transmitted back-to-back after q_1, the transmission time of p and q_1 over L_k must be at least as large as the interarrival time between p and q_2 over L_{k-1}. That is the condition $\frac{s(p)+s(q_1)}{b_k} \geq \frac{s(q_1)+s(q_2)}{b_{k-1}}$ (or equivalently $\frac{s(p)+s(q_1)}{s(q_1)+s(q_2)} \geq \frac{b_k}{b_{k-1}}$) must be satisfied. In general, for q_w to be transmitted back-to-back after q_{w-1} over L_k the condition $\frac{s(p)+\sum_{j=1}^{w-1} s(q_j)}{\sum_{j=1}^{w} s(q_j)} \geq \frac{b_k}{b_{k-1}}$ must be satisfied. In order for all the $[pq_1 \ldots q_m]$ packets to be transmitted back-to-back over L_k, the condition $\frac{s(p)+\sum_{j=1}^{w-1} s(q_j)}{\sum_{j=1}^{w} s(q_j)} \geq \frac{b_k}{b_{k-1}} \quad \forall 1 \leq w \leq m$ (or equivalently $\min_{1 \leq w \leq m} \left(\frac{s(p)+\sum_{j=1}^{w-1} s(q_j)}{\sum_{j=1}^{w} s(q_j)} \right) \geq \frac{b_k}{b_{k-1}}$) must be satisfied.

■

Lemma 8 *Consider a path of n physical links $L_1, L_2, \ldots, L_n$ with base bandwidths $b_1, b_2, \ldots, b_n$ respectively. If $[pq_1 \ldots q_m]$ be a probe consisting of a set of m paced packets which are injected back-to-back behind a pacer packet, where $D(p) = L_n$, $D(q_w) = L_n$ and $s(p) \geq s(q_w)$, then the interarrival time Δ_i^w between p and q_w over link L_i satisfies the condition $\Delta_i^w \leq \frac{\sum_{j=1}^{w} s(q_j)}{min_{1 \leq k \leq i} b_k} \quad \forall 1 \leq w \leq m$.*

Proof: *(By induction over $L_1, L_2, \ldots, L_n$).*

Induction Basis: The packets of a $[pq_1 \ldots q_m]$ probe being transmitted back-to-back from L_1 queue, then by definition of *inter-arrival* time, $\Delta_1^w = \frac{\sum_{j=1}^{w} s(q_j)}{b_1}$.

Induction Hypothesis: Assume that $\Delta_{k-1}^w \leq \frac{\sum_{j=1}^{w} s(q_j)}{min_{1 \leq i \leq k-1} b_i} \quad \forall 1 \leq w \leq m$.

Induction Step: Many scenarios are possible for each $[pq_1 \ldots q_m]$ packet over L_k.

We start by discussing q_1 scenarios.

Case 1: q_1 queues behind p over L_k. In this case, $\Delta_k^1 = \frac{s(q_1)}{b_k} \leq \frac{s(q_1)}{min_{1 \leq i \leq k} b_i}$.

Case 2: q_1 does not queue behind p over L_k. This happens only if $\frac{s(p)}{b_k} < \Delta_{k-1}^1$. In this case, $\Delta_k^1 = \Delta_{k-1}^1 + \frac{s(q_1)-s(p)}{b_k}$. Given that $s(p) > s(q_1))$, we get $\Delta_k^1 < \Delta_{k-1}^1$ and using the induction hypothesis, we get $\Delta_k^1 < \frac{s(q_1)}{min_{1 \leq i \leq k-1} b_i}$. Combining the inequality $\frac{s(p)}{b_k} < \Delta_{k-1}^1$ which must be satisfied in this case with the induction hypothesis, we get that $\frac{s(p)}{b_k} < \frac{s(q_1)}{min_{1 \leq i \leq k-1} b_i}$; i.e. $\frac{s(p)}{s(q_1)} < \frac{b_k}{min_{1 \leq i \leq k-1} b_i}$. Since $\frac{s(p)}{s(q_1)} > 1$ then $b_k > min_{1 \leq i \leq k-1} b_i$ and thus $min_{1 \leq i \leq k} b_i = min_{1 \leq i \leq k-1} b_i$. So, $\Delta_k^1 \leq \frac{s(q_1)}{min_{1 \leq i \leq k} b_i}$.

From cases 1 and 2, $\Delta_k^1 \leq \frac{s(q_1)}{min_{1 \leq i \leq k} b_i}$. We next discuss q_2 scenarios over L_k.

Case 1: q_2 queues behind q_1 over L_k. In this case, $\Delta_k^2 = \Delta_k^1 + \frac{s(q_2)}{b_k}$. So, $\Delta_k^2 \leq \frac{s(q_1)+s(q_2)}{min_{1 \leq i \leq k} b_i}$.

Case 2: q_2 does not queue behind q_1 over L_k. This happens only if $\frac{s(p)+s(q_1)}{b_k} < \Delta_{k-1}^2$. In this case, $\Delta_k^2 = \Delta_{k-1}^2 + \frac{s(q_2)-s(p)}{b_k}$. Given that $s(p) > s(q_2))$, we get $\Delta_k^2 < \Delta_{k-1}^2$ and using the induction hypothesis, we get $\Delta_k^2 < \frac{s(q_1)+s(q_2)}{min_{1 \leq i \leq k-1} b_i}$. Combining the inequality $\frac{s(p)+s(q_1)}{b_k} < \Delta_{k-1}^2$ which must be satisfied in this case with the induction hypothesis, we get that $\frac{s(p)+s(q_1)}{b_k} < \frac{s(q_1)+s(q_2)}{min_{1 \leq i \leq k-1} b_i}$; i.e. $\frac{s(p)+s(q_1)}{s(q_1)+s(q_2)} < \frac{b_k}{min_{1 \leq i \leq k-1} b_i}$. Since $\frac{s(p)+s(q_1)}{s(q_1)+s(q_2)} > 1$ then $b_k > min_{1 \leq i \leq k-1} b_i$ and thus $min_{1 \leq i \leq k} b_i = min_{1 \leq i \leq k-1} b_i$. So, $\Delta_k^2 \leq \frac{s(q_1)+s(q_2)}{min_{1 \leq i \leq k} b_i}$.

From cases 1 and 2, $\Delta_k^2 \leq \frac{s(q_1)+s(q_2)}{min_{1 \leq i \leq k} b_i}$. In general, q_w scenarios over L_k are:

Case 1: q_w queues behind q_{w-1} over L_k. In this case, $\Delta_k^w = \Delta_k^{w-1} + \frac{s(q_w)}{b_k}$. So, $\Delta_k^w \leq \frac{\sum_{j=1}^{w} s(q_j)}{min_{1 \leq i \leq k} b_i}$.

Case 2: q_w does not queue behind q_{w-1} over L_k. This happens only if $\frac{s(p)+\sum_{j=1}^{w-1} s(q_j)}{b_k} < \Delta_{k-1}^w$. In this case, $\Delta_k^w = \Delta_{k-1}^w + \frac{s(q_w)-s(p)}{b_k}$. Given that $s(p) > s(q_w))$, we get $\Delta_k^w < \Delta_{k-1}^w$ and using the induction hypothesis, we get $\Delta_k^w < \frac{\sum_{j=1}^{w} s(q_j)}{min_{1 \leq i \leq k-1} b_i}$. Combining the inequality $\frac{s(p)+\sum_{j=1}^{w-1} s(q_j)}{b_k} < \Delta_{k-1}^w$ which must be satisfied in this case with the induction hypothesis, we get that $\frac{s(p)+\sum_{j=1}^{w-1} s(q_j)}{b_k} < \frac{\sum_{j=1}^{w} s(q_j)}{min_{1 \leq i \leq k-1} b_i}$; i.e. $\frac{s(p)+\sum_{j=1}^{w-1} s(q_j)}{\sum_{j=1}^{w} s(q_j)} < \frac{b_k}{min_{1 \leq i \leq k-1} b_i}$. Since $\frac{s(p)+\sum_{j=1}^{w-1} s(q_j)}{\sum_{j=1}^{w} s(q_j)} > 1$ then $b_k > min_{1 \leq i \leq k-1} b_i$ and thus $min_{1 \leq i \leq k} b_i = min_{1 \leq i \leq k-1} b_i$. So, $\Delta_k^w \leq \frac{\sum_{j=1}^{w} s(q_j)}{min_{1 \leq i \leq k} b_i}$.

From cases 1 and 2, $\Delta_k^w \leq \frac{\sum_{j=1}^{w} s(q_j)}{min_{1 \leq i \leq k} b_i}$.

■

B Main Lemmas

In this section, we prove all the main lemmas presented in the paper.

Lemma 2 *Consider a path of n physical links $L_1, L_2, \ldots, L_n$ with base bandwidths $b_1, b_2, \ldots, b_n$ respectively. If a probe of the form [pq] is injected at L_1, with $D(p) = D(q) = L_n$ and if $\forall i < n$, $\frac{s(p)}{s(q)} \geq \frac{b_{i+1}}{b_i}$, then [pq] will remain back-to-back along the entire path.*

Proof: This is a special case of lemma 7. ■

Lemma 7 *Let L be a sequence of n physical links L_1, L_2, $\ldots L_n$ with capacity bandwidths b_1, b_2, $\ldots b_n$ respectively. Also, let $[pq_1 \ldots q_m]$ be a probe consisting of a set of m paced packets which are injected back-to-back behind a pacer packet, where $D(p) = L_k$, $D(q_i) = L_n$ and $s(p) > s(q_w)$. Then a sufficient condition for all follower packets q_w to queue behind p at link L_k is*

$$\min_{1 \leq w \leq m} \left(\frac{s(p) + \sum_{j=1}^{w-1} s(q_j)}{\sum_{j=1}^{w} s(q_j)} \right) \geq \frac{b_k}{\min_{i<k} b_i}.$$

Proof: The proof is based on the results of lemma 8. Lemma 8 shows that if the $[pq_1 \ldots q_m]$ probe packets are injected back-to-back at $L1$ then the interarrival time Δ_{k-1}^{w} between p and q_w over link L_{k-1} satisfies the inequality $\Delta_{k-1}^{w} \leq \frac{\sum_{j=1}^{w} s(q_j)}{min_{1 \leq i \leq k-1} b_i}$.

In order to guarrantee that q_1 will queue back-to-back after p over L_k, the transmission time of p over L_k must be at least as large as the maximum Δ_{k-1}^{1} value. That is the condition $\frac{s(p)}{b_k} \geq \frac{s(q_1)}{min_{1 \leq i \leq k-1} b_i}$ (or equivalently $\frac{s(p)}{s(q_1)} \geq \frac{b_k}{min_{1 \leq i \leq k-1} b_i}$) must be satisfied. Similarly, in order to guarrantee that q_2 will be transmitted back-to-back after q_1 over L_k, the transmission time of p and q_1 over L_k must be at least as large as the maximum Δ_{k-1}^{2} value. That is the condition $\frac{s(p)+s(q_1)}{b_k} \geq \frac{s(q_1)+s(q_2)}{min_{1 \leq i \leq k-1} b_i}$ (or equivalently $\frac{s(p)+s(q_1)}{s(q_1)+s(q_2)} \geq \frac{b_k}{min_{1 \leq i \leq k-1} b_i}$) must be satisfied. In general, in order to guarrantee that q_w willbe transmitted back-to-back after q_{w-1} over L_k the condition $\frac{s(p)+\sum_{j=1}^{w-1} s(q_j)}{\sum_{j=1}^{w} s(q_j)} \geq \frac{b_k}{min_{1 \leq i \leq k-1} b_i}$ must be satisfied. In order for all the $[pq_1 \ldots q_m]$ packets to be transmitted back-to-back over L_k, the condition $\frac{s(p)+\sum_{j=1}^{w-1} s(q_j)}{\sum_{j=1}^{w} s(q_j)} \geq \frac{b_k}{min_{1 \leq i \leq k-1} b_i}$ $\forall 1 \leq w \leq m$ (or equivalently $\min_{1 \leq w \leq m} \left(\frac{s(p)+\sum_{j=1}^{w-1} s(q_j)}{\sum_{j=1}^{w} s(q_j)} \right) \geq \frac{b_k}{min_{1 \leq i \leq k-1} b_i}$) must be satisfied. ■

Lemma 4 *Consider a path of n physical links $L_1, L_2, \ldots, L_n$ with base bandwidths $b_1, b_2, \ldots, b_n$ respectively. If a probe of the form [p][p] is injected at L_1 with $D(p) = L_n$ and interarrival time of Δ, then Δ will be preserved over all links L_i if and only if $\frac{s(p)}{\Delta} \leq \min_{1 \leq k \leq n} b_k$.*

Proof: *(By induction over $L_1, L_2, \ldots, L_n$).*

Induction Basis: The packets' inter-departure time from L_1 queue is Δ and by definition of *inter-departure* time, Δ cannot be less than the transmission time of p over L_1. That is $\Delta \geq \frac{s(p)}{b_1}$. Thus, $\frac{s(p)}{\Delta} \leq b_1$.

Induction Hypothesis: Assume that packets' interarrival time at all links L_2, $\ldots L_{n-1}$ queues is Δ *and* $\frac{s(p)}{\Delta} \leq \min_{1 \leq k \leq (n-1)} b_k$.

Induction Step: From the induction hypothesis, the interarrival time at L_{n-1} queue is Δ. Either of the following two cases must happen:

Case 1: The second packet queues behind the first packet in L_{n-1} queue. In this case Δ is altered and the new inter-departure time from L_{n-1} queue becomes equal to the time to transmit the second packet over L_n; that is, equal to $\frac{s(p)}{b_n}$. Queueing occurs if and only if the second packet is totally received in L_{n-1} queue before the first packet is totally transmitted over L_n, i.e. if and only if $\frac{s(p)}{\Delta} > b_n$. By combining this condition with the induction hypothesis $\frac{s(p)}{\Delta} \leq \min_{1\leq k\leq (n-1)} b_k$, we infer that $b_n = \min_{1\leq k\leq n} b_k$ and thus $\frac{s(p)}{\Delta} > \min_{1\leq k\leq n} b_k$.

Case 2: The second packet does not queue behind the first packet in L_{n-1} queue. In this case Δ is not altered, the inter-departure time from L_{n-1} queue remains Δ because both packets are of the same size $s(p)$ and thus their transmission times over L_n are equal. Queueing does does not occur if and only if $\frac{s(p)}{\Delta} \leq b_n$. We can therefore again infer that $\frac{s(p)}{\Delta} \leq \min_{1\leq k\leq n} b_k$.

From cases 1 and 2, the packets' inter-departure time from L_{n-1} queue (inter-arrival time at L_n queue) remains Δ if and only if $\frac{s(p)}{\Delta} \leq \min_{1\leq k\leq n} b_k$.

■

Lemma 6 *Let L be a sequence of n physical links L_1, L_2, ...L_n with base bandwidths b_1, b_2 , ...b_n respectively. Given a cartouche of the form $[pm\{pq\}^{r-1}pm]$ over L with L_n as its egress link, let Δ_j^m be the interarrival time between the first and the last marker packets at link L_j, and similarly let Δ_j^p be the interarrival time between the first and the last magnifier packets at link L_j, then $\Delta_j^p = \Delta_j^m = \frac{r(s(p)+s(m))}{\min_{1\leq i\leq j} b_i}$.*

Proof: We will provide the proof for the $r = 1$ case. The general $r > 1$ proof is defined by r straightforward applications of the $r = 1$ case. *Proof by induction over $L_1, L_2, \ldots, L_n$.*

Induction Basis: The packets of a $[pmpm]$ probe being transmitted back-to-back from L_1 queue, then by definition of *inter-departure* time, $\Delta_1^p = \Delta_1^m = \frac{s(p)+s(m)}{b_1}$.

Induction Hypothesis: Assume that marker packets' interarrival time Δ_{j-1}^m and magnifier packets' interarrival time Δ_{j-1}^p at all links L_2, ... L_{j-1} queues are $\Delta_{j-1}^p = \Delta_{j-1}^m = \frac{s(p)+s(m)}{\min_{1\leq i\leq j-1} b_i}$.

Induction Step: Let Δ_{j-1}^1 be the interarrival time between the first p magnifier and the first m marker packets on L_{j-1}. Also, let Δ_{j-1}^2 be the interarrival time between the first m marker and the second p magnifier packets and Δ_{j-1}^3 be the interarrival time between the second p magnifier and the second m marker packets. One can see that $\Delta_{j-1}^p = \Delta_{j-1}^1 + \Delta_{j-1}^2$ and $\Delta_{j-1}^m = \Delta_{j-1}^2 + \Delta_{j-1}^3$. Using the induction hypothesis and lemma 8, we get that $\Delta_{j-1}^1 = \Delta_{j-1}^3 = \mathcal{A} \leq \frac{s(m)}{min_{1\leq k\leq j-1} b_k}$ and $\Delta_{j-1}^2 = \mathcal{B} \geq \frac{s(p)}{min_{1\leq k\leq j-1} b_k}$. These inequalities ensure that there *cannot* be a case where the second magnifier packet is queued behind the first marker packet ($\frac{s(m)}{b_j} \geq \mathcal{B}$) in L_j while the first marker packet is not queued behind the first magnifier packet ($\frac{s(p)}{b_j} < \mathcal{A}$) in L_j.

We will now enumerate the possible scenarios that may happen for the magnifier and marker packets over L_j:

Case 1: ($\frac{s(p)}{b_j} < \mathcal{A}$) No queueing occurs for the cartouche probe packets in L_j queue and $min_{1\leq k\leq j}b_k = min_{1\leq k\leq j-1}b_k$. In this case, $\Delta_j^p = \Delta_{j-1}^p + \frac{s(p)}{b_j} - \frac{s(p)}{b_j} = \Delta_{j-1}^p$ and $\Delta_j^m = \Delta_{j-1}^m + \frac{s(m)}{b_j} - \frac{s(m)}{b_j} = \Delta_{j-1}^m$. Using the induction hypothesis, $\Delta_j^p = \Delta_j^m = \frac{s(p)+s(m)}{\min_{1\leq i\leq j} b_i}$.

Case 2: ($\frac{s(p)}{b_j} \geq \mathcal{A}$) AND ($\frac{s(p)+s(m)}{b_j} < \mathcal{A}+\mathcal{B}$) Only Marker packets queue behind their preceding magnifier packets and $min_{1\leq k\leq j}b_k = min_{1\leq k\leq j-1}b_k$. In this case, $\Delta_j^p = \Delta_{j-1}^p + \frac{s(p)}{b_j} - \frac{s(p)}{b_j} = \Delta_{j-1}^p$ and $\Delta_j^m = \Delta_j^p + \frac{s(m)}{b_j} - \frac{s(m)}{b_j} = \Delta_j^p$. That is $\Delta_j^p = \Delta_j^m = \frac{s(p)+s(m)}{\min_{1\leq i\leq j} b_i}$.

Case 3: ($\frac{s(p)+s(m)}{b_j} \geq \mathcal{A} + \mathcal{B}$) All four cartouche probe packets queue back-to-back in L_j queue and $min_{1\leq k\leq j}b_k = b_j$. In this case, $\Delta_j^p = \frac{s(p)+s(m)}{b_j}$ and $\Delta_j^m = \frac{s(p)+s(m)}{b_j}$. This means that $\Delta_j^p = \Delta_j^m = \frac{s(p)+s(m)}{\min_{1\leq i\leq j} b_i}$.

From cases 1, 2 and 3, $\Delta_j^p = \Delta_j^m = \frac{s(p)+s(m)}{\min_{1\leq i\leq j} b_i}$.

■

References

[1] A. Barbir et al. Known CDN Request-Routing Mechanisms. http://www.globecom.net/ietf/draft/draft-cain-cdnp-known-request-routing-01.html, February 2001.

[2] D. Andersen, H. Balakrishnan, M. F. Kaashoek, and R. Morris. Resilient Overlay Networks. In *Proceedings of SOSP 2001 (to appear)*, Banff, Canada, October 2001.

[3] A. Bestavros, J. Byers, and K. Harfoush. Inference and Labeling of Metric-Induced Network Topologies. Technical Report BUCS-TR-2001-010, Boston University, Computer Science Department, June 2001.

[4] J. C. Bolot. End-to-end Packet Delay and Loss Behavior in the Internet. In *SIGCOMM '93*, pages 289–298, September 1993.

[5] J. Byers, M. Luby, and M. Mitzenmacher. Accessing Multiple Mirror Sites in Parallel: Using Tornado Codes to Speed Up Downloads. In *Proceedings of IEEE INFOCOM '99*, pages 275–83, March 1999.

[6] R. L. Carter and M. E. Crovella. Measuring bottleneck link speed in packet switched networks. *Performance Evaluation*, 27&28:297–318, 1996.

[7] Y.-H. Chu, S. Rao, and H. Zhang. A Case for End-System Multicast. In *ACM SIGMETRICS '00*, Santa Clara, CA, June 2000.

[8] M. E. Crovella, R. Frangioso, and M. Harchol-Balter. Connection Scheduling in Web Servers. In *Proceedings of 1999 USENIX Symposium on Internet Technologies and Systems (USITS '99)*, Boulder, CO, October 1999.

[9] A. Downey. Using Pathchar to Estimate Internet Link Characteristics. In *SIGCOMM' 99*, Boston, MA, August 1999.

[10] N. Duffield, F. Lo Presti, V. Paxson, and D. Towsley. Inferring Link Loss Using Striped Unicast Probes. In *IEEE INFOCOM 2001*, April 2001.

[11] K. Harfoush, A. Bestavros, and J. Byers. Robust Identification of Shared Losses Using End-to-End Unicast Probes. In *8th International Conference on Network Protocols (ICNP)*, Osaka, Japan, November 2000.

[12] V. Jacobson. `Pathchar`: A Tool to Infer Characteristics of Internet Paths. ftp://ftp.ee.lbl.gov/pathchar.

[13] V. Jacobson. traceroute. ftp://ftp.ee.lbl.gov/traceroute.tar.Z, 1989.

[14] J. Jannotti, D. Gifford, K. Johnson, M. F. Kaashoek, and J. O'Toole Jr. Overcast: Reliable Multicasting with an Overlay Network. In *Proceedings of OSDI 2000*, San Diego, CA, October 2000.

[15] J. Kangasharju, J. Roberts, and K. W. Ross. Object Replication Strategies in Content Distribution Networks. In *Proceedings of WCW'01: Web Caching and Content Distribution Workshop*, Boston, MA, June 2001.

[16] S. Keshav. A Control-Theoretic Approach to Flow Control. In *SIGCOMM '91*, September 1991.

[17] S. Keshav. *Congestion Control in Computer Networks.* PhD thesis, University of California at Berkeley, September 1991.

[18] K. Lai and M. Baker. Measuring Link Bandwidths Using a Deterministic Model of Packet Delay. In *SIGCOMM '00*, Stockholm, August 2000.

[19] K. Lai and M. Baker. Nettimer: A tool for Measuring Bottleneck Link Bandwidth. In *Proceedings of USITS '01*, March 2001.

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

[21] `ns`: Network Simulator. http://www-mash.cs.berkeley.edu/ns/ns.html.

[22] V. Paxson. End-to-end Routing Behavior in the Internet. In *SIGCOMM '96*, Stanford, California, August 1996.

[23] V. Paxson. End-to-end Internet Packet Dynamics. In *SIGCOMM*, 1997.

[24] V. Paxson. *Measurements and Analysis of End-to-end Internet Dynamics.* PhD thesis, U.C. Berkeley and Lawrence Berkeley Laboratory, 1997.

[25] P. Radoslavov, R. Govindan, and D. Estrin. Topology-Informed Internet Replica Placement. In *Proceedings of WCW'01: Web Caching and Content Distribution Workshop*, Boston, MA, June 2001.

[26] P. Rodriguez, A. Kirpal, and E. Biersack. Parallel-access for Mirror Sites in the Internet. In *Proceedings of IEEE INFOCOM '00*, March 2000.

[27] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan. Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. In *Proceedings of ACM SIGCOMM'01 (to appear)*, San Diego, CA, August 2001.

[28] J. Yoon, A. Bestavros, and I. Matta. SomeCast: A Paradigm for Real-Time Adaptive Reliable Multicast. In *Proceedings of RTAS'2000: The IEEE Real-Time Technology and Applications Symposium*, Washington, DC, May 2000.