M&M: A Passive Toolkit for Measuring, Tracking, and Correlating Path Characteristics

Sachin Katti MIT CSAIL
Dina Katabi MIT CSAIL
Chuck Blake MIT CSAIL
Eddie Kohler UCLA/ICIR
Jacob Strauss MIT CSAIL

ABSTRACT

This paper presents *M&M*, a passive measurement toolkit suitable for large-scale studies of Internet path characteristics. The `multiQ` tool uses *equally-spaced mode gaps* in TCP flows' packet interarrival time distributions to detect multiple bottleneck capacities and their relative order. Unlike other passive tools, `multiQ` can discover up to three bottlenecks from the trace of a single flow, and can work with acknowledgment as well as data interarrivals. M&M also contains `mystery`, a TCP loss and RTT analyzer designed to work in concert with `multiQ`. The M&M toolkit can *measure* path characteristics and *correlate* different types of measurements of the same path, producing new results. Since M&M is passive, it can also use publicly-available traces to *track* the evolution of measurement results over time.

We validate our tools in depth using the RON overlay network, which provides more than 400 heterogeneous, well-understood Internet paths. We compare `multiQ` with Nettimer and Pathrate, two other capacity measurement tools, in the first large-scale wide-area evaluation of capacity measurement techniques. Our results show that `multiQ` is highly accurate, and though passive, achieves the same accuracy as Pathrate, which is active.

We also use our toolkit to perform several measurement studies using a large reservoir of traced packets collected by NLANR on backbone links over the last two years. Among the results are that bottleneck capacity on these links has grown by around an order of magnitude from 2002 to 2004. In contrast to expectations, however, fair bandwidth share does not increase much with increased bottleneck capacity.

1 INTRODUCTION

Researching the Internet—building simulation and emulation scenarios—depends on a mental model of how the network really behaves. For example, we might assume that flows traverse a single congested link, that the level of statistical multiplexing on bottleneck links is low, that there is no congestion on the reverse path, that the fair bandwidth share increases with increased bottleneck capacity, and so forth. These assumptions must arise from a faithful representation of the current state of the network, or how the network may be expected to behave in the future. Research based on unrealistic representations has little to say about how the actual network should evolve [17].

How, then, to create a faithful representation of the current Internet? The best answer is to measure those properties important for a given research question, in the widest range of expected conditions, and extract relevant parameters from the results. This kind of measurement is challenging. Active measurements, used by many applications, become difficult on such large scales because of probe overhead and the need to avoid perturbing the characteristics being measured. Also, active measurements often assume access to both the sender and receiver, limiting their applicability to a few paths; and they cannot use the large set of Internet traces collected over the years, making it difficult to see how the Internet has evolved over time. A comprehensive set of *accurate, passive, trace-based* measurement tools is therefore required.

This paper presents M&M, a toolkit for the passive measurement of path characteristics. M&M centers on a novel passive multi-bottleneck capacity measurement tool, `multiQ`. It also contains a multi-function TCP analyzer, `mystery`, whose results are designed for correlation with the bottleneck capacities measured by `multiQ`. Both tools analyze medium-to-long TCP flows contained in trace files (although they could be used in active applications). Because M&M expands the set of properties that can accurately be measured with passive techniques, and because it facilitates the correlation of basic measurements into higher-level measurement results, we believe it represents an important step towards our eventual goal: the creation of more faithful representations of the Internet, and the evaluation of the representations we already use.

`multiQ` uses packet interarrivals to investigate questions about the link capacities along a path. Its novel insight is that packet interarrival times, shown as a distribution, demonstrate *equally-spaced mode gaps* caused by intervening cross traffic packets on bottleneck links in the path. `multiQ` is both passive and precise. Unlike earlier capacity-measurement work [36, 28, 12, 30, 5], it can passively discover capacities from sender-side traces as well as receiver-side traces; uniquely for passive tools, it can discover the capacities and relative order of up to three bottleneck links along a path from a single TCP flow trace.

`mystery` reports TCP loss events, lost packets, and fine-grained semi-RTT measurements throughout the duration of a flow using techniques from previous work. Together with `multiQ`, this allows us to correlate path characteristics—capacity, number of bottleneck links, the existence of reverse path bottlenecks—with TCP performance. We validate `multiQ` and `mystery` using nearly 10,000 experiments over the RON overlay, which provides more than 400 heterogeneous paths with detailed information about their characteristics.

We use M&M to measure a few transport-centric path properties; correlate different types of measurement of the same path,

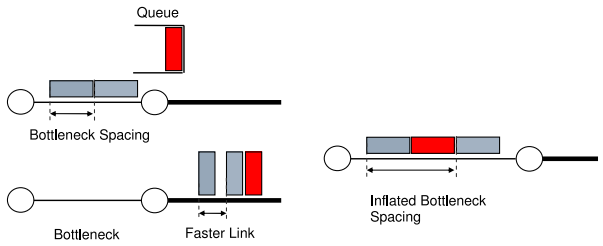
producing new kinds of results; and track the evolution of the measurements over multiple years. Using a large (375-million-packet) reservoir of 258 diverse NLANR traces collected in the backbone over the last two years, we address the following questions: How has the distribution of Internet bottleneck capacity changed over time? Does the TCP fair bandwidth share increase with increased bottleneck capacity? Does the drop rate seen by a TCP depend on the capacity of the link where it is bottlenecked?

1.1 Contributions

This paper has the following contributions.

- We introduce EMG (Equally-spaced Mode Gaps), a new passive technique for inferring link capacities from packet inter-arrival times. Prior work has inferred link capacity from the location of the modes in the packet inter-arrival distribution [28, 23, 15, 35]. In contrast, EMG uses the distance between consecutive modes. We show that the EMG technique is more robust to error caused by cross traffic and can discover the capacity of multiple congested links and their relative order along the path, from a `tcpdump` trace of a single flow.
- We correlate path capacity with transport-centric behavior. Our analysis of a large trace library collected by NLANR on backbone links shows that significant flows (see Table 1) have experienced a dramatic increase in bottleneck capacity between 2002 and 2004, but the higher capacity bottlenecks did not result in a larger per significant flow bandwidth share. Further, the drop rates experienced by significant flows on low capacity paths are considerably similar to the drop rates seen on high capacity paths.
- This paper presents the first wide-scale Internet evaluation of recent advancements in capacity measurement. Using over 10,000 experiments on 400 heterogeneous Internet paths with known likely capacities, we evaluate `multiQ`'s accuracy and compare it with Nettimer [28], another passive capacity measurement tool, and Pathrate [15], an active tool. Our results confirm that link capacity measurement tools are mature and accurate, with more than 85% of their measurements within 10% of their correct value. `multiQ` has the same accuracy as Pathrate, which is an active tool. Compared to Nettimer, `multiQ` is more accurate when both tools have access only to receiver-side traces. Nettimer can be as accurate as `multiQ` only when it is given access to both ends of the measured path.
- `multiQ` is the first tool that can effectively extract capacity information solely from sender-side traces (i.e., ack traces).[1]
- We reveal subtle differences between active and passive capacity estimation tools, showing that they may disagree on the capacity of a particular path and yet both be correct. For example, when traversing a path with a leaky bucket limiter, Pathrate measured the bucket's token rate while Nettimer and `multiQ` measured the actual capacity of the link.

[1]Nettimer can theoretically work on sender-side traces but its accuracy is extremely low; less than 10% of the measurements are within 20% of their correct values.



(a) Compression **(b) Inflation**

Figure 1—Cross traffic impact on packet pair measurements. (a) Cross traffic (dark) compresses the interarrival times of probe packets; (b) cross traffic intervenes between consecutive probe packets, inflating their interarrival time.

- We uncover the distribution of cross-traffic bursts, which intervene between consecutive packets in a medium to moderate size TCP flow, and discuss the implications of this distribution for passive measurements.
- We have implemented our toolkit, M&M, as a software package which we intend to make publicly available under an open source license by final publication.[2]

Table 1 defines several important terms used throughout the paper. Particularly, we define a bottleneck as a link at which traffic faces persistent queuing.

2 Capacity Estimation with EMG

We begin by explaining the operation of our capacity-estimation tool, `multiQ`, and its underlying basis: the equally-spaced mode gaps (EMGs) induced by cross traffic on packet interarrival time distributions.

2.1 Cross Traffic: Noise or Data?

The *packet pair* technique has traditionally been used to infer the minimum capacity along a path. A sender emits probe packets back-to-back; assuming cross traffic does not intervene, the probes arrive spaced by the transmission time on the bottleneck link. The capacity of the bottleneck is computed as

$$C = \frac{S}{T}, \tag{1}$$

where S is the size of the second probe and T is the time difference between the arrivals of the two packets at the receiver (their *interarrival time*).

Cross traffic can cause substantial errors in packet pair-based capacity estimates [15] by changing the interarrival time between probes. Compression errors happen when the first packet of a probe packet pair gets delayed more than the second packet due to getting queued up behind cross traffic at a downstream congested link. (Figure 1a); inflation errors occur when cross-traffic packets intervene between the probe packets upstream from the bottleneck link (Figure 1b). To eliminate these cross-traffic effects, prior work sends trains of packets (packet bunch mode) [39] or a variety of packet sizes [14]; uses the global

[2]Unlike `multiQ`, the algorithms under-lying mystery rely heavily on prior work and do not constitute a contribution. We implemented `mystery` to correlate the output of `multiQ` with TCP performance. We wrote our own tool because existing tools either had no source code available, or produced less fine-grained results than we wanted to use.

Term	Definition
Significant flow	A TCP flow that achieves an average packet rate $>$ 10 pps ($\approx$ 1 pkt/RTT), contains at least 50 packets, and has an MTU of 1500 bytes. (The vast majority of medium-to-long data flows have this MTU.)
Cross-traffic burst	Traffic intervening between two consecutive packets of a traced flow
Bottleneck	Link where traffic faces queuing
Capacity	The maximum rate at which packets can be transmitted by a link
Narrow link	The link with the smallest capacity along a path
Tight link	The link with minimum available or unused capacity along a path
Path capacity	Capacity of the narrowest link on that path
Statistical Multiplexing	No. of flows sharing a bottleneck link
Semi-RTT	Latency between the sending of a data packet and the reception of the *ack* corresponding to that packet as they appear in the trace

Table 1—Definitions of the terms used in this paper.

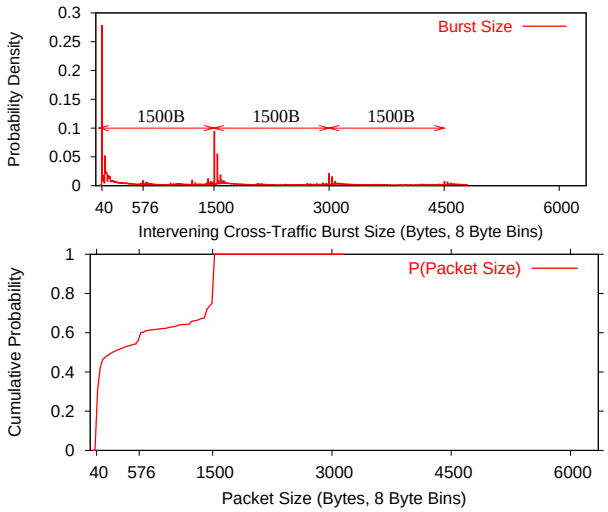


Figure 2—(a) Distribution of cross traffic between consecutive packets in a significant flow has equal mode gaps of 1500 bytes. (b) The CDF of packet size reveals frequencies of 40- and 1500-byte packets.

mode in the interarrival histograms [28]; and so forth. Yet, as the bottleneck becomes more congested, eliminating the effect of cross traffic becomes more challenging, particularly with passive measurements, where one cannot control the rate and sending times of the analyzed TCP flow.

Given this, is it possible that cross-traffic effects contain any useful information, rather than just being noise? We demonstrate that cross traffic, with proper interpretation, actually helps detect not only the minimum capacity along the path, *but also the capacities of other congested links.*

We define a *cross-traffic burst* to be the traffic that intervenes between two consecutive packets of a flow. We seek to understand the probability distribution of different cross-traffic burst sizes: that is, the chance that a given amount of traffic will intervene between consecutive packets of a flow, at a congested link. We have studied 375 million packets in 258 NLANR traces, collected at 21 backbone locations, with a total of about 50,000 significant flows.[3] The diversity and size of this data set makes it a plausible sample of the Internet. We have identified all significant flows in these traces. For each significant flow, we have considered the rest of the traffic in the trace as cross-traffic. For each pair of packets in a significant flow, we have computed

[3]Section 6 describes this dataset further.

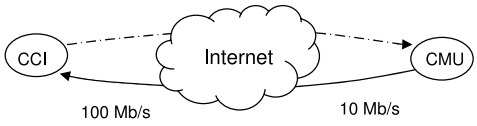


Figure 3—The experiment that generated the graphs in Figure 4.

the intervening cross-traffic burst at the link where the trace is taken. This was repeated for each significant flow in the trace and the resulting samples for cross-traffic bursts were collected. Figure 2a shows the distribution of the sizes of these bursts.

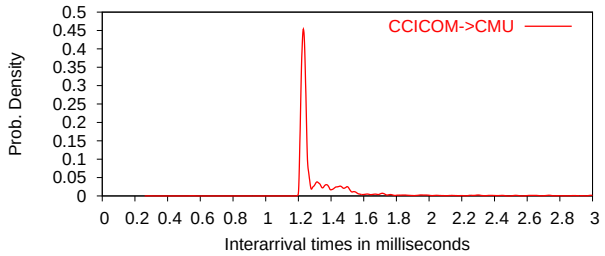
Note the surprising regularity: sharp modes separated by equal gaps of 1500 bytes. This structure is caused by the distribution of Internet packet sizes. Figure 2b shows a cumulative distribution function (CDF) of packet sizes in these traces, which replicates previously reported results [44]. The dominant sizes are 40 and 1500 bytes; many other sizes are represented, but none are as highly pronounced. Thus, we would expect that the modes in the burst distribution will stem from 40- and 1500-byte packets; and since 1500-byte packets are so much larger than 40-byte packets, their size should dominate the modes in Figure 2a. The 40-byte packets broaden the 1500-byte modes, and less common sizes create the bed of probability under the modes.

How will these modes be reflected in passive measurements that might not see the physical cross traffic? Once the measured flow reaches a point of congestion, i.e., a queue, the idle intervals squeeze out, and the packets (of both our flow and cross traffic) compress nearer in time. Thus, provided subsequent links are uncongested, the interarrival times observed at the receiver are proportional to the sizes of cross-traffic bursts on the congested link. Since the PDF of cross-traffic burst size contains modes separated by 1500 bytes, *we expect the PDF of interarrival times in a flow to have modes separated by the transmission time of 1500 bytes at some bottleneck link.*

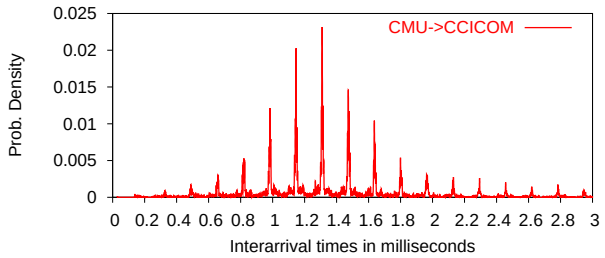
The rest of this section expands this insight into a practical measurement technique.

2.2 Examining an Interarrival PDF

We motivate our work by describing the outcome of a simple experiment. We examine the path connecting two machines: one at CMU with a 10 Mb/s access link, and one at CCICOM with a 100 Mb/s access link (Figure 3). The path between the two machines traverses 18 Internet hops. We first download a large file



(a) Flow from CCICOM to CMU



(b) Flow from CMU to CCICOM

Figure 4—Interarrival PDFs for CCICOM–CMU path in both directions.

Link capacity	Transmission time
380 Kb/s (DSL)	32 ms
1 Mb/s	12 ms
1.5 Mb/s (T1)	8 ms
10 Mb/s	1.2 ms
45 Mb/s	0.267 ms
100 Mb/s	0.12 ms
155 Mb/s	0.08 ms
622 Mb/s	0.018 ms

Table 2—Transmission times of 1500-byte packets on various capacity links.

from CCICOM to CMU while collecting a `tcpdump` trace at CMU. Figure 4a shows the interarrival PDF for this significant flow. The distribution shows a single spike at 1.2 ms, which is the transmission time of a 1500-byte packet on a 10 Mb/s link. There is nothing special about this PDF; 10 Mb/s is the minimum capacity link along the path, and the spike in the PDF shows that most packets were queued back-to-back. Normal packet-pair techniques would have worked well on this trace.

Next, we repeat the experiment along the reverse path: we download a large file from CMU to CCICOM and plot the interarrival distribution as seen by `tcpdump` at CCICOM. The result, shown in Figure 4b, has an interesting structure. The envelope of the distribution is again centered near 1.2 ms, because of the upstream 10 Mb/s link; but it is modulated with sharp spikes separated by *equally-spaced mode gaps* (EMGs) of 0.12 ms, which is the transmission time of a 1500-byte packet on a 100 Mb/s link.

To understand this PDF, consider what happens to packets as they go from CMU to CCICOM. As packets traverse the 10 Mb/s CMU access link (which is also the narrow link along the path), they become spaced by 1.2 ms, the transmission time of one packet on that link. In this case, the Internet backbone is not congested (most queuing happens at access links to stub domains [18]), so the interarrivals remain relatively unperturbed until they reach the 100 Mb/s CCICOM access link, where the flow faces congestion again. There, the spacing of two consecutive packets in our flow changes in one of three ways, as follows.

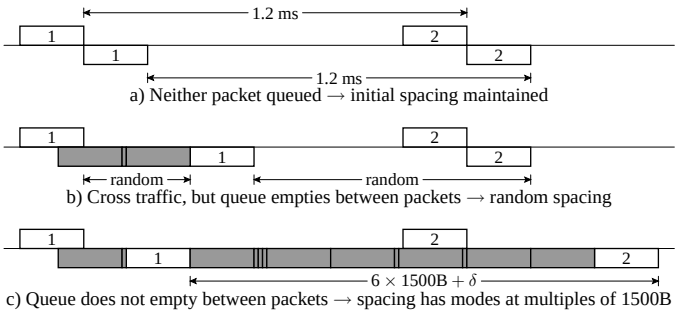


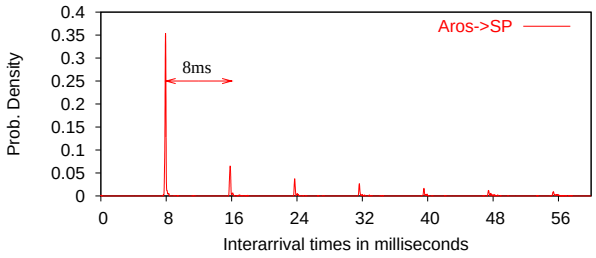
Figure 5—Various cases of packet spacing on CCICOM's access link. Above the line represents arrivals; below the line represents departures. Light packets are from the traced flow and dark packets are cross traffic. Cross traffic arrivals are not shown.

(a) *Neither packet is queued* (Figure 5a). The time between the trailing edges of the two packets—their interarrival—remains 1.2 ms.

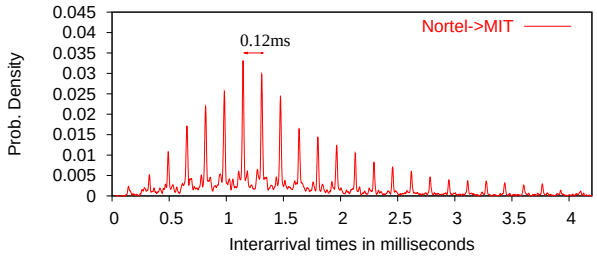
(b) *Either packet is queued and the queue empties between the departure time of the two packets.* Figure 5b shows an example where the first packet arrives while a cross-traffic packet is in the process of being transmitted. The packet has to wait for that transmission to finish, plus any remaining cross-traffic packets in the queue. This waiting time takes values spread over a wide range, depending on the total number of bytes that must be transmitted before our packet. Since the initial cross-traffic packet was *partially* through transmission, this number of bytes will show no pronounced modes. If one assumes the second packet is not queued, then the interarrival time becomes 1.2 ms minus the delay of the first packet. Interarrival samples of this type are spread over a wide range with no pronounced values or modes, and contribute to the bed of probability under the spikes in Figure 4. A similar argument applies if the second packet is queued and the first is not, or even if the two packets are both queued, as along as the packets belong to different queuing epochs (the queue empties between their departures).

(c) *Either packet is queued and the queue does* not *empty between the departure times of the two packets* (Figure 5c). The resulting interarrival is the transmission time of the intervening cross-traffic burst plus the second packet in the pair. As we have seen, cross-traffic bursts have modes at multiples of 1500 bytes, so interarrival samples of this type will show modes spaced by 0.12 ms (the transmission time of 1500 bytes on 100 Mb/s). The input interarrival of 1.2 ms is a factor of 10 higher than this mode spacing, so these modes will be centered around 1.2 ms unless the queuing is extremely bursty.

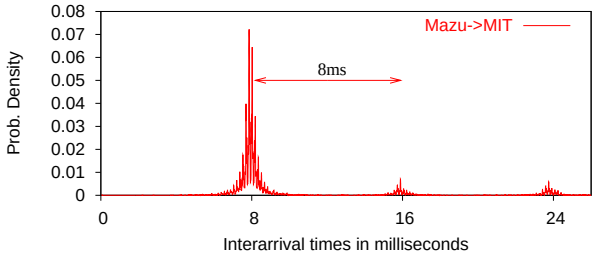
Figure 4b also shows some symmetry in the modes around 1.2 ms. We have argued that modes are caused by interarrivals of type (c), where a cross-traffic burst is queued between our packets with no idle period. Our traced packets arrive at the CCICOM queue equally spaced by 1.2 ms. If cross-traffic effects stretch a pair of packets in the traced flow, the resulting interarrival sample will lie to the right of the 1.2 ms mode. If cross-traffic effects squeeze the pair, the interarrival sample lies to the left of the 1.2 ms mode. On this link, it seems that the probability of squeezing and stretching were close.



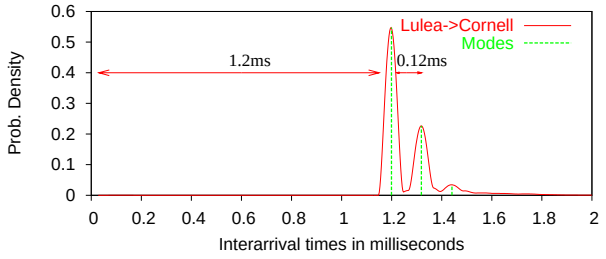
(a) A single congested T1 link.



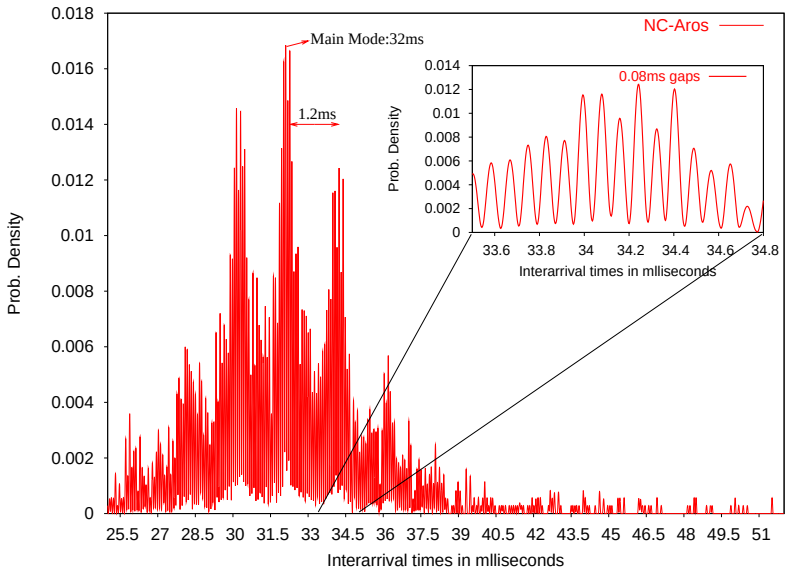
(b) Upstream 10 Mb/s and a downstream highly congested 100 Mb/s.



(c) Upstream congested T1 and downstream 100 Mb/s.



(d) Upstream highly congested 100 Mb/s and downstream 10 Mb/s.



(e) PDF shows 3 bottlenecks. The envelope peaks at 32 ms, indicating an upstream 380Kb/s link, mode gaps at 1.2 ms correspond to the 10 Mb/s downstream link and mode gaps of 0.08ms in the zoomed figure show a 155Mb/s bottleneck.

Figure 6—Example interarrival PDFs. All show equally spaced mode gaps (EMG).

This simple experiment teaches us two lessons: **(1)** Equally-spaced mode gaps in a flow's interarrival PDF correspond to the transmission times of 1500-byte packets on some bottleneck along the path. **(2)** The envelope of the PDF describes the minimum-capacity congested link along the path, whose output gets modulated by downstream congested links.

2.3 Interarrival PDF Variations

Inspection of interarrival PDFs for over 400 different Internet paths from the RON testbed (see Section 5 for a description) shows that most PDFs exhibit equally-spaced mode gaps separated by the transmission time of a 1500-byte packet on a well-known link capacity—see Table 2 for a list. For lack of space we show only a few PDFs, chosen to expose the various possible shapes.

Figure 6a shows an interarrival PDF for a flow going from a 100 Mb/s access link to a T1. (We know the access link capacities of all nodes in the RON testbed.) The downstream low-capacity T1 link creates EMGs of 8 ms and erases the spacing produced by the upstream bottleneck. In most cases, we are only able to see secondary bottlenecks *downstream* of the minimum-capacity link, since the minimum-capacity link destroys any upstream spacing. The large number of modes shows that the bottleneck link had a high degree of queuing/congestion. Lower capacity bottlenecks usually obscure upstream higher capacity bottlenecks. To discover these bottlenecks, one needs to examine the ack inter-arrival PDF. Fortunately, the EMG technique works on ack traces, though less accurately.[4]

[4]It is incorrect to assume that Ack packets traverse the same data path in the opposite direction. However, in many cases the capacities of the bottlenecks on

Figure 6b shows an interarrival PDF for a flow going from a 10 Mb/s access link to a 100 Mb/s link, similar to Figure 4b. The EMGs of 0.12 ms continue along a long tail, indicating that the downstream high-capacity 100 Mb/s link is highly congested.

Like Figure 6b, Figure 6c demonstrates a flow going from a lower-capacity bottleneck to a higher-capacity bottleneck, except this time the upstream bottleneck (a T1) is highly congested. This generates primary EMGs of 8 ms, modulated by smaller EMGs of 0.12 ms corresponding to the 100 Mb/s link.

Figure 6d demonstrates a rare case where the PDF contains evidence of a congested link *upstream* of the minimum-capacity link. The flow traverses an upstream highly congested 100 Mb/s bottleneck and then a downstream 10 Mb/s bottleneck. The downstream bottleneck erases the first few spikes, piling up their probability at 1.2 ms, but the tail of 0.12 ms EMGs from the highly-congested 100 Mb/s link is long enough that a second spike remains.

Figure 6e shows an interesting structure which reveals three bottlenecks. The minimum-capacity bottleneck is a 380 Kb/s link, which is apparent from the envelope's peak. The envelope is modulated by EMGs of around 1.2 ms, revealing a 10 Mb/s link. If we then look closely around one of these modes, we see smaller modes equally spaced at intervals of 0.08 ms, revealing a downstream 155 Mb/s link.

As more bottlenecks leave their fingerprints on the flow's interarrivals, it becomes harder to disentangle their marks. It is relatively easy to identify two bottlenecks from an interarrival PDF, but we have never seen more than 3 bottlenecks. We do not know whether there were any cases in which our `ttcp` flow traversed 4 or more congested links, but we expect this to be unlikely. We cannot confidently tell the maximum number of detectable bottlenecks in a single PDF, but we believe that, without additional information, it will be difficult to identify more than 3 bottlenecks.

2.4 Ack Interarrivals

Thus far, we have created PDFs from data packet interarrivals, using traces collected downstream of any bottlenecks. This kind of analysis is useful when we have control of the receiver or some observation point close to the receiver. When the trace is taken at the sender-side, data packet interarrivals are not interesting because the packets are spaced by the sender's link; the *ack stream* holds whatever information can be recovered. When the observation point is in the middle of the network, both data and ack interarrivals should be studied to discover bottlenecks upstream and downstream of the observation point. In general, ack interarrival PDFs contain more information than data interarrival PDFs, but they also have a higher level of noise. The major differences between the two PDFs are:

• **Forward- and reverse-path bottlenecks.** If every data packet generated an ack, and ack spacing was undisturbed by the network, then sender-side ack interarrivals would exactly equal the receiver-side data packet interarrivals. Of course, the world is more complicated than this. Acks also traverse the network, where their interarrival times pick up a record of any bottlenecks on the reverse path. This record is superimposed on the record of forward-path bottlenecks generated by the data packets. We cannot tell whether a specific bottleneck is on the forward or reverse path unless we examine the data interarrivals as well.

the reverse path match the capacities of the bottlenecks on the forward path.

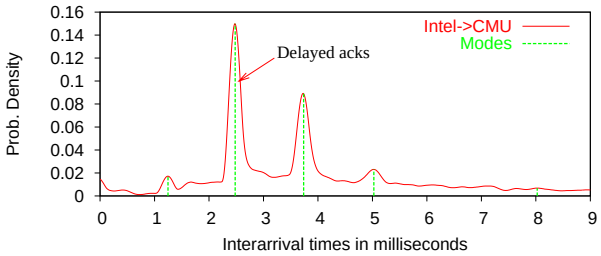


Figure 7—An example PDF showing delayed acks. The tall spike is caused by delayed acks. It happens at around 2.4 ms which is twice as long as the time taken to transmit 1500 bytes on a 10 Mb/s link. The modes are separated by 1.2 ms which is the transmission time of 1500 bytes on a 10 Mb/s link.

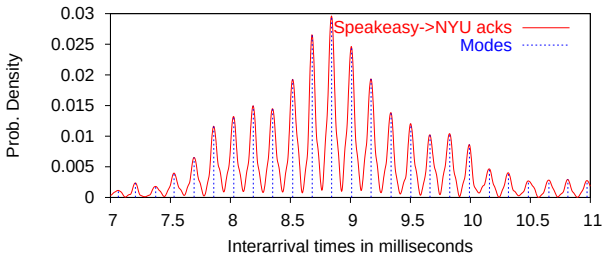


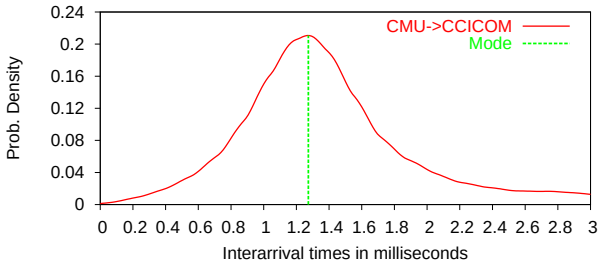
Figure 8—Ack interarrivals hold information about both the forward and reverse path bottlenecks. Data flows from nyu to speakeasy. The envelope peaks at 8ms, which is the transmission time on the speakeasy downlink. The modes are separated by 0.12ms, which is the transmission time on the speakeasy uplink.

To demonstrate this, Figure 8 shows an ack PDF with information about both forward- and reverse-path bottlenecks. The receiver is at the RON node "speakeasy", which has 1.5 Mb/s downstream capacity and 100 Mb/s upstream capacity. The PDF's envelope peaks at 8 ms, corresponding to the 1.5 Mb/s forward-path bottleneck. This envelope is modulated by 0.12 ms EMGs corresponding to the upstream 100 Mb/s link. If we plot the data-packet PDF for a flow that traverses the reverse path, we see only the 100 Mb/s link.

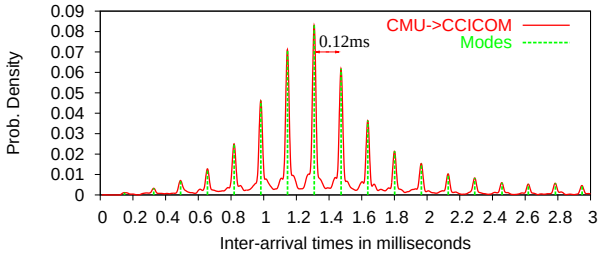
• **Noise.** Ack PDFs are significantly noisier than data-packet PDFs. Data packets are mostly 1500 bytes. When they arrive back-to-back they are spaced by the transmission time of a 1500 bytes packet on the bottleneck, and thus enforce the mode structure created by cross traffic bursts. Acks, on the other hand, are only 40 bytes long. Back-to-back acks do not enforce the mode structure created by cross traffic bursts.

• **Delayed acks.** The 1.2 ms EMGs in Figure 7, a sender-side ack interarrival PDF, clearly reveal that the flow has crossed a 10 Mb/s bottleneck; but the biggest spike is at 2.4 ms, twice the expected value. This is caused by delayed acks: the receiver generates most acks at half the rate of the minimum-capacity bottleneck. Thus, when working with ack interarrival, one should take into account that the first mode may be missing.

The examination of many ack PDFs shows EMG can be applied to ack interarrivals, but with lower accuracy than data packet interarrivals. Section 5.4 quantifies the difference.



(a) Main mode at around 1.2ms shows the 10Mb/s CMU link



(b) Gaps of 0.12ms show the 100Mb/s CCICOM link

Figure 9—The data from Figure 4b at two different resolutions.

3 MULTIQ: AUTOMATING EMG

The `multiQ` passive bottleneck detection tool automates the EMG capacity detection technique. It takes as input a `tcpdump` trace, and automatically discovers and estimates the capacity of the bottlenecks traversed by certain flows specified by the user.

Automating multiple bottleneck discovery is tricky because it requires interpreting the visual image of the interarrival PDF to extract the relevant information and ignore the noise. To do this, `multiQ` analyzes the interarrival PDF at a *progression of resolutions* corresponding to a known set of common link speeds. To demonstrate this, Figure 9 plots the CMU-to-CCICOM data from Figure 4b at two different resolutions. At the lower resolution, we see one large mode in the distribution, which corresponds to the upstream lower-capacity bottleneck. As we increase the resolution, the large mode becomes fractured into smaller spikes corresponding to the higher-capacity bottleneck. The envelope traced by the peaks of the smaller spikes follows the original broader mode.

The procedure works as follows. At each resolution, starting with the highest resolution, `multiQ` constructs a kernel density estimate of the PDF and scans it for statistically-significant modes.[5] The gaps between these modes are computed. Then, `multiQ` finds the probability distribution of the gaps themselves. A mode in the gap's PDF corresponds to a highly repeated gap length—the hallmark of a congested link. If `multiQ` finds a significantly dominant mode in the gap distribution at the current resolution, it decides that mode represents the transmission time of 1500 bytes on some bottleneck, and outputs that bottleneck's capacity. If there is no dominant gap at the current resolution, `multiQ` decreases the resolution by increasing the kernel width, and repeats the procedure. [6] Figure 10 shows this procedure in pseudocode.

1. Compute flow interarrivals from trace file
2. Set *scale* := 10 μs
3. While *scale* $<$ 10,000μs:
4. Compute kernel PDF estimate with width = *scale*
5. Find the modes
6. If there's only one mode, at M:
7. Output a capacity of (1500*8/M) Mb/s
8. Exit
9. Compute the mode gaps
10. Compute the PDF of the gaps
11. Set G := the tallest mode in the gap PDF
12. If the probability in $G > 0.5$:
13. Output a capacity of (1500*8/G) Mb/s
14. Increment *scale*

Figure 10—Pseudocode for `multiQ`.

A few details are worth discussing. First, since we are looking at the interarrival PDF at different resolutions, we need to use a kernel PDF estimator to detect the modes—the flat bins of a histogram would prevent precise mode estimation at low resolutions. Second, modes are identified as local maxima in the density estimate that have statistically significant dips.[7] Finally, when `multiQ` analyzes ack inter-arrival PDFs, it uses a slightly different procedure to deal with the first mode in the PDF: a large spike close to zero is a sign of compressed acks and should be ignored, whereas a spike located at twice as much as the repeated gap in the PDF is a sign of delayed acks and corresponds to the transmission time of 3000 bytes on the bottleneck link.

3.1 Miscellaneous

1. EMG estimation is more robust on data-packet traces than ack traces. When run on ack traces, the current version of `multiQ` does not try to discover bottlenecks whose capacity is higher than 155 Mb/s.

2. The EMG technique relies on the cross-traffic burst structure, which depends on the packet size distribution. If 1500 bytes stops being the dominant large-packet mode, this technique will fail. Fortunately, this distribution appears to be changing towards further emphasis of the 40-byte and 1500-byte modes; for instance, compare the 1998 and 2001 packet size distributions in Claffy's papers [13, 44].

3. `multiQ` estimates the capacity of bottlenecks upstream from the observation point. To estimate bottleneck capacities downstream of the observation point, `multiQ` needs access to ack traces.

4. If the traced flow traverses a high capacity bottleneck followed by a lower capacity bottleneck, it is unlikely there will be signs of the first bottleneck in the data interarrival PDF collected at the receiver-side (Figure 6d is a rare case). However,

[5] Kernel density estimation is a standard method for constructing an estimate of a PDF from measurements of the random variable. We use the quartic kernel density function [43].

[6] The width of the kernel function is similar to to the bin width of a histogram.

[7] A significant dip [43] is defined as one in which the dips on either side of a local maximum drop by more than the standard deviation of the kernel density estimate at the local maximum. The standard deviation is given by

$$StdDev(g(x)) = \sqrt{g(x) \times R(K)/nh}, \quad (2)$$

where $g(x)$ is the estimate at point x, $R(K)$ is the roughness of the kernel function, n is the number of points, and h is the kernel's width.

if multiQ has access to both sender- and receiver-side traces, then it is likely to detect both bottleneck capacities. Essentially, the ack stream would be traversing the lower capacity bottleneck first then the higher capacity one. If the bottlenecks on the forward path are also bottlenecks on the reverse path (usually true when they are access links), and have the same capacity in both direction (usually true except for DSL and cable modem links), then the capacities detected in the ack interarrival PDF match the capacities of the two bottlenecks on the forward path.

4 MYSTERY

The M&M toolkit can also correlate multiQ's bottleneck capacity information with other passively-measured path characteristics. To demonstrate this approach, we implemented the mystery tools, which measure loss event rates, packet loss rates, and RTT variability from TCP traces. These kinds of characteristics are well studied in the literature [37, 49, 6, 22, 10, 29, 4, 21]; mystery uses similar techniques to this prior work. We wrote our own tool because existing tools either had no source code available, or produced less fine-grained results than we wanted to use (a count of lost packets rather than the identities of particular lost packets, for example). For future work, we plan on integrating other tools into M&M as they become available.

The three tools that make up mystery are as follows:

- A *loss event detector* detects loss events by watching for retransmissions, a standard technique. A new loss event is detected every time mystery sees a reordered or retransmitted packet whose original transmission was not part of a previous loss event. True loss events can sometimes be differentiated from spurious retransmissions using timing information.
- A *lost packet detector* detects individual lost packets using algorithms somewhat like those of Allman et al. [6], but more dependent on timing information than duplicate ack counting.
- An *ack correspondence detector* measures semi-RTTs: the latencies between data packets and the acks sent in response, as seen by the trace collection point. Its algorithms follow those of Jaiswal et al. [21] and Aikat et al. [4].

The loss event detector works at either the sender or receiver side, and only requires access to the data packets. The lost packet detector and the ack correspondence detector are designed for the sender side, and require access to both data and acks.

mystery operates on tcpdump, NLANR, or other format traces containing one or more TCP flows. Its output is in XML format. More information about mystery can be found in [45]. Section 5.7 presents a validation.

5 VALIDATION

We evaluate the accuracy of multiQ using 10,000 experiments over 400 diverse Internet paths from the RON overlay network, and compare it both with known topology information and with two other capacity measurement tools, Pathrate and Nettimer. Our results show the following:

- When measuring minimum-capacity bottlenecks, multiQ is as accurate as Pathrate, an active measurement tool; 85% of its measurements are within 10% of the true value. Nettimer is equally accurate if operated with both sender and receiver traces, but its accuracy goes down to 74% with only receiver side traces, and becomes negligeable (about 10%) with only sender side traces.
- On sender side traces, which consist mainly of acks, 70% of multiQ's measurements are within 20% of their correct value.
- As for tight links (i.e. non-minimum capacity links), multiQ automatically detects 64% of them, misses 21%, (though a human could detect them visually on an interarrival PDF using our EMG technique), and mislabels 15%.
- The average error of both multiQ and Nettimer is highly independent of flow size for flows larger than 50 packets.
- We also validate mystery using 155 diverse paths from RON. When run at the sender side (the hard case), its error rate for lost packets is under 1% for more than 80% of the paths we tested, and under 10% for all paths. Ack correspondence is slightly less reliable.

5.1 Experimental Methodology

Ideally, we would like to have information about all the capacities and loss rates along a large number of heterogeneous paths that form a representative cross section of the network. This is inherently difficult on the Internet, of course, but we have tried to evaluate our tools on as representative a network as possible. We use the RON overlay network [2], whose 22 geographically-distributed nodes have a diverse set of access links, ranging from DSL to 100 Mb/s connections,[8] and ISPs on both the commercial Internet and Internet2. RON has 462 heterogeneous paths, 25% of which use Internet2. We therefore have good reason to believe that these paths' characteristics are representative of what we would encounter on the Internet.

We compare the capacity tools' estimates for each RON path against that path's "true" bottleneck capacity. A fair amount of legwork was required to determine these values. We contacted each node's hosting site and obtained a list of all their access links and the capacities of the local networks to which the nodes are connected. For multi-homed nodes, we learned the access capacities of each upstream ISP. RON nodes not on Internet2 have low-speed access links ranging from DSL to 10 Mb/s; paths terminating at one of these nodes are unlikely to encounter a lower-capacity link on the Internet backbone. For RON nodes in Internet2, we additionally obtained information about *all* Internet2 links on the relevant paths. On top of this, we used a wealth of information obtained from the RON overlay operator about path characteristics over the last 3 years.

To verify the consistency of these "true" capacities, we ran all three capacity measurement tools and a number of ttcp and UDP flows of varying rates on each path. If a path's results pointed out an inconsistency—for example, if ttcp or UDP obtained more bandwidth than the "true" capacity—then

[8] 9 nodes have 100 Mb/s uplinks, 6 have 10 Mb/s, 3 have T1, and 4 have DSL.

Source	Destination	Capacity estimate (Mb/s)		
		multiQ	Nettimer	Pathrate
jfk1-gblx	cybermesa	10.519	11.89	.998
nyu		10.563	10.514	.9985
cornell		8.134	8.1	.997
gr		8.134	8.139	.9985
cmu		8.13	8.121	.996

Table 3—Estimate differences between Pathrate and the other tools (see § 5.3).

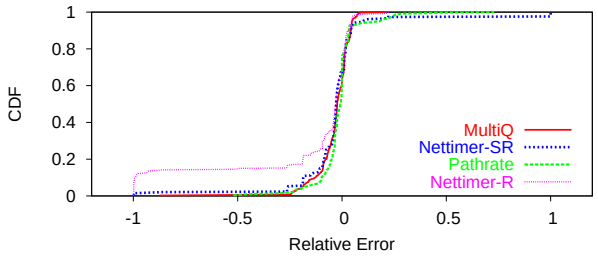


Figure 11—Comparison of the accuracy of MultiQ, Nettimer and Pathrate. Graphs show the CDF of the relative error. MultiQ and Nettimer-R require only receiver-side traces, while Nettimer-SR requires both receiver- and sender- side traces.

we eliminated the path from our experiments. Only 57 out of a total of 462 paths needed to be eliminated.

5.2 Timestamp Errors

An important source of possible error is the timestamps we get from `tcpdump`. Our tools work on single passive traces, so we don't need to worry about calibrating timestamps from multiple sites [38]; only errors in time *differences* are relevant. These errors may arise from fluctuations in the time it takes to go from an on-the-wire packet delivery to the network interrupt handler, which timestamps the packet on `tcpdump`'s behalf.

We analyzed a data set that contains both DAG hardware timestamps and `tcpdump` timestamps collected at RIPE [48]. Although `tcpdump` timestamps can differ from DAG hardware timestamps by 20 μs, the errors in the timestamps of consecutive packets are highly correlated. Hence, compared to interarrival times calculated from the DAG timestamps, the errors in interarrivals of successive packets computed from `tcpdump` timestamps are only a few μs. Such small errors should not affect our results.

5.3 Minimum Capacity Estimation

We now turn to an evaluation of `multiQ`'s minimum capacity estimation. We compute the relative error of `multiQ`'s estimates compared with the "true" minimum capacities, and compare that relative error with two other capacity measurement tools—Pathrate, which is active, and Nettimer, which is passive. We find that `multiQ` is very precise.

We tried to ensure that the three tools encountered the same path characteristics, such as loss rate and delay, by running the tools immediately after one another on each path. We first conduct a 2 minute run of `ttcp` and collect traces at both endpoints. These traces serve as data sets for `multiQ` and Nettimer. Immediately thereafter, we run Pathrate on the same path and compute its estimate; we use the average of Pathrate's high and low estimates. This procedure is repeated five times, and we report the average of those 5 trials. Finally, the same set of experiments is run both at day and night, to compensate for any traffic fluctuations due to the time of the day. In total, we performed more than 10000 experiments.

We plot the *relative error* ξ for each capacity estimate C_e, which is defined as

$$\xi = \frac{C_e - C_t}{C_t}, \tag{3}$$

where C_t is the path's "true" capacity.

Figure 11 shows the cumulative distribution function (CDF) of the relative errors of `multiQ`, Nettimer, and Pathrate estimates on RON's 405 paths. Nettimer has two lines: Nettimer-SR requires both sender- and receiver-side traces, while Nettimer-R requires only receiver-side traces. `multiQ` also requires only receiver-side traces. Ideally, the CDF should be a step function at "0", meaning that all experiments reported the "true" capacity. A negative relative error means that the tool has underestimated the capacity, whereas a positive relative error means that the tool has overestimated it.

Our results show that minimum capacity measurements are relatively accurate. On 85% of the paths, `multiQ`, Pathrate, and Nettimer-SR all report estimates within 10% of the "true" value. When Nettimer is given only the receiver-side trace, however, only 74% of its estimates are within 10% of the actual values. All three methods are biased towards underestimating the capacity.

Next, we look more closely at the errors exhibited by each tool. `multiQ` errors are caused mainly by over-smoothing in the iterative procedure for discovering mode gaps, which flattens the modes and prevents accurate computation of the gaps. Pathrate's logs indicate that its errors happen when the interarrival's distribution exhibits many modes. Though the correct bottleneck capacity is usually one of the modes discovered by Pathrate, the tool picks a different mode as the bottleneck capacity. When Nettimer made errors, we found that often the path has low RTT ($<$ 16 ms). The tool mistakes the RTT mode in the inter-arrival PDF for the transmission time over the bottleneck. The effect is most pronounced when Nettimer is operating with only traces at the receiver side; when it has both traces, we theorize that it can estimate the RTT and eliminate the corresponding mode.

Our experiments show that different tools can disagree on the capacity of a particular path, but can all be correct. We noticed that, on some paths, the Pathrate estimate differs substantially from the Nettimer and `multiQ` estimates. In particular, Pathrate repeatedly reports capacities of 1 Mb/s for paths going to cybermesa, while Nettimer and `multiQ` estimate them as 10 Mb/s (Table 3). Further investigation revealed that the differences are due to the flows being rate limited. The cybermesa access link capacity of 10 Mb/s is correctly estimated by Nettimer and `multiQ`. Pathrate's relatively long trains of back-to-back packets, however, trigger cybermesa's leaky bucket rate limit; they exceed the maximum burst size of the leaky bucket and become limited by the token rate, which is 1 Mb/s. TCP windows stay smaller than the bucket size, and so its packets are spaced by the actual link. This information has been confirmed by the

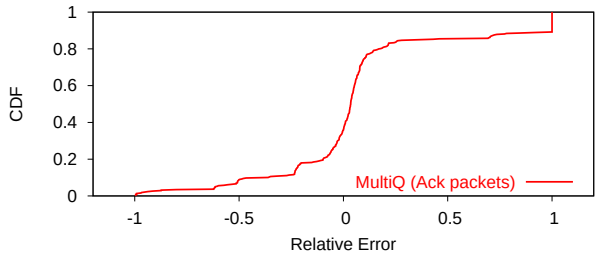


Figure 12—The accuracy of capacity estimates based on ack interarrivals.

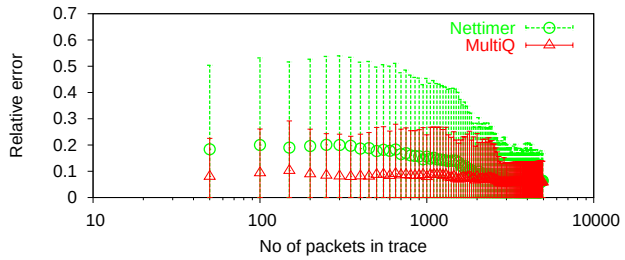


Figure 13—The relative error of MultiQ and Nettimer as a function of the traced flow size. Both average error and deviation are lower in the case of MultiQ.

site's owner.

5.4 Minimum Capacity Estimation Using Acks

Unlike existing tools, `multiQ` can obtain a reasonable capacity estimate exclusively using a *sender*-side trace, using the interarrival times of ack packets. Figure 12 shows the relative error of `multiQ`'s sender-side ack estimation, compared with its receiver-side data-packet estimation; the data comes from the experiments described in § 5.1. Since acks contain information about both forward and reverse links, we define the true capacity C_t for sender-side `multiQ` measurements as the minimum of the forward and reverse paths' capacities. Sender-side ack interarrivals produce lower-quality results than receiver-side data packet interarrivals, but still, 70% of the measurements are within 20% of the "true" value. Unlike receiver-side `multiQ`, the errors on sender-side `multiQ` tend towards overestimation.

5.5 Relative Error and Flow Size

We would expect capacity estimate error to be dependent on the amount of data available: more data should mean a better estimate. In this section, we quantify this effect.

Figure 13 plots the absolute value of the relative error of Nettimer-SR and `multiQ`'s estimates, as a function of the number of packets in the traced flow. We use the traces generated for § 5.1, truncated to various lengths; the relative errors are averaged over the whole set of RON paths. The bars show one standard deviation away from the average error. `multiQ`'s error is lower than Nettimer's for smaller numbers of packets. In fact, `multiQ`'s average error does not depend much on the number of packets, but the error variance decreases substantially as the number of traced packets increases. This means that there are particular flows in the data set that were hard to analyze and required a large number of packets for correct estimation. Also, the average error and error variance converge to nonzero values as the number of packets increases. This means that there are certain very noisy paths which neither `multiQ` nor Nettimer can correctly analyze, regardless of the number of traced packets.

Pathrate, on the other hand, is active. On our tests, it uses an average of 1317 probe packets, with a standard deviation of 1888 packets; but since it uses probes of varying sizes, a better metric is the amount of traffic it sends: 1.75 MB on average, with a standard deviation of 2.56 MB. The large standard deviation indicates that Pathrate uses far more traffic on paths that are hard to estimate.

Result	Fraction
Correct	64%
Incorrect	15%
Not estimated	21%

Table 4—`multiQ` tight link estimates

Avg. Relative Error	Std. Deviation in Error
0.156	0.077

Table 5—Average relative error and standard deviation in the correctly estimated tight links.

5.6 Tight Links

This section evaluates `multiQ`'s ability to discover non-minimum-capacity bottlenecks, or *tight links*; as discussed above, `multiQ` can report up to three bottleneck capacities per flow. Unfortunately, we usually cannot say with confidence what the true tight links along a path could be, and we can't correlate any results with other tools. To deal with this issue, we limit this test to Internet2 paths. Internet2 has a very low utilization (MRTG plots a maximum utilization $< 10\%$ [3]), so any observed queuing should be at the edges. Thus, for these paths we are reasonably confident that congestion happens at one or both access links, whose capacities we know. Also, because downstream narrow links tend to erase the effect of upstream bottlenecks (see § 2.2 and § 3.1) from data packet interarrivals, we limit this test to paths in which the downstream bottleneck capacity is larger than the upstream bottleneck capacity.

We run `ttcp` over each of these paths and log the packet arrival times at the receiver using `tcpdump`. The experiment is repeated multiple times during both peak and off-peak hours. We run `multiQ` on the resulting traces and record the various link capacities which are output. Each of these estimates could be a link on the path. We say that a tight link on a path is correctly estimated if one of the non-minimum-capacity estimates from `multiQ` is within 20% of the actual tight link capacity. All other estimates for that path are considered to be incorrect. If only the minimum capacity is found for a path, the answer for that path is logged as "not estimated". Tables 4 and 5 summarize the results: 64% of the experiments reported a tight link present on the path, 15% reported an invalid tight link (a bottleneck that differed from the correct value by more than 20%), and the remainder only reported the minimum bottleneck. The experiments that correctly found a tight link had an average relative error of 0.156.

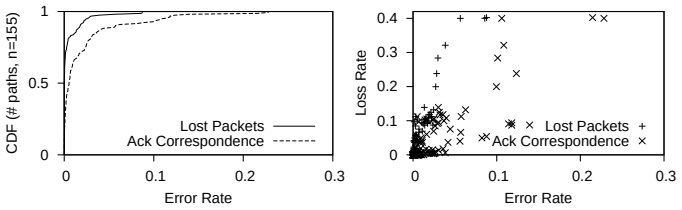


Figure 14—Error rates for `mystery`'s lost-packet and ack-correspondence detectors. On the left: error rate CDF; on the right: loss rate vs. error rate.

5.7 Lost Packets and Ack Correspondence

To validate `mystery`, we used 155 pairs of traces from the RON testbed, similar to those described in § 5.3. We run `mystery` on the sender-side trace (the hard case) and collect its main results—a set of lost data packets, and an ack correspondence mapping *AC*, which associates an ack with the data packet that triggers it. These results can contain four kinds of mistakes: "lost" packets that were actually received; "delivered" packets that were actually lost; incorrect ack correspondences; and missing ack correspondences. All of these results are easy to check given the receiver-side trace. If we assume that all drops happen inside the network, then packets are delivered iff they show up in the receiver-side trace;[9] and ack correspondences is easy to determine at the receiver side, where acks show up in a few milliseconds rather than an RTT.

Figure 14 shows the results. Each graph has error rate as its X axis, where the error rate is the number of mistakes divided by the total number of events (data or ack packets sent). The lost packet detector is quite reliable, achieving 99% accuracy on 80% of the 155 paths; the ack correspondence detector is also reliable, but less so. Both error rates rise with the loss rate (right-hand graph), but the lost packet detector still achieves 90% accuracy on all paths. We investigated particular traces with high error rates, and found that many of the errors are impossible to fix without DSACK information or other explicit feedback. In particular, reverse-path losses cause problems for the tool. When the network drops the single ack sent in response to a packet, `mystery` cannot hope to detect that the packet was delivered.

6 MEASUREMENT STUDIES

We now present several M&M-based measurement studies of Internet path characteristics, as examples of results that are relatively easy to find using our measurement methodology and toolkit. These studies are not intended to provide precise values but rather to show trends and loose estimates.

- **Evolution of bottleneck capacity.** We use `multiQ` to determine the path capacity distribution in two large sets of NLANR traces [33], taken in 2002 and 2004.

- **Statistical multiplexing.** We estimate the level of statistical multiplexing on the NLANR traces' bottleneck links using `multiQ` (to measure capacity) and `mystery` (to measure throughput and RTT).

[9] We do account for the very few packets that are dropped after the receiver trace point.

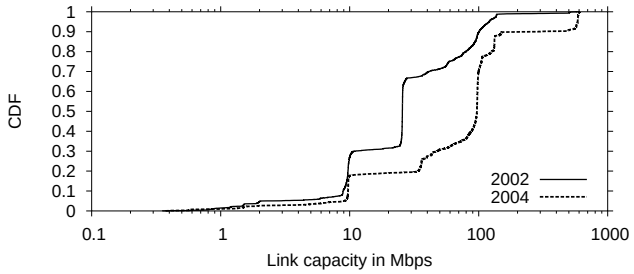


Figure 15—The empirical cumulative distribution of path minimum capacity in the 2002 and 2004 NLANR datasets. Graphs show a substantial increase in path capacity over a relatively short period.

- **Loss and bottleneck capacity.** `mystery` calculates the loss event rate for packets in the NLANR traces; we plot this against bottleneck capacity calculated by `multiQ`.

In addition to providing realistic parameter values for researchers to use in their simulations or models, our study reveals some interesting characteristics of Internet paths and their evolution. It shows that significant flows (see Table 1) have experienced a dramatic increase in bottleneck capacity between 2002 and 2004, but the higher capacity bottlenecks did not result in a larger per significant flow bandwidth share. Further, the drop rates experienced by significant flows on low capacity paths are considerably similar to the drop rates seen on high capacity paths.

The NLANR [33] traces, used in this study, contain more than 375 million packets in 258 traces, collected on one OC-48, five OC-12, and fifteen OC-3 links. There are two sets of traces, one collected in 2002 and one in 2004. The traces contained over 50,000 significant flows. Although this data is not representative of all Internet traffic—for example, it all comes from within the US—it is large and diverse, and was collected at major connection points to the backbone.

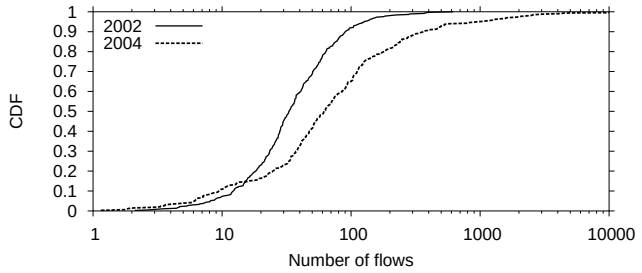
6.1 Bottleneck Capacity Distribution

We analyzed both the 2002 and 2004 NLANR trace sets using `multiQ`, extracting the bottleneck capacities experienced by every significant flow. Figure 15 shows the shift in path capacity (i.e., minimum capacity along a path) that occurred between the sets. In 2002, less than 20% of the significant flows were bottlenecked at a 100 Mb/s or higher capacity link. This number increased to 60% in 2004, showing a substantial and rapid growth in the capacity of bottleneck links. The highest bottleneck capacity that we identified in the 2002 data set is an OC-3 link. In contrast, the highest bottleneck capacity in the 2004 data set is an OC-12 link. Although this increase in bottleneck capacity is not uniformly distributed across all traces, it is impressive that the average bottleneck capacity has grown so much in a short period.

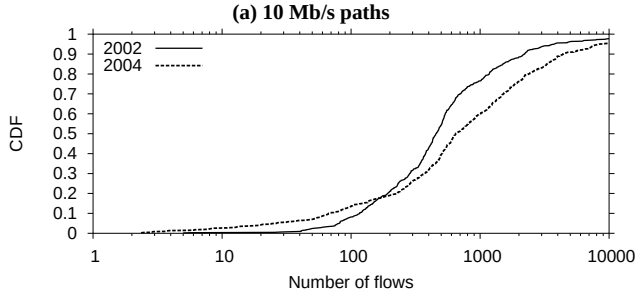
6.2 Statistical Multiplexing

Many published simulation scenarios assume low levels of statistical multiplexing on bottleneck links [17]. With `multiQ` and `mystery`, we can check this assumption.

We took the same NLANR traces from January, 2002 and 2004, and computed the level of statistical multiplexing for the



(a) 10 Mb/s paths



(b) 100 Mb/s paths

Figure 16—Distribution of statistical multiplexing on 10 and 100 Mb/s links in the 2002 and 2004 datasets. Graphs indicate that the average number of significant flows on 100 Mb/s bottlenecks is much larger than that on 10 Mb/s bottlenecks, causing the average per significant flow fair bandwidth share to be lower on 100Mb/s links than on 10 Mb/s links.

two prevalent bottlenecks, the 10 Mb/s and the 100 Mb/s links. `multiQ` tells us the minimum-capacity bottleneck link; because this link is likely congested, we assume, as a first approximation, that the bottleneck capacity is distributed fairly among flows on that link. We then estimate the number of flows on a bottleneck as the ratio of the bottleneck's capacity to the throughput of the flow. Because TCP flows share a link in inverse proportion to their respective RTTs, we first normalize each flow's throughput with respect to the average RTT across all flows traversing the same bottleneck capacity. We used `multiQ` to determine the bottleneck capacity of each flow and `mystery` to compute its RTT. We did not calculate statistical multiplexing for traces with incomplete TCP header information.

Figure 16 shows CDFs of the level of statistical multiplexing on these paths. For the 10 Mb/s links, the median degree in the 2002 traces was 30, whereas it is 60 in the 2004 traces, corresponding to a fair share changing from 330 to 160 Kb/s. For the 100 Mb/s links, the median degree in 2002 was 450, and in 2004 was 650. The fair share bandwidth for these paths was somewhat lower than the 10 Mb/s links, decreasing from 220 Kb/s to 150 Kb/s.[10] Contrary to conventional wisdom, the per significant flow fair bandwidth share does not increase with increased bottleneck link capacity.

6.3 Loss Rate and Bottleneck Bandwidth

Finally, Figure 17 shows a CCDF of loss event rates for groups of flows with different bottleneck capacities. We used `multiQ` to determine the bottleneck capacities of 15,000 significant flows from 2004 NLANR traces, and `mystery` to determine the loss event rate for each. We use TFRC's definition

[10] Some of these flows might be receive-window or application-limited, but the fair share bandwidth seems too low for this to be a major effect.

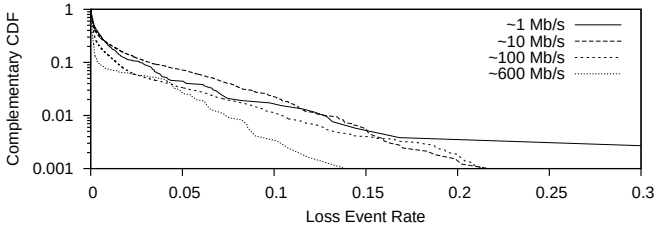


Figure 17—Complementary CDF of loss event rates for 13,627 significant flows from 2004 NLANR traces, divided into 4 bins by bottleneck capacity. Graphs show that TCP loss event rate does not decrease much with increased bottleneck capacity.

for loss event rate, namely the inverse of the average number of packets between loss events [16]; this is easily extracted from `mystery`'s output, a list of true loss events in the trace.

Loss events occur at all bottleneck capacities. Somewhat unexpectedly, the range of loss rates on 100 Mb/s-bottleneck flows is the same as for 10 Mb/s-bottleneck flows. Flows with 600 Mb/s bottleneck links still experience losses, but less so than flows with smaller bottlenecks.

7 Potential Applications of M&M

Building Models of the Internet: In Section 6, we have only scratched the surface of the Internet path properties M&M can measure. In the future, we would like to use M&M to address questions including: How many bottlenecks is a flow likely to encounter? When multiple queuing points exist, can one tell which among them is dropping the packets? Do published TCP equations accurately estimate the throughput obtained by real TCP flows? Additionally, by running `multiQ` on both sender and receiver traces of the same flow, we would like to investigate whether bottlenecks on the reverse path are the same as those on the forward path.

Overlay Management & Path Optimization: Overlays such as RON [2] or Planetlab [1] have multiple paths between any two nodes. Not all overlays provide routing functionality. Those that do, such as RON, use ping probes to discover loss rate and path characteristics. The resulting probe traffic make it hard to scale to a large number of nodes [8, 32]. The M&M tools can extract a path's characteristics from recent TCP flows traversing it, providing a cheap and scalable mechanism for route optimization in overlays.

M&M can help the operator do a better job even when the overlay does not offer routing functionality. Overlay nodes generally reside at distant hosting sites that do not fall under the control of the overlay operator. Analyzing the traffic using M&M can alert the operator to major changes in path characteristics in the overlay, such as changes in site access capacity, loss rate, or connectivity. Using active measurement to track these properties is usually considered unfriendly toward the hosting sites. Our work on M&M has helped in managing the RON network.

Network Tomography: One potential application is to use traceroute or equivalent [46] to discover the topology of a network, then use M&M to annotate the topology graph (or some segments of it) with link loss rates and capacities. Prior work has shown that TCP loss information, similar to that provided by

mystery, can be used to identify the lossy links inside the network from end-to-end traces [34]. It would be interesting to see whether capacity information can be used in a similar way. In particular, it may be possible to run multiQ on multiple paths, discover the capacity of the bottlenecks along each path, then correlate the common capacity values with shared path segments. Once a shared path segment is suspected to contain a bottleneck link with a certain capacity, a shared bottleneck detection technique [24, 26] is run to check whether the paths with the shared segment do actually share a bottleneck.

8 Related Work

Internet measurements can be divided into two classes, active and passive. Active measurements send probe traffic along a studied path to induce a network reaction that reflects the state of the path, where passive measurements extract information from packet traces or data flows that have already traversed the studied path. Active measurements are usually more powerful because the investigator can control the timing and the sending rate of the probes, but the extra load generated by probes can be undesirable, and active measurements cannot be executed on paths not controlled or accessible to the measurement tool.

Our work on multiQ is particularly related to prior work on capacity measurements and tight link discovery. Capacity measurement is already a mature field with many relatively accurate tools. Currently, Nettimer [28] is the main passive tool for discovering path capacity. Our work builds on the insight gained from Nettimer, but achieves higher accuracy and can discover multiple bottleneck capacities. Further, our tool can discover bottleneck capacities from sender side traces or receiver side traces, whereas Nettimer requires the receiver side trace to achieve any accuracy. Jiang and Dovrolis [23] describe a passive method of capacity estimation based on histogram modes.

There are many active tools for measuring path capacity. Some of these tools try to find the capacities of all links along the path [36, 30]. Others, such as Pathrate, focus on the minimum capacity of a path [14]. The accuracy and the amount of generated traffic vary considerably from one tool to another. Being passive, our tool differs from active tools in its methodology and characteristics.

Prior work that detects *tight* links—non-minimum-capacity bottlenecks—has all been active to our knowledge [5, 30]. There are also tools for discovering the available bandwidth along a path [20, 31, 47, 41, 42, 19], which all actively probe the network.

Shifting focus from tools to the underlying techniques, much prior work used packet inter-arrival times to estimate link capacities. Keshav proposed the concept of "Packet Pair" for use with Fair Queuing [25]. This refers to sending two back-to-back packets and computing the bottleneck capacity as the packet size divided by the pair dispersion at the receiver side. Packet pair is at the heart of many capacity and available bandwidth estimation methods, including ours.

Cross traffic can cause errors in packet pair-based capacity estimates. In particular, Paxson observed that the distribution of packet-pair capacity measurements is multi-modal [40], and Dovrolis et al [15] show that the true capacity is a local mode of the distribution, often different from its global mode. Many researchers have noted that some of the modes in the inter-arrival distribution may be created by secondary bottlenecks or post-narrow links [15, 27, 35]. Various mechanisms to filter out the cross traffic effects were proposed, such as using the minimum dispersion in a bunch of packet pairs, using the global mode in the dispersion distribution [28, 23], and using variable size packet pairs [15]. This paper complements the above prior work, but takes the opposite tactic—rather than filtering out the impact of cross traffic, we leverage the useful structure in the packet dispersion distribution created by cross-traffic to detect the capacities of multiple bottlenecks. Using all the modes in this distribution allows us to infer novel information from inter-arrivals PDFs missed by prior techniques. Specifically, we can infer the capacity and relative location of *multiple* bottlenecks along a path and also reverse path link capacities from TCP acks.

mystery's tools for measuring TCP losses, loss events, and semi-RTTs are based on prior work, in particular Allman et al. for losses [6] and Jaiswal et al. [21] and Aikat et al. [4] for semi-RTTs. The literature on passive TCP measurement is extensive ([37, 22, 10, 9, 7, 29, 49] and many more), and we hope to integrate several other tools into M&M for future work. The T-RAT tool [49] would be a natural fit for M&M; it analyzes passive traces to classify TCP flows based on the main factors limiting their rates.

Finally, our work greatly benefits from CAIDA and NLANR's efforts to collect traces and analyze Internet traffic [11, 33].

9 Conclusions

We have presented the M&M passive toolkit for large-scale measurement and analysis of Internet path properties. M&M is centered around a multi-capacity bottleneck estimator, multiQ, which introduces the novel insight that equally-spaced mode gaps (EMGs) in the packet interrarrival PDF correspond to the transmission time of 1500-byte packets on some congested link along the path. Uniquely to passive measurement tools, multiQ can discover the capacity of up to three bottlenecks and their relative location from a tcpdump trace of a single flow. M&M also contains, mystery, a TCP analyzer whose results can be combined with multiQ to correlate path properties with TCP performance. We have calibrated these tools using extensive tests on 400 heterogeneous Internet paths.

We have used M&M to analyze a huge trace library collected by NLANR on backbone links. Our analysis shows that Internet TCP flows (or more accurately those traced by NLANR) have experienced a dramatic increase in bottleneck capacity between 2002 and 2004, but the higher capacity bottlenecks did not result in a larger per flow bandwidth share. Further, the drop rates experienced by these flows on low capacity paths are considerably similar to the drop rates seen on high capacity paths.

References

[1] Planetlab. www.planet-lab.org.

[2] Reilient Overlay Network. nms.lcs.mit.edu/ron.

[3] Abilene. http://monon.uits.iupui.edu/.

[4] J. Aikat, J. Kaur, F. D. Smith, and K. Jeffay. Variability in TCP

round-trip times. In *Proc. IMC*, Oct. 2003.

[5] A. Akella, S. Seshan, and A. Shaikh. An Empirical Evaluation of Wide-Area Internet Bottlenecks. In *Proc. IMC*, Oct. 2003.

[6] M. Allman, W. Eddy, and S. Ostermann. Estimating loss rates with TCP. *ACM Performance Evaluation Review*, Dec. 2003.

[7] M. Allman and V. Paxson. On Estimating End-to-End Network Path Properties. In *ACM SIGCOMM*, 1999.

[8] D. Andersen, H. Balakrishnan, M. F. Kaashoek, and R. Morris. Resilient Overlay Networks. In *Proc. of the 18th ACM SOSP'01*, Oct. 2001.

[9] H. Balakrishnan, V. Padmanabhan, and R. Katz. TCP Behavior of a Busy Internet Server: Analysis and Improvements. In *INFOCOM (1)*, 1998.

[10] P. Barford and M. Crovella. Critical path analysis of TCP transactions. In *SIGCOMM*, 2000.

[11] Cooperative Association for Internet Data Analysis (CAIDA). http://www.caida.org/.

[12] R. Carter and M. Crovella. Measuring Bottleneck link Speed in Packet-Switched Network. Technical Report TR-96-006, Boston University, Mar. 1996.

[13] K. Claffy, G. Miller, and K. Thompson. The Nature of the Beast: Recent Traffic Measurements from an Internet Backbone, 1998. http://www.caida.org/outreach/resources/learn/packetsizes/.

[14] C. Dovrolis, P. Ramanathan, and D. Moore. Packet Dispersion Techniques and Capacity Estimation. *submitted to IEEE/ACM Transactions in Neworking*.

[15] C. Dovrolis, P. Ramanathan, and D. Moore. What do Packet Dispersion Techniques Measure? In *IEEE INFOCOM '01*, 2001.

[16] S. Floyd, M. Handley, J. Padhye, and J. Widmer. Equation-Based Congestion Control for Unicast Applications. In *Proc. SIGCOMM*, Aug. 2000.

[17] S. Floyd and E. Kohler. Internet Research Needs Better Models. In *HotNets-I*, Oct. 2002.

[18] C. Fraleigh. Packet Level Traffic Measurements from a Tier-I IP Backbone. Technical Report TR01-ATL-110101, Sprint, 2001.

[19] N. Hu and P. Steenkiste. Evaluation and Characterization of Available Bandwidth Techniques. *IEEE JSAC Special Issue in Internet and WWW Measurement, Mapping, and Modeling*, 2003.

[20] M. Jain and C. Dovrolis. Pathload: A Measurement Tool for End-to-End Available Bandwidth. In *Passive and Active Measurements*, March 2002.

[21] S. Jaiswal, G. Iannaccone, C. Diot, J. Kurose, and D. Towsley. Inferring TCP connection characteristics through passive measurements. In *Proc. IEEE INFOCOM*, Mar. 2004.

[22] H. Jiang and C. Dovrolis. Passive Estimation of TCP Round-Trip Times, 2002. To appear in ACM CCR.

[23] H. Jiang and C. Dovrolis. Source-Level IP Packet Bursts: Causes and Effects. In *Proc. IMC*, Oct. 2003.

[24] D. Katabi, I. Bazzi, and X. Yang. A passive approach for detecting shared bottlenecks. In *The 11th IEEE International Conference on Computer Communications and Networks (ICCCN '01)*, Scottsdale, Arizona, Oct. 2001.

[25] S. Keshav. A Control-Theoretic Approach to Flow Control. In *ACM SIGCOMM '88*, September 1991.

[26] D. R. J. Kurose and D. Towsley. Detecting shared congestion of flows via end-to-end measurements. In *The Proc. of ACM SIGMETRICS '00*, June 2000.

[27] K. Lai and M. Baker. Measuring Bandwidth. In *INFOCOM*, 1999.

[28] K. Lai and M. Baker. Nettimer: A Tool for Measuring Bottleneck Link Bandwidth. In *Proc. USENIX*, 2001.

[29] G. Lu and X. Li. On the Correspondency between TCP Acknowledgement Packet and Data Packet. In *Proc. IMC*, Oct. 2003.

[30] R. Mahajan, N. Spring, D. Wetherall, and T. Anderson. User Level Internet Path Diagnosis. In *Proc. ACM SOSP*, Oct. 2003.

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

[32] A. Nakao, L. Peterson, and A. Bavier. A Routing Underlay for Overlay Networks. In *ACM Sigcomm*, 2003.

[33] National Laboratory for Applied Network Research. http://pma.nlanr.net/.

[34] V. Padmanabhan, L. Qiu, and H. Wang. Server-based Inference of Internet Link Lossiness. In *IEEE Infocom '03*, 2003.

[35] A. Pasztor and D. Veitch. The Packet Size Dependence of Packet Pair Methods. In *Proc. of 10th IWQoS*, 2003.

[36] pathchar. ftp://ee.lbl.gov/pathchar.tar.Z.

[37] V. Paxson. Automated Packet Trace Analysis of TCP Implementations. In *ACM SIGCOMM*, pages 167–179, 1997.

[38] V. Paxson. On Calibrating Measurements of Packet Transit Times. In *Proc. SIGMETRICS 1998*, June 1998.

[39] V. Paxson. End-to-End Internet Packet Dynamics. *IEEE/ACM Transactions on Networking*, June 1999.

[40] V. E. Paxson. *Measurements and Analysis of End-to-End Internet Dynamics*. PhD thesis, Berkeley, 1997.

[41] V. J. Ribeiro, M. Coates, R. H. Riedi, S. Sarvotham, and R. G. Baraniuk. Multifractal Cross Traffic Estimation. In *Proc. of ITC Specialist Seminar on IP Traffic Measurement*, September 2000.

[42] V. J. Ribeiro, R. H. Riedi, R. G. Baraniuk, J. Navratil, and L. Cottrell. pathChirp: Efficient Available Bandwidth Estimation for Network Paths. In *Passive and Active Measurement Workshop*, 2003.

[43] D. Scott. *Multivariate Density Estimation*. John Wiley, 1992.

[44] C. Shannon, D. Moore, and K. Claffy. Beyond Folklore: Observations on Fragmented Traffic. In *IEEE/ACM Transactions on Networking*, Dec. 2002.

[45] S.Katti, D. Katabi, C. Blake, E. Kohler, and J. Strauss. M&M: A passive toolkit for measuring, tracking and correlating path characteristics. In *MIT CSAIL Technical Report 945*, 2001.

[46] N. Spring, R. Mahajan, and D. Wetherall. Measuring ISP topologies with Rocketfuel. In *Proc. SIGCOMM*, Aug. 2002.

[47] J. Strauss, D. Katabi, and F. Kaashoek. A Measurement Study of Available Bandwidth Estimation Tools. In *Proc. IMC*, Oct. 2003.

[48] H. Uijiterwall and M. Santcroos. Bandwidth Estimations for Test Traffic Measurement Project, Dec. 2003.

[49] Y. Zhang, L. Breslau, V. Paxson, and S. Shenker. On the Characteristics and Origins of Internet Flow Rates. In *ACM SIGCOMM 2002*, 2002.