Measuring Bottleneck Bandwidth of Targeted Path Segments

Khaled Harfoush
Department of Computer Science
North Carolina State University
Raleigh, NC 27695
harfoush@cs.ncsu.edu

Azer Bestavros
Computer Science Department
Boston University
Boston, MA 02215
best@cs.bu.edu

John Byers
Computer Science Department
Boston University
Boston, MA 02215
byers@cs.bu.edu

Abstract— **Accurate measurement of network bandwidth is crucial for network management applications as well as flexible Internet applications and protocols which actively manage and dynamically adapt to changing utilization of network resources. 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 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. We evaluate our technique through extensive ns simulations, then provide a comparative Internet perfomance evaluation against hop-by-hop techniques. We also describe a number of applications which we foresee as standing to benefit from solutions to this problem, ranging from network troubleshooting and capacity provisioning to optimizing the layout of application-level overlay networks to optimized replica placement.**

I. Introduction

Measurement of network bandwidth is crucial for many Internet applications and protocols, especially those involving the transfer of large files and those involving the delivery of content with real-time QoS constraints, such as streaming media. Some specific examples of applications which can leverage accurate bandwidth estimation include end-system multicast and overlay network configuration protocols [6], [19], [1], content location and delivery in peer-to-peer (P2P) networks [33], [3], network-aware cache or replica placement policies [20], [31], and flow scheduling and admission control policies at massively-accessed content servers [7]. In addition, accurate measurements of network bandwidth are useful to network operators concerned with problems such as capacity provisioning, traffic engineering, network troubleshooting and verification of service level agreements (SLAs).

Bandwidth Measurement: Two different measures used in end-to-end network bandwidth estimation are *bottleneck bandwidth*, or the maximum transmission rate that could be achieved between two hosts at the endpoints of a given path in the absence of any competing traffic, and *available bandwidth*, the portion of the bottleneck bandwidth along a path that could be acquired by a given flow at a given instant in time. Both of these measures are important, and each captures different relevant properties of the network. Bottleneck bandwidth is a static baseline measure that applies over long time-scales (up to the time-scale at which network paths change), and is independent of the particular traffic dynamics at a time instant. Available bandwidth provides a dynamic measure of the load on a path, or more precisely, the residual capacity of a path. Additional application-specific information must then be applied before making meaningful use of either measure; for example, the rate appropriated by an additional TCP flow is quite different than the unused capacity along a path [18], [11]. While measures of available bandwidth are certainly more useful for control or optimization of processes operating at short time scales, processes operating at longer time scales (e.g. server selection or admission control) will find estimates of both measures to be helpful, while many network management applications (e.g. capacity provisioning) are concerned primarily with bottleneck bandwidth. In this paper, we focus on measuring bottleneck bandwidth.

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 sources to use to download data in parallel. This scenario may arise when downloading content in parallel from a subset of mirror sites or multicast sources [4], [32], [12], or from a subset of peer nodes in P2P environments [3]. 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 [6], [19].

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. Similarly, in the second scenario illustrated in Figure 1 (right), the identification of

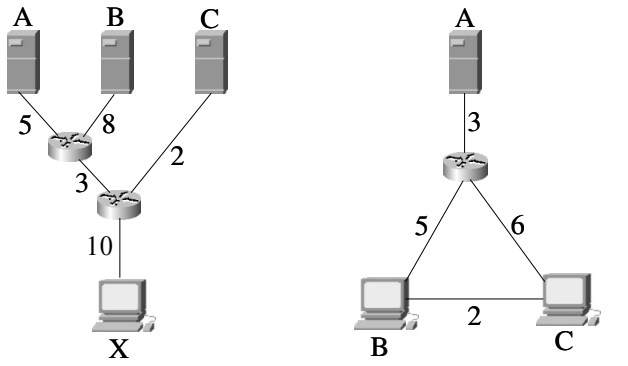

Fig. 1. Leveraging shared bandwidth measurement for optimizing parallel downloads (left) and overlay network organization (right). Numeric labels represent bottleneck bandwidth of path segments in Mbps.

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 an *arbitrary subpath* of a route between a set of end-points. By subpath, we mean a sequence of consecutive network links between any 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 achieve this by (1) targeting the packet to i (if i's IP address 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' are known diverge at node i. Our methods are much less *resource-intensive* than existing hop-by-hop methods for estimating bandwidth along a path and much more *general* than end-to-end methods for measuring bottleneck bandwidth. In particular, our method provides the following advantages over existing techniques: (1) it has more robust filtering, (2) it can estimate bandwidth on links not visible at end-to-end scope, and (3) it can measure the bandwidth of fast links following slow links.

The remainder of this paper is organized as follows. In Section II, we review existing literature. In Section III, we develop a basic probing toolkit, comprising existing methods and our new ideas. We compose several of these tools together in Section IV to measure the bottleneck bandwidth along arbitrary subpaths, as well as that shared by a set of flows. In Sections V and VI, we present results of simulation and Internet validation experiments, showing the effectiveness of our constructions. For detailed proofs of the lemmas and corollaries in this paper, we refer the reader to [13].

II. Related Work

One way of classifying bandwidth estimation techniques is based on whether they conduct hop-by-hop [16], [25], [9], [23], [24] or end-to-end [21], [2], [5] measurements. Hop-by-hop 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 existing end-to-end techniques. Another classification of bandwidth measurement techniques is based on whether they measure the bottleneck bandwidth [16], [25], [9], [21], [5], [23], [24] or the available bandwidth [2], [5], [18], [11] 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: *single packet* probing [2], [16], [25], [9]; *packet bunch* probing, employing a group of packets sent back-to-back [5]; *uniform packet-pair* probing, employing two back-to-back packets of the same size [21], [5]; and *non-uniform packet-pair* probing, employing two back-to-back packets of different sizes [23], [24]. The probing techniques we will propose can be classified as packet-bunch probes with non-uniform packet sizes.

Finally, one can classify bandwidth estimation techniques into *active* 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 [24], which uses a packet-pair technique at the transport level to passively estimate bottleneck link bandwidth. The techniques we propose in this paper are applied actively.

The probing constructions most closely related to ours are the "packet-pair" [22] and "tailgating" [23] constructions. We discuss relevant technical properties of these constructions, which we employ and build upon in Section III.

III. Probing Toolkit

In this section, we describe 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 the end-to-end measurement of subpath bottleneck bandwidth, which we describe in Section IV.

A. Basic Definitions

For the purposes of this paper, 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 [2], [21], [5], [24], [23]. 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) [10], [14]. 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 that we describe later in the paper.

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 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.

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; packets in each of the two-packet probes are transmitted back-to-back while there is time-separation between the three probes.

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 is the same as the interdeparture time of packets p and q at the preceding link on the path.

B. Existing Probing Methods and Properties

One of the essential techniques that we build upon is the use of "packet-pairs", originally used by Keshav [22], and subsequently refined by Carter and Crovella [5], Paxson [28], [30], [29] and Lai and Baker [24], to determine bottleneck bandwidth on a network path. Packet-pair techniques rely on the following property, which holds under an assumed network model discussed later in this section.

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 "packet-tailgating". This technique was introduced by Lai and Baker in [23] and evaluated within their *nettimer* tool [24] to estimate the bottleneck bandwidth of all physical links along a path. The packet-tailgating technique hinges on the following property [24], 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, which are common to most probing studies (e.g., [2], [5], [23], [24]), are enumerated below:

(1) Routers are store-and-forward and use FIFO queueing.
(2) Probing hosts can 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 discarded in experimental (simulation or implementation) settings to establish the robustness of the constructions in realistic settings.

C. Ensuring Back-to-Back Queuing at a Given Link

We now describe the first of our constructions—a construction that allows us to establish conditions that guarantee that all constituent packets of a probe will queue up back-to-back at a given intermediate link along a given path. We do so through the use of a (typically large) *pacer* packet, which leads the probe into the network.

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 p in the queue at router $D(p)$. At this point, p is dropped.

The following lemma expresses the condition guaranteeing that probe remains back-to-back at the pacer packet's final destination.

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

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

D. Preserving Packet Interarrival Times Over a Subpath

Our next construction allows us to tackle another challenge, which is to some extent complementary to pacing—namely how we can ensure that the interarrival time of two packets at a specific link L_i along a path can be *preserved* as these packets traverse additional links en route to their common destination L_n. The ability to preserve packet spacing over the subpath $L_i L_{i+1} \dots L_n$ enables us to measure such spacing remotely (at L_n). The following lemma establishes a necessary condition for the preservation of packet spacing over a sequence of links.

Lemma 4: Preservation of Spacing. Consider a path of n physical links $L_1, L_2, \dots, L_n$ with base bandwidths $b_1, b_2, \dots, b_n$ respectively. If a probe of the form $[p][p]$ is injected at L_1 with $D(p) = L_n$ and an 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 $L_{i+1}, \dots L_n$, the condition $\frac{s(p)}{\Delta_i} \le \min_{(i+1) \le k \le n} b_k$ must hold.

IV. Subpath Bandwidth Measurement Using Cartouche Probing

In this section we present our main probing structures, which enable us to achieve our stated goal of estimating the bottleneck bandwidth for an arbitrary path segment. In particular, given a path consisting of a sequence of links $L_1, \dots, L_n$ with base bandwidths $b_1, \dots, b_n$, our goal is to estimate the bottleneck bandwidth of an arbitrary sequence of links along that path, i.e. estimate $\min_{i \le k \le j} b_k$, for arbitrary i and j such that $i \le j \le n$. We use the shorthand $b_{i,j}$ to denote the bottleneck bandwidth in the interval between links i and j inclusive.

We proceed by first demonstrating how to estimate the bottleneck bandwidth over a *prefix* of a path and over a *suffix* of a path. Techniques for handling these two easier cases, which are often useful in their own right, will provide insight as to how to approach the general problem.

A. Estimating Bandwidth Over a Prefix of the Path

We begin by estimating the bottleneck bandwidth along a path prefix, i.e. inferring $b_{1,j}$. Since the packet-pair technique described in Section III provides an estimate for $b_{1,n}$, it follows that if $b_{1,j} \le b_{j+1,n}$, 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 the end-host may infer $b_{1,j}$. Indeed, Lemma 4 gives us the condition we must satisfy to ensure such preservation. To do so, 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.

Lemma 5: Consider a path of n physical links L_1, L_2, $\dots L_n$ with capacity bandwidths b_1, b_2, $\dots b_n$ 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 $1 \le j \le n$, is $\frac{r \cdot s(p)}{\min_{1 \le k \le j} b_k}$.

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}, $\dots L_n$. 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 we need the condition $r \ge \frac{b_{1,j}}{b_{j+1,n}}$ to be satisfied. 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 preserve the interarrival times of probe packets at an internal link L_j as these packets traverse subsequent links $L_{j+1} \dots L_n$ is to use small packets as "markers" that delimit measurement boundaries. Small packets have lower transmission delays and thus are less susceptible to variation in their interarrival times. 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* $[pm\{pq\}^{r-1}pm]$ over the set of links $L_1, \dots L_j, \dots, L_n$ is a sequence of $r+1$ heterogenous packet-pairs in which $s(p) \ge s(m) = s(q)$, $D(p) = D(q) = L_j$, and $D(m) = L_n$. 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, and r as the *cartouche size*. With the exception of the first and last marker packets m, all packets of the cartouche are targeted to L_j, which is called the *egress link* of the cartouche. L_n, the destination of the first and last marker packets is called the *target* of the cartouche.

Figure 2 shows the composition and progression of a cartouche of size r injected at link L_i towards a target end-host $L_n = A$ with link L_j as its egress link.

Lemma 6: Let L be a sequence of n physical links L_1, L_2, ...L_n with base bandwidths b_1, b_2 , $\dots b_n$ 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 $t_\ell - t_f = \frac{r(s(p)+s(m))}{\min_{1 \le k \le j} b_k}$.

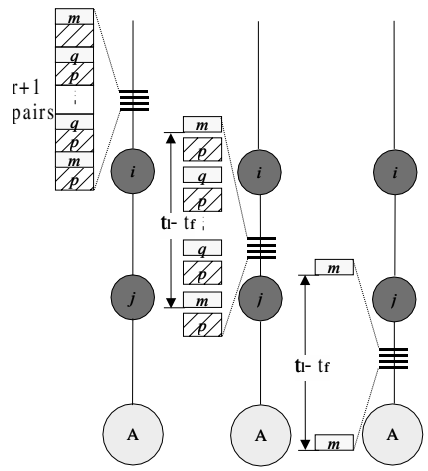

Fig. 2. A cartouche of size r consisting of back-to-back packet pairs of the form $[pm\{pq\}^{r-1}pm]$ injected at L_i (left). Cartouche constituents are spread out over time until they arrive at L_j (middle), where only the first and last markers continue on to target A (right) with an interarrival time $t_\ell - t_f = \frac{r(s(p)+s(m))}{\min_{1\le k\le j} b_k}$.

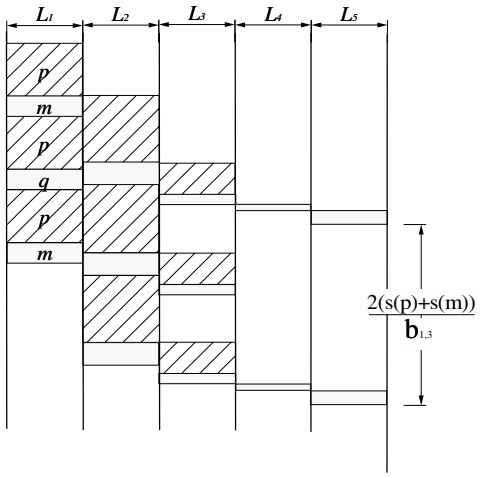

Fig. 3. Illustration of cartouche probing: A cartouche of size $r = 2$ is used to measure $b_{1,3}$. Here, the maximal spacing between marker packets m is introduced at L_2 (which is the slowest link on the subpath $L_1L_2L_3$) and preserved until the markers reach their target L_5.

Lemma 6 provides the most important property of cartouche probing. It defines the interarrival time for the first and last marker packets over every physical link up to the cartouche egress link. Figure 3 illustrates this through a specific example.

Corollary 1: Let L be a sequence of n physical links L_1, L_2, ...L_n with base bandwidths b_1, b_2 , $\ldots b_n$, 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}, $\ldots L_n$ if and only if $\frac{b_{1,j}}{b_{j+1,n}} \le \frac{r(s(p)+s(m))}{s(m)}$.

Corollary 1 follows directly from Lemma 6 and Lemma 4 to derive a sufficient condition for the preservation of markers interarrival times upon exit from the cartouche egress link L_j and throughout the sequence L_{j+1}, $\ldots L_n$. Note that with $s(p) = 1500$ bytes and $s(m) = 40$ bytes, preservation holds even when $\frac{b_{1,j}}{b_{j+1,n}} \le 38.5r$; 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 size.

Lemma 6 and Corollary 1 are all that are needed to provide a solution to the problem of inferring the bottleneck bandwidth of a path prefix. Specifically, this is done by: (1) sizing a cartouche to satisfy the conditions of Corollary 1, (2) setting the cartouche egress link to be L_j, (3) injecting the cartouche 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 $b_{1,j}$.

B. Estimating Bandwidth over a Path Suffix

We now turn our attention to the complementary problem of estimating the bottleneck bandwidth of a path suffix–namely, $b_{i,n}$ for an arbitrary i such that $1 < i \le n$. For the case $b_{1,i-1} \ge b_{1,n}$ the task is trivial since $b_{i,n}$ would be equal to $b_{1,n}$, which can be inferred using the packet-pair technique. In the rest of this section, we concentrate on the case when $b_{1,i-1} < b_{1,n}$ and our approach to estimating $b_{i,n}$ is to attempt to identify the bottleneck link over the subpath of interest $L_i, \ldots L_{i+1}, \ldots L_n$ and estimate that link's bandwidth. We do so using *cartouche trains*.

Definition 3: A *cartouche train* over a set of links $L_1, \ldots L_i, \ldots L_j, \ldots L_n$ is a probe consisting of a sequence of $l = j-i+1$ possibly overlapping cartouches of size r each, whose egress links are $L_i, L_{i+1}, \ldots L_j$, respectively. Link L_i is called the *initial egress link* of the cartouche train and link L_j is called the *final egress link* of the cartouche train. The number l of possibly overlapping cartouches in a cartouche train is called the *length* of the cartouche train.

A cartouche train is completely defined by its length l, by the size r of its constituent cartouches, by its initial (or final) egress link L_i (or L_j), and by its target L_n. For instance a cartouche train of length 2 and of size 3, whose final egress link is L_5 and whose target is L_8 is given by $[p_3mp_4q_4p_4q_4p_4mp_5q_5p_5q_5p_5m]$, where $D(p_w) = D(q_w) = L_w, w = 3, 4, 5$, and $D(m) = L_8$.

Consider a cartouche train of the form $[p_{i-1}mp_imp_{i+1}m \ldots p_nm]$, whose target as well as final egress link is L_n. This cartouche train consists of $(n-i+1)$ overlapping cartouches (each of the form $[p_{w-1}mp_wm]$). Clearly, $D(m) = L_n$ and $D(p_w) = L_w$, which means that all marker packets are targeted to L_n whereas magnifier packets are targeted to successive links starting at L_{i-1}.

Figure 4 shows the composition and progression of such a cartouche train of length $l = 2$ transmitted from L_1 and targeted to L_5 with L_4 (L_5) as its initial (final) egress link. Note that the structure of the cartouche train over the subpath before L_4 (i.e., over L_1, L_2 and L_3) resembles that of a cartouche of size $r = 2$. This means that the interarrival time between any pair of successive marker packets just before the initial egress link L_4 is $\frac{s(p)+s(m)}{b_{1,3}}$. Also, due to the way the

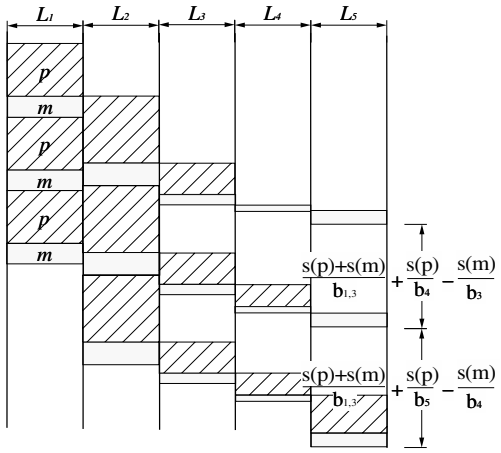

Fig. 4. Illustration of a cartouche train $r = 1$ and $l = 2$ used to estimate $b_{4,5}$.

magnifier packets exit the subpath L_4L_5 on successive links, each of these interarrival times may be updated at only one specific link. For instance, the interarrival time at L_3 between marker packets 1 and 2 can only be altered at L_4. Similarily, the interarrival time between marker packets 2 and 3 can only be altered at L_5. This key property is formulated in the following lemma, which quantifies the interarrival time (Δ_n^k) between the two marker packets immediately preceding and immediately following a magnifier packet egressing at link L_k.

Lemma 7: Let L be a sequence of n physical links L_1, $\dots L_i$, $\dots L_n$ with base bandwidths b_1, $\dots b_i$, $\dots b_n$ respectively, such that $b_{1,i-1} \leq b_{i,n}$. Let $[p_{i-1}mp_im, \dots p_nm]$ be a cartouche train of size $r = 1$ and length $l = n - i + 1$ over L with L_i and L_n as the initial and final egress links, respectively. If $\frac{s(p_w)}{s(m)} < \frac{b_k}{b_{k-1}}$, the interarrival time between the two marker packets immediately preceding and immediately following the magnifier packet egressing at L_k is given by $\Delta_n^k = \frac{s(p_w)+s(m)}{b_{1,i-1}}$, otherwise it is given by $\Delta_n^k = \frac{s(p_w)+s(m)}{b_{1,i-1}} + \frac{s(p_w)}{b_k} - \frac{s(m)}{b_{k-1}}$.

Corollary 2: Let L be a sequence of n physical links L_1, $\dots L_i$, $\dots L_n$ with base bandwidths b_1, $\dots b_i$, $\dots b_n$ respectively, such that $b_{1,i-1} \leq b_{i,n}$. Let $[p_{i-1}mp_im, \dots p_nm]$ be a cartouche train of size $r = 1$ and length $l = n - i + 1$ over L with L_i and L_n as the initial and final egress links, respectively. If *at least one* link k of the subpath L_i, $\dots L_n$ satisfies the tailgating property then $\Delta_n^{k^*} = \max_{i \leq k \leq n}(\Delta_n^k)$ would indicate that L_{k^*} is the subpath bottleneck link and $b_{i,n} = b_{k^*} = \frac{s(p_w)}{\Delta_n^{k^*} - \frac{s(p)+s(m)}{b_{1,i-1}} + \frac{s(m)}{b_{k^*-1}}} \approx \frac{s(p_w)}{\Delta_n^{k^*} - \frac{s(p_w)+s(m)}{b_{1,i-1}}}$.

Corollary 2 spells out how cartouche trains of size $r = 1$ and of length $l = n - i + 1$ could be used to estimate $b_{i,n}$. The approximation in $b_{i,n}$ equation reflects ignoring the transmission delay of one marker packet over L_{k-1}. The small size of the marker packets makes this approximation practically tolerable. Also, notice that if all Δ_n^k are equal to $\frac{s(p_w)+s(m)}{b_{1,i-1}}$ then this signifies that links L_i, $\dots L_n$ are fast enough that none of the marker packets is queued behind its magnifier packet. In this case $b_{i,n}$ cannot be pinpointed and we would then have to rely on an incremental hop-by-hop technique such as pathchar to estimate $b_{i,n}$.

C. Estimating Bandwidth over an Arbitrary Subpath

We are now ready to tackle our main goal of estimating $b_{i,j}$ for arbitrary i, j satisfying $1 \leq i < j \leq n$. First, we observe that using cartouche probing we can measure $b_{1,i-1}$ and $b_{1,j}$. If $b_{1,i-1} > b_{1,j}$ then $b_{i,j} = b_{1,j}$. Otherwise, we need to find out a way to measure $b_{i,j}$. We do so using cartouche trains.

Consider a cartouche train of length $l = j - i + 1$ targeted at L_n with L_i (L_j) as the initial (final) egress link. Clearly, the interarrival times Δ_j^k $i \leq k \leq j$ between the marker packets at L_j can be used to estimate $b_{i,j}$. Thus, the problem of measuring $b_{i,j}$ reduces to figuring out a way of preserving the spacing $\Delta_j^{k^*}$ as it markers travel through links $L_{j+1} \dots L_n$. This can be readily achieved using the results of Lemma 4 which sets the conditions for the preservation of spacing over a subpath.

According to Lemma 4, in order for the marker interarrival time $\Delta_j^{k^*}$ at link L_j to be preserved, the condition $\frac{s(q)}{min\Delta_j^{k^*}} \leq b_{j+1,n}$ must be satisfied. Using $s(p)$=1500 bytes and $s(m)$=40 bytes, the $\Delta_j^{k^*}$ spacing is preserved if $\frac{b_{1,i-1}}{b_{j+1,n}} \leq 38.5$. Notice that this bound is similar to the one we obtained for cartouches of size $r = 1$. In order to preserve $\Delta_j^{k^*}$ over L_{j+1}, $\dots L_n$ even if $\frac{b_{1,i-1}}{b_{j+1,n}} > 38.5$ we need to magnify $\Delta_j^{k^*}$ (as we did for cartouche probing in Section IV-A) using cartouche trains of size $r > 1$. The following Lemma spells this out.

Lemma 8: Let L be a sequence of n physical links L_1, $\dots L_i$, $\dots L_j$, $\dots L_n$ with base bandwidths b_1, $\dots b_i$, $\dots b_j$, $\dots b_n$ respectively. Let $b_{1,i-1} < b_{i,j}$. Given a cartouche train of length $l = j - i + 1$, size r, and with L_i as its initial egress link and L_j as its final egress link, if $\frac{s(p_w)}{s(m)} < \frac{b_k}{b_{k-1}}$ then $\Delta_j^k = \frac{r(s(p_w)+s(m))}{b_{1,i-1}}$, otherwise $\Delta_j^k \approx \frac{r(s(p_w)+s(m))}{b_{1,i-1}} + \frac{s(p_w)-s(m)}{b_k}$, where $i \leq k \leq j$ and $i - 1 \leq w \leq j$.

D. Summary of Measurement Procedure

We conclude this section with a summary of our procedure for measuring the bottleneck bandwidth $b_{i,j}$.

Step 1: Using a packet-pair technique, we measure $b_{1,n}$. This will enables us to appropriately size the cartouches used in later steps (using the results of Corollary 1).

Step 2: Using appropriately-sized cartouches, we measure $b_{1,i-1}$ and $b_{1,j}$ using the relationship established in Lemma 6. If $b_{1,i-1} \geq b_{1,j}$ then $b_{i,j} = b_{1,j}$ and we are done, otherwise we proceed to Step 3.

Step 3: Using an appropriately-sized cartouche train of length $l = j - i + 1$, with initial egress link L_i, we estimate $b_{i,j}$ using Lemma 8. If such estimation is possible, $\Delta_j^{k^*} > \frac{r(s(p_w)+s(m))}{b_{1,i-1}}$, then we are done, otherwise we conclude that our tool cannot accurately measure $b_{i,j}$.

E. Shared Bottleneck Bandwidth

We now 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 S and destined to

two different clients A and B. Estimation of the bottleneck bandwidth over the shared links, is tantamount to computing the bottleneck bandwidth over a path prefix. However, we typically will not have *a priori* knowledge of the length of the shared prefix, nor the IP address of the branching point. One option is to use *traceroute* [17] on the path from S to A, and from S to B, to determine this missing information, but this method is error-prone and inelegant. A more effective approach is to use a cartouche probe, but instead of using *hop-limited* probe packets as in the previous sections, we use *multi-destination* probes instead: Magnifiers are destined to one of the clients, say A, and markers are destined to the other client B. This way, magnifiers and markers travel together only over the shared path and client B can use $\frac{r(s(p)+s(m))}{\Delta}$ as an estimate of the shared bandwidth, where Δ is the interarrival time between the markers at B.

V. Impact of Cross Traffic

In Section IV, we presented an analysis of our end-to-end bottleneck bandwidth estimation procedures. As stated in Section III, the analysis assumes an environment free from cross-traffic, and it is under this idealistic assumption that we prove the various properties of cartouche probing. Clearly, in any practical setting, cross-traffic cannot be ignored. In this section, we present results from extensive simulations intended to characterize the impact of cross traffic on cartouche probing. Our goal in this section is to identify *traffic conditions* under which cartouche probing is and is not effective. We also demonstrate scenarios in which structural characteristics of the *network path* itself impact our results.

Prelude: Recall that cartouche probing relies on the preservation of spacing between marker packets to estimate the bottleneck bandwidth of a path segment at endpoints. In general, cross traffic may impact marker spacing in two possible ways: it may cause *marker compression*, i.e. inter-packet spacing between a pair of markers is reduced in transit, or *marker decompression*, i.e. inter-packet spacing between a pair of markers is increased in transit. Both compression and decompression can result from the arrival of cross-traffic at a link [8].

Marker compression is also possible even in the absence of cross traffic. Recall our constructions in Section IV-A. There, we preserved spacing between the two markers used to measure the bottleneck bandwidth $b_{i,j}$ as the markers travel over subsequent links $L_{j+1} \ldots L_n$. But if $b_{j+1,n}$ is small enough to violate the condition stated in Corollary 1, interpacket spacing is not preserved. To avoid such compression, which is due entirely to the static properties of the path, the size r of the cartouches employed must be increased so as to satisfy the conditions of Corollary 1.

To reduce the effects of cross traffic, a bottleneck bandwidth measurement experiment must be conducted repeatedly and estimates that may have been affected by marker compression or decompression must be identified and excluded using heuristics [8]. All of our methods require the end-host A conducting the experiment to compile a histogram[1] of the frequency of each estimate it obtains. One simple heuristic is to pick the bin with the largest frequency, i.e. the mode. But with a more refined understanding of how marker compression or marker decompression affects our bottleneck bandwidth estimation in specific experiments, we develop better alternative heuristics to simply picking the mode.

From equations in Section IV, one can see that marker compression results in *overestimation* of bottleneck bandwidth, whereas marker decompression results in *underestimation* of bottleneck bandwidth. Moreover, as we will subsequently demonstrate, marker compression due to cross traffic is more prevalent in experiments involving path prefixes, whereas marker decompression is more prevalent in in experiments involving path suffixes and targeted path segments. This suggests that picking the first modality of a histogram in prefix experiments and picking the last modality of a histogram for suffix/subpath experiments are better heuristics to use for filtering the effects of cross traffic. For lack of space, we do not detail in this paper how we automate mode detection and in the experimental results we use the mode as our final estimate.

Experimental Setup: We used the Network Simulator (ns) [27] to simulate a path L connecting two hosts A and B. L consists of 20 physical links L_1, L_2, $\ldots L_{20}$. Link bandwidth values b_1, b_2, $\ldots b_{20}$ and link latencies d_1, d_2, $\ldots d_{20}$ were hand-picked to illustrate various scenarios. Link cross-traffic was modeled by a set of aggregated Pareto ON/OFF UDP flows with 1.8 as the distribution shape parameter, 0.5 seconds as the average burst and idle times, and 32Kb/sec as mean flow rate. Packet sizes of cross-traffic flows were also Pareto with 1.8 as the distribution shape parameter, 200 bytes as the mean, and 1500 bytes as the maximum. By varying the number of cross-traffic flows over each link we control the level of congestion on that link. Probe transmission, time measurements, logging and estimation functions were all performed at host A. In all experiments presented in this section, we use the following settings in the construction of cartouches: $s(p) = 1500$ bytes, $s(m) = s(q) = 40$ bytes.

Path Prefix Experiments: As described in Section IV-A, our technique for measuring the bottleneck bandwidth of path prefix relies on sending a sequence of $[pm\{pq\}^{r-1}pm]$ cartouches from source A, with L_i as the egress link. Host A monitors the interarrival time Δ of the responses to the marker packets m, and uses the formula $\frac{r(s(p)+s(m))}{\Delta}$ to estimate $b_{1,i}$.

Figure 5 shows the histograms we obtain (at host A) trying to infer $b_{1,10}$ using a sequence of 100 $[pm\{pq\}^{r-1}pm]$ cartouches of sizes $r = 1$ (left) $r = 2$ (middle) and $r = 3$ (right). The setup includes 16 cross-traffic flows over each physical link in L, with an actual $b_{1,10} = 50$ Mbps and $b_{11,20} = 1$ Mbps. Examining the results in Figure 5, we observe that the $r = 2$ and $r = 3$ cases lead to a correct $b_{1,10} = 50$ estimate while the $r = 1$ case does not. Lemma

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

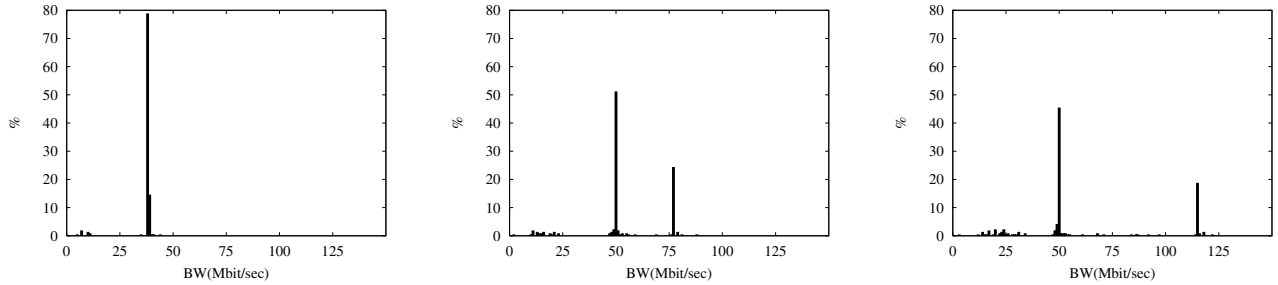

Fig. 5. Histograms of estimated $b_{1,10}$ values after using a sequence of 100 $[pm\{pq\}^{r-1}pm]$ cartouches of dimensionality $r = 1$ (left) $r = 2$ (middle) and $r = 3$ (right). The setup includes 16 cross-traffic flows over each physical link, actual $b_{1,10} = 50$Mbps and $b_{11,20} = 1$Mbps.

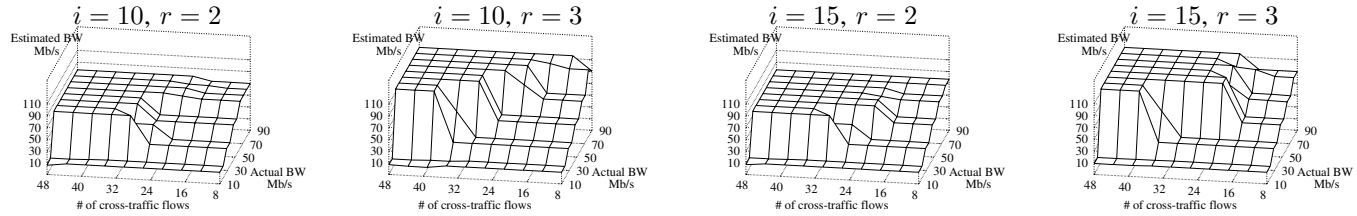

Fig. 6. Grids corresponding to path prefix bandwidth estimates $b_{1,i}$ for different values of i ($i = 10, 15$) and using cartouche sizes of $r = 2, 3$.

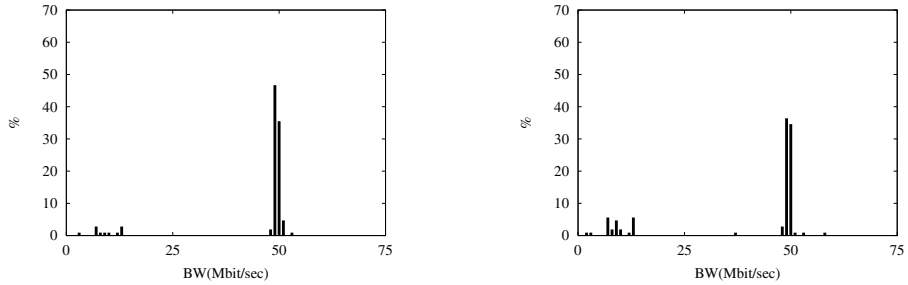

Fig. 7. Histograms of estimated $b_{15,20}$ (left) and $b_{10,20}$ (right) values after using a sequence of 100 cartouche trains. The setup includes 16 cross-traffic flows over each physical link, actual $b_{15,20} = 50$Mbps and $b_{10,20} = 50$Mbps.

6 explains why this happens. 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 endhost. Since $\frac{b_{1,10}}{b_{11,20}} = 50$, the condition holds only for $r = 2$ and $r = 3$, but not for $r = 1$. Thus, when $r = 1$, our cartouches were undersized, resulting in marker decompression and an underestimate of the value of $b_{1,i}$.

The histograms corresponding to $r = 1, 2$ and 3 show few instances of severe underestimation of $b_{1,10}$, indicated by short histogram bars for $0 < b_{1,10} < 38.5$. These are examples of marker decompression due to bursty cross traffic Notice that this decompression is more pronounced for $r = 3$. This is due to the fact that larger cartouches imply longer marker interarrivals, which in turn leads to a higher probability of cross traffic bursts further separating the markers.

The histograms corresponding to $r = 2$ and $r = 3$ show instances of overestimation of $b_{1,10}$. These are examples of marker compression due to bursty cross traffic . Notice that our overestimates of $b_{1,10}$ were capped at 77Mbps and 115.5Mbps, for $r = 2$ and $r = 3$, respectively. Again, this is a direct consequence of the inequality in Lemma 6, which in effect specifies an upper bound on the maximum observable value for $b_{1,i}$ using cartouches of size r. This bound (confirmed in Figure 5) is $r * b_{i+1,n} * 38.5$, which is 38.5Mbps and 77Mbps and 115.5Mbps for $r = 1, 2$ and 3 respectively.

Figure 6 represents 3-dimensional grids plotting the estimated $b_{1,i}$ values (the modes) when we vary the number of cross-traffic flows over each link and for different actual $b_{1,i}$ values, while always keeping $b_{i+1,20} = 1$Mbps. These plots are given for prefix lengths of $i = 10$ and $i = 15$ and for cartouche sizes $r = 2$ and $r = 3$. By carefully inspecting the resulting grids, one can make the following observations. First, the results confirm that the maximum possible $b_{1,i}$ value we can estimate (given the sizes we picked for m, p, and q packets) is 77Mbps and 115.5Mbps for $r = 2$ and $r = 3$ 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 are to interarrival time alterations due to cross-traffic. This advocates using larger values of r. Third, incorrect estimates due to marker compression are much more significant than those resulting from marker decompression. Finally, prefix length (the value of i) does not seem to pose any significant impact on the accuracy of our techniques.

Path Suffix Experiments: As described in Section IV-B, our technique to measure path suffix bottleneck bandwidth relies on sending a cartouche train of size $r = 1$ and of length l equal

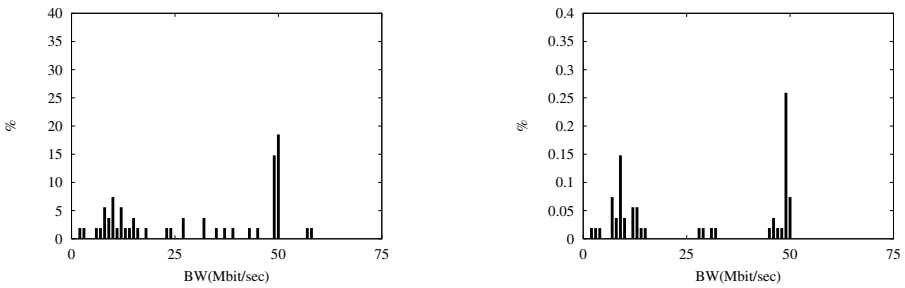

Fig. 8. Histograms of estimated $b_{5,10}$(left) and $b_{5,15}$(right) values after using a sequence of 100 cartouche trains of dimension $r = 2$. The setup includes 16 cross-traffic flows over each physical link, actual $b_{5,10} = 50$Mbps and $b_{5,15} = 50$Mbps.

to the suffix length. As before, host A monitors the interarrival time Δ of the responses to markers m, then uses the largest Δ between any pair of successive marker packets, $\Delta_n^{k^*}$, and $b_{1,i-1}$ to estimate $b_{i,n}$.

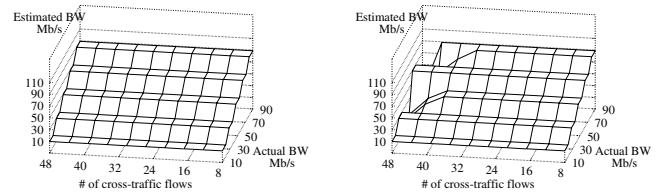

Fig. 9. Grids corresponding to path suffix bandwidth estimates $b_{i,20}$ for $i = 15$ (left) and $i = 10$ (right).

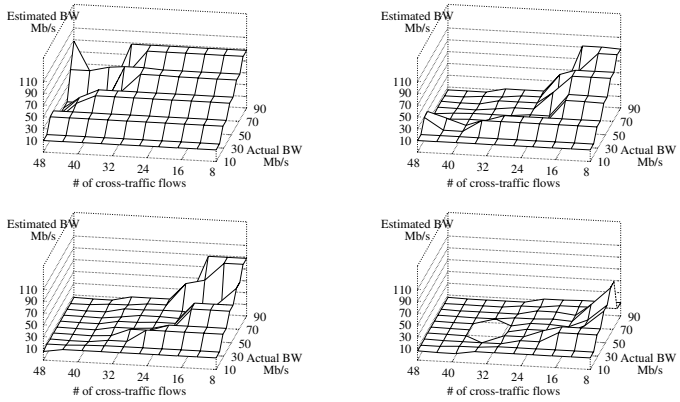

Fig. 10. Grids corresponding to arbitrary sub-path bandwidth estimates $b_{i,j}$ for i=5, $j = 10$ (top), $j = 15$ (bottom), and using $r = 2$ (left), $r = 3$ (right).

Figure 7 shows the histograms we obtain when estimating $b_{15,20}$ (left) and $b_{10,20}$ (right) after using a sequence of 100 cartouche trains of length $l = 5$ and $l = 10$, respectively. The setup includes 16 cross-traffic flows over each physical link in L, with actual $b_{15,20} = 50$Mbps and $b_{10,20} = 50$Mbps, with the overall path bottleneck $b_{1,20}$ being set to 1Mbps. Clearly, in both cases, the mode represents the correct bandwidth estimate. The figure also shows that incorrect estimates due to marker decompression are non-negligible, whereas those due to marker compression are virtually non-existent. In addition, incorrect estimates due to marker decompression are more prevalent for longer cartouche trains. This is in sharp contrast to our results for prefix bandwidth estimation, in which we have shown that incorrect estimates due to marker compression are more prevalent, and in which the prefix length does not play a significant factor.

Figure 9 shows grids from experiments estimating $b_{15,20}$ (left) and $b_{10,20}$ (right) under a range of cross-traffic and actual bandwidth conditions. These grids confirm that as cross traffic increases, marker decompression becomes more significant, and that this is more prevalent for long subpaths.

Arbitrary Path Segments Experiments: As described in Section IV-C, our technique to measure arbitrary subpath bandwidth relies on sending a sequence of appropriately-sized cartouche trains from source A. Host A monitors the interarrival time Δ of the responses to markers m, then uses the largest Δ between any pair of successive marker packets, $\Delta_n^{k^*}$, and $b_{1,i-1}$ to estimate $b_{i,j}$.

Figure 8 shows the histograms we obtain trying to infer $b_{5,10}$ (left) and $b_{5,15}$ (right) after using a sequence of 100 cartouche trains of size $r = 2$. The setup includes 16 cross-traffic flows over each physical link in L, with actual $b_{5,10}$ = 50Mb/s, $b_{1,4}$ = 1Mb/s, $b_{11,20}$ = 5Mb/s (left) and $b_{5,15}$ = 50Mb/s, $b_{1,4}$ = 1Mb/s, $b_{16,20}$ = 5Mb/s (right). The histograms show that both $b_{5,10}$ and $b_{5,15}$ are correctly estimated. It also shows that understimates (due to marker decompression) are quite prevalent. As we observed in our suffix experiments, which uses the same cartouche train construction, marker decompression is more pronounced when the subpath length l is longer.

Figure 10 shows the grids resulting from experiments to estimate $b_{5,10}$ and $b_{5,15}$. These confirm that it is desirable to use cartouche trains of the smallest possible size r and suggest that for long subpaths, it is not advisable to simply use long cartouche trains. A better divide-and-conquer alternative may be to partition a long subpath into segments, to which shorter cartouche trains could be applied.

Postlude: We conclude this section with a summary of our findings regarding the susceptibility of our constructions to cross traffic. Specifically, we observe that: (1) Marker compression and hence overestimation of $b_{1,i}$ presents the most significant hurdle for bottleneck bandwidth estimation of path prefix using cartouches. (2) Marker decompression and hence

underestimation of $b_{j,n}$ presents the most significant hurdle for bottleneck bandwidth estimation of path suffix (or arbitrary path segments) using cartouche trains. This difficulty can be alleviated through the use of the smallest cartouches that would satisfy the structural constraints imposed by Lemma 6, and through a divide-and-conquer approach.

VI. Internet Measurement Experiments

In this section, we compare cartouche probing performance and efficiency to those of existing hop-by-hop techniques (namely `pchar` and `nettimer`). We also present Internet measurement results intended as a proof of concept of the efficacy of the cartouche probing technique.

A. Comparison to `pchar` and `nettimer`

Both `pchar`[25] and `nettimer` [24] are hop-by-hop techniques which means that in order to estimate the bottleneck bandwidth along a path segment they need to incrementally run tests to estimate the bandwidth of every hop in the segment. Also, these techniques are cumulative, in the sense that estimates for any hop depend on estimates made for previous hops in the segment. On the other hand, cartouche probing directly targets the bottleneck bandwidth of the segment avoiding the propagation of erroneous results.

In comparing against `pchar` and `nettimer`, we consider two measures: (1) *Time efficiency*: which reflects the time it takes to return a bandwidth estimate, and (2) *Byte efficiency*: which is a measure of the amount of traffic injected into the network to get an estimate. In terms of time efficiency, cartouche probing is more efficient since it does not have to orchestrate a round of probing for every hop in a segment before returning the final estimate. In terms of byte efficiency, cartouche probing is more efficient than `pchar`, which uses linear regression in its statistical analysis for every hop, thus requiring it to inject quite a bit of extra traffic.[2] Cartouche probing byte efficiency is comparable to that of `nettimer`.[3] Careful inspection of the cartouche constructions reveal that the byte requirements of a sequence of cartouche probes is $k(r+1)(s(p)+s(m))$ bytes, where k is the number of probes in the sequence, and the byte requirements of a sequence of cartouche trains is $k(lr+1)(s(p)+s(m))$ bytes, where l is the path segment length.

B. Experimental Setup

Two Internet paths (connecting our laboratory at Boston University to Georgia Tech in the US and Ecole Normale Superieure in France) were handpicked to demonstrate the different scenarios that we discussed in section V. Figure 11 shows that these two paths share the first three hops and then diverge. The labels on the figure reflect the *a priori*-known bottleneck bandwidth of links in our own laboratory, links of Internet2 hops published by Abilene [26], and links on the far end of the paths obtained through personal contacts.

[2]In fact, `pchar` default settings inject more than 35MB in the network *per hop*, whereas cartouche probing typically injects much less than 1MB.

[3]A cartouche of size r has as many bytes as $r+1$ `nettimer` tailgated pairs and a cartouche train of length l and size r has as many bytes as in $(r+1)l-1$ tailgated pairs.

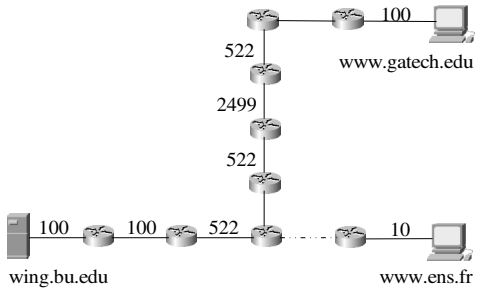

Fig. 11. The Internet paths used in our experiments. Labels are *a priori*-known link bandwidths. All units are in Mbps.

To evaluate our mechanisms, we incorporated our cartouche probing functionality into PeriScope[15], a Linux API providing a flexible interface to define arbitrary probing structures. This allows the cartouche probing transmissions to be orchestrated from the Linux kernel, ensuring back-to-back transmissions and accurate packet timestamping.

We installed PeriScope in our laboratory on a Pentium IV processor, running RedHat Linux 2.2.19 over a 100Mbps LAN. In all experiments, we use magnifier probe packets of size 1500 bytes and marker packets of size 60 bytes. Experiments were conducted once per second until we obtain 100 valid probe replies. Both packet reordering and packet loss invalidated a reply.

Hop	Actual BW	Cartouche	Pchar
1	100	90	70
2	100	85	78
3	522	640	604
5	522	680	?
6	2499	4400	?
7	522	850	1319
9	100	60	49

TABLE I

Results from cartouche probing and `pchar` for *a priori*-known links of the path to Georgia Tech.

C. Results

Figure 12 shows the histograms we obtained when using cartouche probing to estimate b_1, $b_{1,3}$, b_5 for the path to Georgia Tech and the bandwidth of the closest link for Ecole Normale. The figure shows that the estimated values are close to the *a priori*-known bandwidth values. Table I compares cartouche probing and `pchar` estimates over the *a priori*-known links of the path to Georgia Tech.[4] Note that `pchar` does not return an estimate for some links. Apart from that, the estimates are comparable. While the cartouche probing estimate for the OC-48 link on hop 6 is not highly accurate, `pchar` does not return an estimate for that link at all.

VII. Conclusion

We have described an end-to-end probing technique capable of inferring the bottleneck bandwidth along an arbitrary path segments in the network, or across the portion of a path

[4]`nettimer` documentation does not specify how to use the tool to estimate the bandwidth of every hop along a path; however, the techninque details can be found in [24].

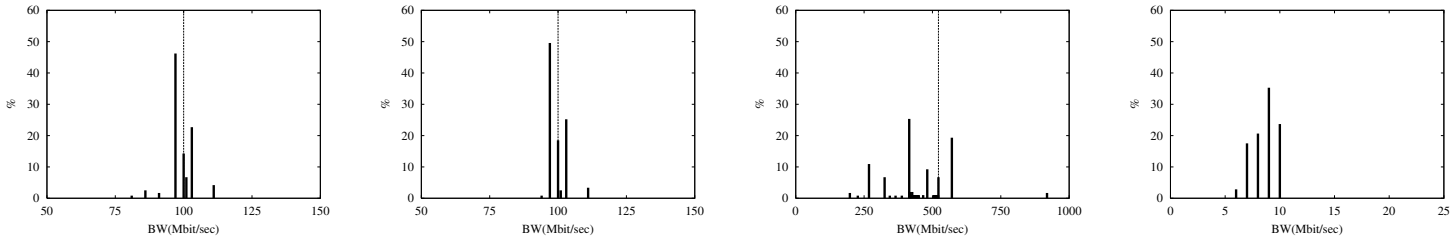

Fig. 12. Histograms of estimated bottleneck bandwidth along different segments of the paths to Georgia Tech, and Ecole Normale Superieure. The histograms (from left to right) represent bandwidth estimates for Georgia Tech b_1, $b_{1,3}$, b_5 and for the home network link of Ecole Normale Superieure. Dotted lines represent *a priori*-known bandwidth values.

shared by a set of connections, and have presented results of extensive simulations and preliminary Internet measurements of our techniques. The constructions we advocate are built in part upon 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.

While the end-to-end probing constructions we proposed in this paper are geared towards a specific problem, we believe that there will be increasing interest in techniques which conduct remote probes of network-internal characteristics, including those across arbitrary subpaths or regions of the network. We anticipate that lightweight mechanisms to facilitate measurement 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.

Acknowledgments: This work was done while Khaled Harfoush was at the Computer Science Department of Boston University, and was partially supported by NSF grants ANI-9986397, CAREER ANI-0093296, ANI-0095988, EIA-0202067, and ITR ANI-0205294.

REFERENCES

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

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

[3] J. Byers, J. Considine, M. Mitzenmacher, and S. Rost. Informed Content Delivery Across Adaptive Overlay Networks. In *Proceedings of ACM SIGCOMM '02*, Pittsburgh, PA, August 2002.

[4] 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.

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

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

[7] 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)*, October 1999.

[8] C. Dovrolis, P. Ramanathan, and D. Moore. What Do Packet Dispersion Techniques Measure? In *INFOCOM '01*, Anchorage AK, April 2001.

[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] M. Goyal, R. Guerin, and R. Rajan. Predicting TCP Throughput From Non-invasive Network Sampling. In *Proceedings of IEEE INFOCOM '02*, June 2002.

[12] K. Hanna, N. Natarajan, and B. Levine. Evaluation of a Novel Two-Step Server Selection Metric. In *9th International Conference on Network Protocols (ICNP)*, Riverside, CA, November 2001.

[13] K. Harfoush. *A Framework and Toolkit for the Effective Measurement and Representation of Internet Internal Characteristics*. PhD thesis, Boston University, June 2002.

[14] 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)*, November 2000.

[15] K. Harfoush, A. Bestavros, and J. Byers. `PeriScope`: An Active Probing API. In *Proc. of the 2002 Passive and Active Measurement Workshop, PAM '02*, Fort Collins, Colorado, March 2002.

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

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

[18] M. Jain and C. Dovrolis. End-to-end Available Bandwidth: Measurement Methodology, Dynamics, and Relation with TCP Throughput. In *Proceedings of ACM SIGCOMM '02*, August 2002.

[19] 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.

[20] 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.

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

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

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

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

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

[26] The Abilene Network Logical Map. http://www.abilene.iu.edu/images/logical.pdf, January 30 2002.

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

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

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

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

[31] 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.

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

[33] 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*, San Diego, CA, August 2001.