Estimating Bulk Transfer Capacity

Jacob Strauss and M. Frans Kaashoek
MIT Laboratory for Computer Science
{jastr, kaashoek}@pdos.lcs.mit.edu

***Abstract*—**

Many applications wish to predict what bandwidth TCP can achieve on a path between Internet hosts without performing extensive measurements. More precisely, they would like to know what the bulk transfer capacity (BTC) of a given path is. This paper investigates the differences between available bandwidth (the unused capacity along a path) and BTC over a wide range of network conditions. We use Pathload [1] to obtain available bandwidth estimates over all paths between machines in the RON [2] testbed. The median BTC rate is under 40% of the available bandwidth on these paths. The BTC rate was less than one fifth the available bandwidth on a full third of paths. This paper also introduces a method, *squeezed pairs*, to characterize queuing delays along paths. We show that squeezed pairs allows us to make BTC estimates from available bandwidth estimates with accuracy comparable to BTC estimates from short TCP connections.

I. Introduction

Measurements of end-to-end bulk transfer capacity, or BTC for short, between Internet hosts are useful for overlay networks, congestion control, streaming media, and network monitoring applications. For example, resilient overlay networks (RONs) can use BTC to select the Internet path that will achieve the highest TCP throughput [2]. All existing methods used to measure BTC are intrusive, sending large amounts of probe traffic which interferes with other flows using the same path. Non-intrusive measurement methods do exist for other end-to-end path characteristics such as capacity and available bandwidth, but not for BTC.

A. Definitions

We will use the terms: bulk transfer capacity, available bandwidth, path capacity, link capacity, narrow link, and tight link, throughout this paper. Not all related work assigns the same meaning to these terms, especially available bandwidth and bulk transfer capacity. To avoid confusion, we define these terms here before continuing. The definitions here are the same as in much of the related work [3], [4], [5], [6], [1].

Consider the path between two hosts on the Internet, from A to B. This path is made up of n links, each of which has a link capacity, c_1 through c_n. Each link capacity is the fastest rate at which packets can be forwarded over that link. The *path capacity* from A to B is $\min_{i=1...n}(c_i)$. We call the link with smallest capacity c_i the narrow link from A to B. There may be more than one narrow link along a path, in which case the capacity of each must be the same. We assume that link capacities change rarely – They should only change with either a route change, or a physical change to the underlying links. In our tests, the narrow link is almost always no more than a few hops away from the endpoint. As such, we assume that c_i is constant for the duration of one experiment, which may be a few minutes long.

Each link also has a current utilization u_i. Utilization is the ratio of that links' capacity which is used over some time interval. $1 - u_i$ is the idle fraction of that link for the same interval. The unused capacity on each link is $c_i(1 - u_i)$.

The *available bandwidth* from A to B is $\min_{i=1...n} c_i(1 - u_i)$. The tight link is the link from A to B with smallest unused capacity. As with narrow links, there may be more than one tight link if queuing due to competing traffic or dropped packets occurs at more than one link.

The *bulk transfer capacity*, as described by Mathis and Allman [6], is the fastest a protocol that implements congestion control can forward packets from A to B. TCP is one example of such a protocol, which we use to measure BTC. BTC depends only on network conditions. For example, buffer space on routers between A and B, queuing policies, and cross traffic on all hops will affect BTC. Buffer space available on A or B does not. Like available bandwidth, BTC is time dependent. Unless stated otherwise, we refer to averages over several seconds to a minute for BTC and available bandwidth.

All of the quantities we have defined may be asymmetric. The path capacity or BTC from A to B may be different from B to A.

B. Bulk Transfer Capacity Properties

In many applications, BTC is the most desired metric to evaluate an Internet path. Capacity is certainly interesting if ones goal is topology mapping. However, for any application that is considering using the path and wishes an estimate of what the network will support, or picking between paths, BTC is more useful than capacity estimates or available bandwidth. However, we have effective non-intrusive methods to measure capacity and available bandwidth, and none for BTC.

Why is BTC hard to measure? Unlike capacity and available bandwidth, we lack a simple way to express BTC in terms of network observables. BTC is instead described in terms of TCP's congestion control algorithms [7][8].

TCP throughput is affected by many different factors, which include path latency and capacity of every link, nature of competing traffic at each link, router queuing policy and buffer sizes, random losses, link-level performance, reverse path conditions, data corruption, among other factors. The Amherst model [9], expresses TCP throughput in terms of observed loss rate along with measurable parameters such as round trip time, and TCP implementation. However, the loss rate cannot be extracted from other information easily. Goyal et al. [10] tried to refine this model to use loss rates obtained from routers along the path in question. However, their estimates hinged on correctly predicting loss rates based on router drop rates.

C. Contributions

This paper uses a recent tool, Pathload, for measuring available bandwidth to examine differences between bulk transfer capacity and available bandwidth. We show that BTC and available bandwidth often differ on the RON testbed, which covers a wide range of conditions. We know of no prior comparison between BTC and available bandwidth under real network conditions.

This paper also introduces a method, squeezed pairs, to characterize queuing delays on paths. We show that squeezed pairs allows us to make BTC estimates from available bandwidth estimates with accuracy comparable to BTC estimates from short TCP connections.

Finally, much related work on capacity and load estimates uses simulation and measurements over a few Internet paths. Only a few such methods have been tested over a wide variety of Internet paths. This paper offers some practical experience of using a number of tools under real network conditions.

D. Overview

The remainder of this paper is outlined as follows. In section II we discuss related work. In section III, we describe the measurements that we examine, and introduce squeezed pairs. Section IV discusses details of our measurement experiment, and section V presents measurement results. We conclude with a discussion of our results.

II. Related Work

Much work has been done in the area of capacity estimation. Most tools are variations on packet pairs. Nettimer [11], Packet Bunch Modes [12], bprobe [13], and Pathrate [3] all discuss methods for filtering packet pair measurements to determine path capacity. Pathchar [14] determines hop-by-hop capacity.

Tools to measure BTC include Treno [15] and cap [5]. Both tools aim to abstract away the details of TCP implementations, but they still require long intrusive measurement periods in order to obtain accurate measurements. Allman found that BTC values reported by cap generally agreed with the BSD TCP implementation to within 10% [5]. We use TCP directly in our measurements, and assume that rates we observed are close to the BTC.

Pathload [1], [4], which we use in our measurements, uses Self Loading Periodic Streams to measure available bandwidth. Self Loading Streams measure available bandwidth by sending short packet bursts and observing increasing queueing delays over the course of the burst. Sending bursts at rates below the available bandwidth does not cause increasing trend through the course of the burst. By starting with an initial estimate of the Aymptotic Dispersion Rate [3], an iterative search proceeds between rates that show increasing trends and rates which do not. Each rate tested requires a large amount of data, on the order of a hundred kilobytes, in 12 bursts each one hundred packets long. The authors argue that the short duration of each burst does not interfere with competing traffic. Pathload's authors verified correct operation through simulation and averages reported from in-path routers using Multi Router Traffic Grapher (MRTG)[16]. They did not address the issue of whether, or under what conditions, pathload could be used to predict TCP throughput.

An earlier attempt to use Self Loading Streams was done in [17]. Pathload is more advanced, and so we incorporated many details from Pathload's early descriptions. In a few cases, however, our implementation gives better estimates than Pathload, and in those few cases we report the numbers of our implementation as Pathload's.

Zhang et al. [18] examined stationarity of TCP measurements over many internet paths. They found that in

many cases, TCP speeds varied by less than a factor of three over the course of an hour or more.

III. Approach

Based on our experiences with self loading streams tests on the RON network, and attempts at other simple attempts to measure BTC, we believe that no single BTC measurement method exists that will work in all conditions. Instead, we believe that a better approach is to find simple methods that work in some conditions, along with tests for the appropriate conditions.

In this paper, we evaluate whether Pathload can be used as a simple method to meausure BTC in some conditions. We also present a condition test we call squeezed pairs to determine appropriate conditions for Pathload use.

A. Squeezed Pairs

Squeezed pairs represents an attempt to directly examine the state of queues on the routers between sender and receiver. Squeezed pairs is an approach derived from Bolot's observations on packet delay [19].

The squeezed pair test sends a squence of packet pairs from sender to receiver, in a manner similar to packet pair. Instead of sending the two probes back-to-back, they are spaced by a small, but deliberate, spacing. We perform the test for two spacings: one that is equal to half of the transmission time of 500 bytes and one that is equal to half of the transmission time of 1500 bytes, the two prevalent packet sizes used most often in the current internet [20]. We compute the spacings based on the path capacity.

Each pair encounters one of three effects before arriving at during their travel. The pair may arrive with a spacing unchanged from the spacing at the sender, the packets may be squeezed together at the receiver, or they may be spread further apart.

The amount by which the pairs are squeezed or spread shows relations between the queues the first and second packet experienced. If a packet from cross traffic queues between the two probes, and the probes are undisturbed on later portions of the path, the receiving gap must be at least as large as the transmission time of the cross traffic packet. Because the test spaces probe packets at an interval smaller than the transmission times of 500 and 1500 bytes, it is likely that a cross-traffic packet cannot be queued between one of the probe packets without disrupting the initial spacing.

IV. Methodology

We performed our measurements on the RON[2] testbed. The RON testbed currently consists of hosts installed at business, residential, and educational installations. While running our tests, we used fifteen different sites. Three are in Europe, one in Korea, and the remainder are in the US. Table I lists properties of each host.

TABLE I
RON sites and locations. Bandwidths are in Mbps

Name	Description	Connection type	Speed
MS	Residence, CA	DSL	0.384
Mazu	.COM in MA	T1	1.544
M1MA	Residence, MA	Cable Modem	10
Aros	ISP in UT	Fractional T3	10
CCI	.COM in UT	Ethernet	100
PDI	.COM in CA	Ethernet	3..30
CMU	Pittsburg, PA	Ethernet	100
Cornell	Ithaca, NY	Ethernet	10
MIT	Cambridge, MA	Ethernet	100
NYU	Manhattan, NY	Ethernet	100
Utah	U. of Utah,	Ethernet	100
NL	Vrije U,Holland	Ethernet	100
Lulea	Sweden	Ethernet	10
Korea	Korea	Ethernet	100
Gr	Greece	Ethernet	100

We wished to test possible methods in as wide a range of conditions as possible. To achieve this goal, we chose to run tests a few times over as many paths as possible, rather than repeating the same tests on a smaller number of paths. This approach is a reasonable one, as we want to consider as many unique network features as possible. Features present in the RON testbed include traffic shapers, trancontinental links, a variable capacity path, and machines at business, residential, and university locations.

We have combined the packet-sending and receiving portions of our tests into a single sender program, and a single receiver program. Each method we wish to test has a corresponding phase in the sender and receiver, which runs in turn. After all phases are complete, the sender begins a pair of bulk TCP transfers to the receiver, to establish the bulk transfer capacity of that path to evaluate the other methods. The TCP transfer is done last since it is potentially intrusive to other flows sharing portions of the path between the sender and receiver.

Each test from source to destination consists of the following phases, each separated by a pause of a few seconds:

1) Tcpdump starts at both source and destination
2) Source opens a TCP control connection to the receiver
3) Source sends 30 1400 byte packets back to back, pauses one round trip via a null command and response over the control connection, and sends an-

other set of 30 packets.

4) The destination calculates the average arrival rate of the two bunches and informs the sender, which uses the rate as an under-estimate of the path capacity.
5) Squeezed Pair Phase: Source sends up to 4000 pairs of packets, with the spacing within each packet set to alternate between half the width of a 500 byte and half the width of a 1500 byte packet based on the path capacity estimate. The pairs are sent with exponentially distributed spacings, with a mean space of the larger of ten times the largest intra-pair gap, or 2 milliseconds. Probe packets are 40 bytes long, including IP and UDP headers.
6) Self Loading Periodic Streams Phase: The source chooses 4 sending rates as fractions of the capacity estimate. Packet streams are sent at these four sending rates, with parameters the same as in Pathload. However, the a stream of each rate is done in turn rather than doing all at a single rate before changing.
7) TCP test: The source begins a bulk TCP transfer of 2 MBytes of data to the receiver. After the first completes, a second transfer is run. Sender and receiver windows are set to 1 Mbyte to ensure that the connection will not be bound by sender and receiver window sizes. We had used 200Kbyte windows initially, and noticed no changes with the larger window sizes.

After the above tests end, we start a Pathload run from sender to receiver, and then repeat the whole process with the sender and receiver swapped. Each test takes at most a few minutes on slower links, so we have data for forward and reverse paths separated by a few minutes. We then move on to the next pair of hosts, with the order selected randomly.

We have separated the execution and analysis of each method by collecting packet timings at both the sender and receiver by running tcpdump. This separation gives us several advantages over a combined approach, the greatest of which is flexibility. Much of the sender and receiver code is time-critical, and kept as sparse as possible. The analysis, however, is not time critical, and may change often. Since we are operating on packet traces, we can change the analysis code without having to re-run the overall test when the packets sent do not change.

The RON machines are primarily Pentium III Celerons running at 700 Mhz. All run FreeBSD-4.5. Most of our tests involve sending packets at specified time, or with a small specified spacing between them, so we take an optimistic approach to scheduling. By scheduling packet send times in advance, and then waiting until the appropriate time, the sending process will either be running, or swapped out at the desired time. Then we either send our packet on time, notice that we cannot, and thus either drop or retry the current test, or fall in the rare case that we are interrupted between checking the current time and calling send(). In this way, no more than one packet out of a single interval will be late. Using median-based filtering at the receiver will disregard these errant packets.

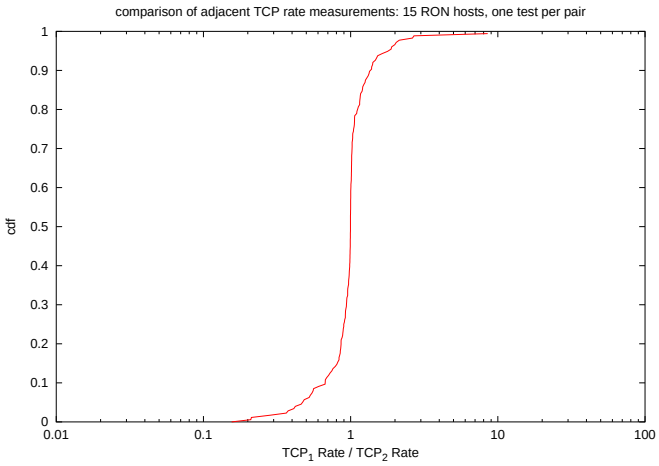


Fig. 1. Ratio of throughput acheived by first tcp transfer to second tcp transfer

Out of 210 total paths, 13 were discarded due to incomplete tests. In these cases, either the tcp or pathload tests aborted due to losses, or the end hosts could not be reached from the experiment coordinating host. Approximately five of these 13 incomplete tests were due to operator error.

V. Results

First we examine each individual method, and show that none provides an adequate BTC estimate when used in isloation. For each method, we evaluate it in two manners. The first is to determine, for each path, the ratio of that method's measurements to the actual achieved BTC, as measures by a bulk TCP transfer. This evaluation method is appropriate for applications where the application wishes to maintain a given level of service.

The second evaluation method is to pick pairs of paths, and for each possible pair, compare the two paths to predict which is faster. If the prediction matches the actual comparison, this test is counted as a correct prediction. If they differ, then the test is counted as incorrect. We divide the totals to get the frequency of correct decisions.

A. Individual Methods

1) TCP predicting TCP: Our test includes two adjacent TCP measurements. Figure 1 shows the ratio of

acheived throughput in the first test to the second test. All of our graphs are CDFs in this form. They each plot the ratio of one measured or estimated transfer rate to a measured baseline rate. If a prediction is a perfect match to the actual rate, the plot would appear as a vertical line at $x = 1$. When the predicted values are too low, the plotted ratio will be to the left of $x = 1$, and when the prediction is too high, the plotted value will fall to the right.

Each TCP test consisted of a 2 MByte transfer. Timings were taken at the sender from the time the first SYN packet was sent, to the time the last data packet was sent. The two TCP tests were separated by a pause of 5 seconds. There is no significant bias towards either the first or second transfer rate. From figure 1, we can make a few statements about the predictive quality of adjacent TCP connections. First, the distribution shows a long tail. Given one rate, there is a nonzero, though small, possibility that the other rate will vary by as much as a factor of 10. Given one rate and no other information, there is a 70 % chance that the other rate will fall within 20% of the first rate. Over 90% of rates fall within 50% of their adjacent rate.

We expect these results to be similar to cap and TReno, as all of these tools implement a TCP-friendly congestion-avoidance analysis.

These esimates provide a baseline for how to evaluate other rate estimates. Ideally, the two should rates should only differ if the network conditions change between the two tests. However, if a single event such as a loss of number of burst packets causes the sender to wait for a long timeout, then the two rates may appear significantly different without a clear cause.

This effect should be less of a concern on slower paths, as a single timeout would have a smaller effect on overall average throughput. An analog to figure 1, but instead limited to slow paths, did not change the shape noticeably. Using the first TCP rate to pick which will show the faster rate on the second transfer was correct in 93% of path pairs.

2) Self Loading Periodic Streams: Our tests included two implementations of Self Loading Periodic Streams: Pathload, and one we implemented ourself. Both tools output a range of values, between which the "true" available bandwidth fell. We found that in all but a handful of cases, the range estimated by our tool and Pathload were consistent. Pathload yielded tighter ranges except when the available bandwidth was measured below 1.2 Mbps. In that case, Pathload would report a lower range estimate of 0.0 Mbps, wheras our tool could sometimes report a lower range estimate greater than zero.

On many of the slowest paths, Pathload's upper range estimate was only slightly below the capacity of that path.

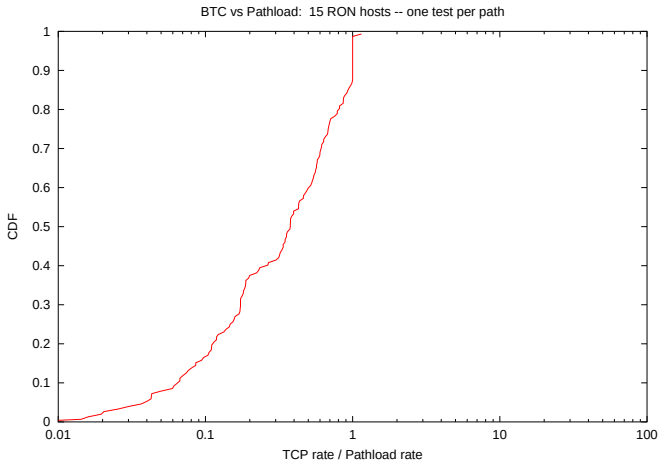


Fig. 2. Available Bandwidth as measured by Pathload, vs. BTC as measured by TCP rate. The Pathload rate is a range – the closer of the range endpoints for each point is used. If the TCP rate falls within the range, a ratio of 1 is plotted.

As we wished to evaluate how much Pathload estimates differ from TCP rates, on these paths Pathload would appear to always be correct, though not tell us anything useful about the path conditions. To correct for this problem, we substituted results from our tool for pathload numbers on paths where Pathload reported a lower range of zero, and our tool reported a tighter range. On all other paths, we used Pathload results exclusively.

We ran Pathload with a resolution parameter of 2 Mbps. As a result, the difference between high and low range esimates would typically be no smaller than 2 Mbps. For slower paths where we did not use Pathload's estimates, the upper range and lower range were two to three hundred Kbps apart.

Figure 2 compares the available bandwidth as computed by Pathload to the BTC as determined by TCP rates. We have found that BTC is often significantly less than available bandwidth. Without any further corrections, Pathload estimates were within 20% of the TCP rate in only 20% of paths. As the remainder of the TCP rates were far less than the Pathload estimates, the results support the assumption that TCP does not use all available bandwidth.

3) Small TCP transfers: Can a short TCP transfer provide a good estimate of the average of a longer connection? Our experiment included two TCP bulk transfers. Figure 3 shows results of using prefixes of the first connection to predict the rate of the complete second transfer. When either 1MByte or the full 2 MBytes of the first transfer are used to predict the second, the rates differ little. When only the first 128 or 256K of data is considered, estimates are much poorer. This result is to be expected, as TCP's slow start algorithm can overshoot the true BTC

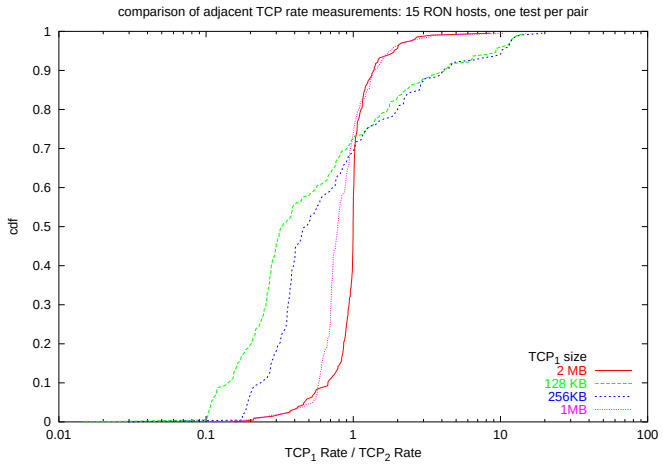


Fig. 3. Ratio of throughput acheived by first tcp transfer to second tcp transfer

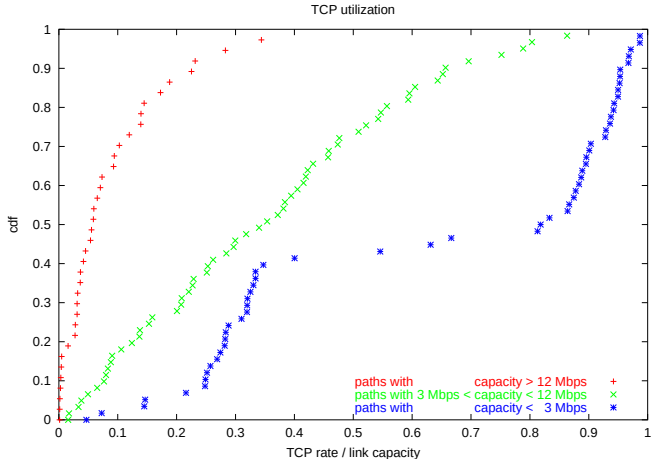


Fig. 4. BTC as a fraction of Path Capacity.

by as much as a factor of two. Also, any packet losses in the first few round trips will likely cause require a retransmission timeout and throw off an estimate greatly.

4) Path Capacity: Any of the recent packet pair tools can determine the path capacity between end hosts fairly well. We found that PathRate works quite well under most conditions. With the exception of paths involving M1MA, the capacity estimates matched information we had about the host's internet connection. In the case of M1MA, PathRate failed to provide an estimate until we relaxed its constraints regarding the allowed number of lost probes.

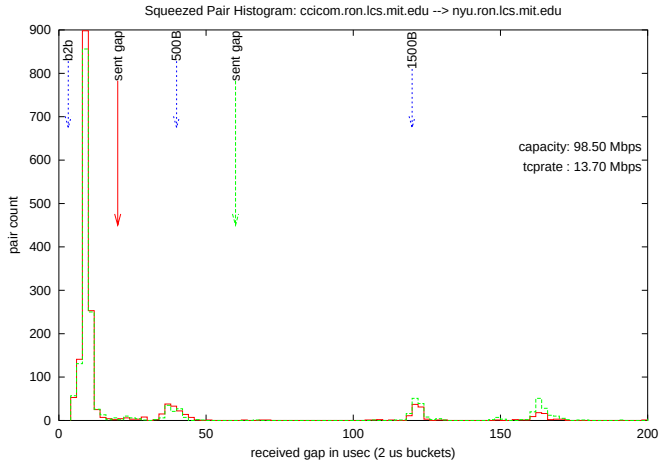
The path capacity between hosts will normally change only in the case of routing changes. Our tests do include one host, PDI, which has an internet connection that changes anywhere between 3 and 30 MBps on the scale of a few hours. Paxson [18] found that many routes are stable over a scale of hours. We expect that path capacity will change even more infrequently than routes, as the capacity limiting link is often a last-mile link. These access paths tend not to change on time scales smaller than weeks or months.

Capacity measurements can be conducted fairly, and are valid for long periods. How much does BTC differ from path capacity? We have found that the difference is largely dependent on the speed of the link in question. Figure 4 shows BTC as a fraction of the path capacity, broken up along three different capacity ranges. The highest range, $\geq$ 12 Mbps, primarily includes hosts on the Internet2, or well connected businesses. These paths often serve a great number of users, and may operate near capacity at all times. Even when the paths are largely idle, TCP has trouble utilizing all possible bandwidth without careful attention to implementation parameters. The middle range, 3 to 12 Mbps, includes cablemodem downlinks, and 10Mbps ethernet hosts, and inlcludes most of the transcontinental paths. The slowerest range primarily includes cablemodem uplinks and both directions of dsl links. The slower links are often otherwise unused, and so the BTC is nearly all of the path capacity. We believe that the pattern of rates we have found is largely a reflection of access methods that are popular now, and will likely change over time. However, similar observations will likely still hold.

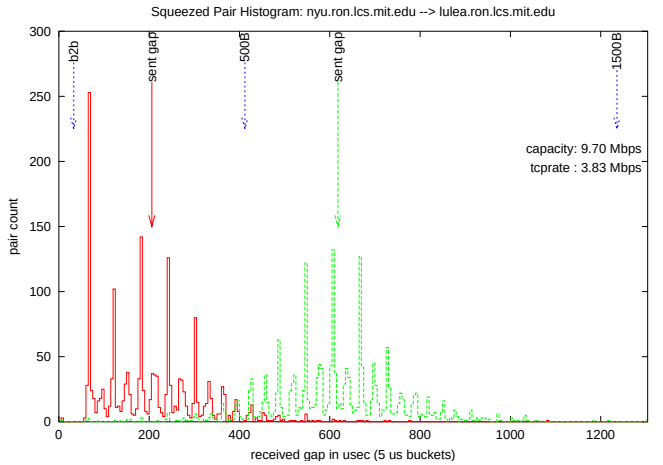
We examined the predictive ability of capacity to pick the path with higher BTC for all pairs of paths. We found that capacity picked correctly in 73% of all possible path pairings. If the possible choices include a common hosts, the path capacities are likely to be identical, which makes such a comparison impossible.

5) cprobe: One phase of our measurements is quite similar to cprobe [13]. We send a number of large back-to-back packets from sender to receiver, and compute the average arrival rate based on packet interarrival times. We found that this method is not very useful as a BTC measurement per se, as it nearly always overestimates the BTC, but it is quite useful to get a quick estimate of a rate somewhere between the BTC and path capacity.

6) Squeezed Pairs: Figure 5 shows two common histogram shapes we found when run between differen RON hosts. Each plot shows results for a single path. The solid line shows pairs that were sent spaced by half the transmission time for a 1500 byte packet. The long solid arrow marked 'sending gap' shows the gap between pairs at the sender for these packets. The dashed arrow marked 'sending gap' shows the sending gap for pairs sent at the half width of a 500 byte packet, which are plotted with dashed lines. Pairs plotted in solid lines that are to the left of the solid arrow arrived squeezed at the receiver. There are three other arrows on each graph. The leftmost is marked as 'b2b.' This arrow marks the expected arrival gap at the



(a) A path with tight and narrow links of the same speed



(b) A path with tight and narrow links of differing speeds

Fig. 5. Typical Squeezed Pair Histograms

receiver if packets exit the narrow link back-to-back and maintain this spacing at the receiver. Pairs may arrive with a spacing smaller than this arrow if the pair were squeezed together after the tight link. The arrow marked as '500B' shows the expected arrival gap if the probe packets are spaced by a 500 byte packet on the tight link. The '1500B' arrow shows the same value for 1500 byte cross traffic.

In Figure 5(a) only a very few packets arrive at the receiver with the same spacing at which they were sent. The other peaks are around 10 μs, 40 μs, 120 μs, and 160 μs. All of the links between ccicom and NYU are at least 100 Mbps or faster. At 100 Mbps, a 1500 byte packet takes 120 μs, and a 500 byte packet takes 40 μs. We therefore assume that packets which arrived at the sender spaced by 40 μs were queued with a 500 byte packet between the two probes. Those that arrived spaced by 120 μs either had one 1500 byte packet or three 500 byte packets between them. The lack of any prominent peak around 80 μs suggests the latter. There is very little difference between the histograms of packets sent with a 20 μs spacing and those sent with a 60 μs spacing. The large mode around zero indicates that in many cases, the probe packets were squeezed together, and essentially arrived back-to-back at the receiver.

In contrast, Figure 5(b) shows almost no overlap between the histograms for the two different sending gaps. In this case, the received gaps are spread symmetrically around the sending gap, with submodes at even intervals. These patterns result when queuing occurs on a faster link along the path than the end-to-end capacity limiting hop. The modes to the right of the sending gap result when the first packet of the pair is queued less than the second. The space between sub-modes can be expressed simply in terms of the size of cross traffic packets. Since there are no packets that arrive with gaps appropriate for cross traffic on the 10 Mbps link, we conclude that all queueing delay occurs on links with speeds higher than 10 Mbps. This data does not imply that the 10 Mbit hops are idle, merely that they are idle enough that our probe packets are never queued there as a result of cross traffic.

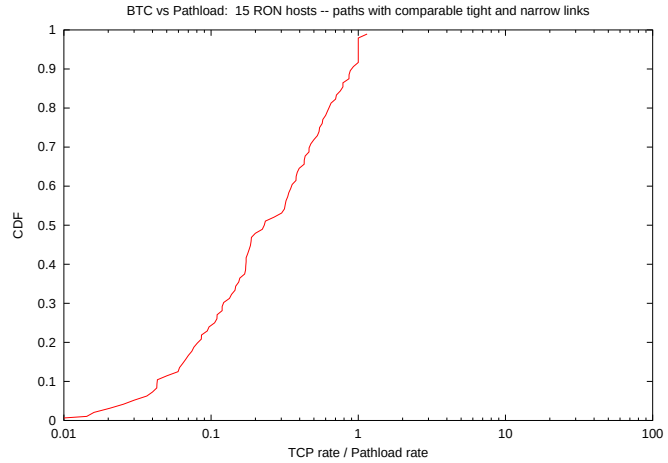
Most plots of Squeezed Pair histograms show features in one or both of the plots shown. Some do not show distinct sub-modes around the sending gap, but instead show a gradual smear up to a peak at the sending rate. This effect can occur when the the link where queues form is much faster than the capacity limiting link. In this case, the histogram bins are not fine enough to see a single packet gap. Paths where cross-traffic packet sizes are uniformly distributed rather than only a few discrete sizes would cause the same plots.

We have developed a simple scoring program to describe whether or not the histogram modes are symmetric around the sending gaps. Instead of searching for a peak around the sending rate, check each histogram bin to see whether there is a mode in one or both of the sending rate plots. If a mode appears in both plots, we increment a counter. If a mode appears in only one plot, we decrement the same counter. At the end if the counter is positive, we group that path in those like in figure 5(a), and call these paths tight-like-narrow paths. If the counter is negative, we label the path as tight-unlike-narrow at the time the test was run.

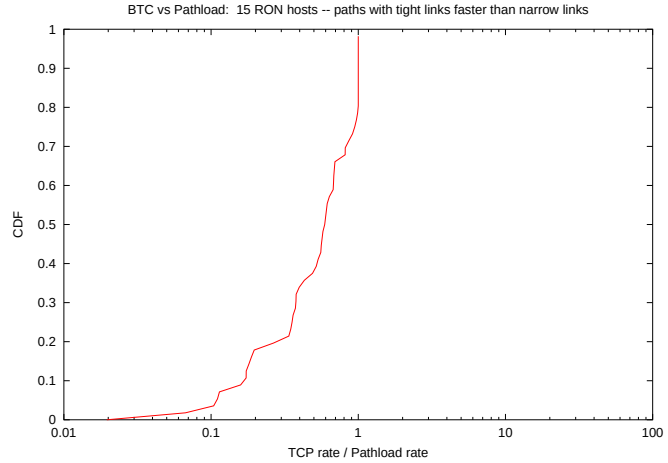
B. *Combined Methods*

When different methods are combined

1) Pathload with Squeezed Pairs: This test combines the path characterization done with squeezed pairs with

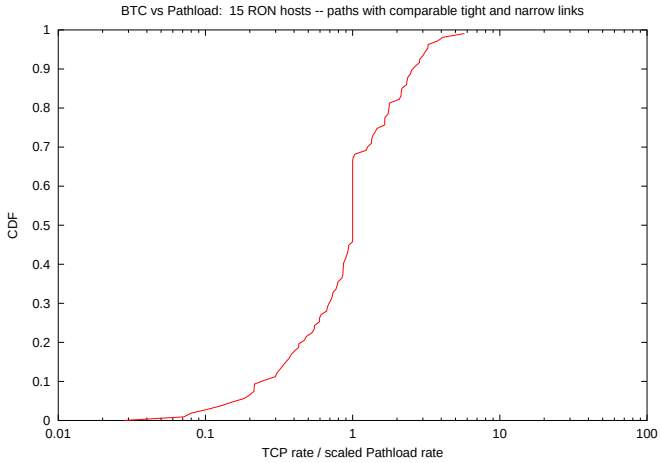


(a) Paths with tight-like-narrow Squeezed Pair results

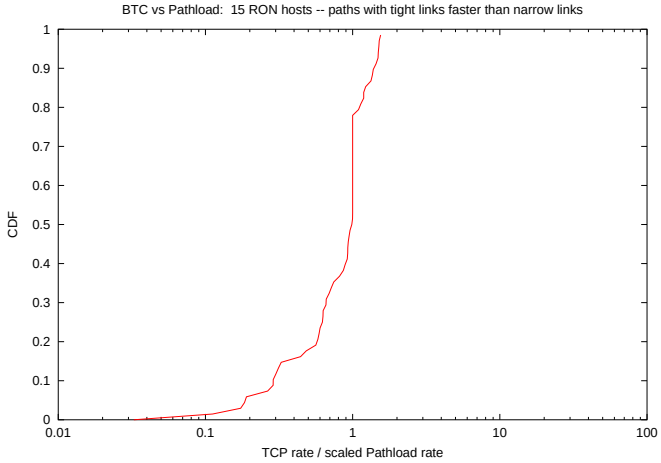


(b) Paths with tight-unlike-narrow Squeezed Pair results

Fig. 6. Pathload ranges split based on Squeezed Pair histograms



(a) Pathload ranges for tight-like-narrow paths, scaled by 0.2



(b) Pathload ranges for tight-unlike-narrow paths, scaled by 0.6

Fig. 7. Pathload ranges scaled and split by Squeezed Pair histograms

the comparison of Pathload rates to TCP rates. Figure 6(a) shows Pathload results for paths marked as tight-like-narrow. 61% of paths are included in this plot. Figure 6(b) shows Pathload results for paths marked as tight-unlike-narrow. 39% of paths are included in this plot. This cumulative distribution is much tighter: in 80% of these paths, TCP achieves at least one third of the Pathload reported speed. For tight-like-narrow paths, the 80th percentile is near a tenth of the Pathload estimate.

If we scale the pathload ranges before compairing them to actual tcp rates, by 0.2 for tight-like-narrow paths (figure 7(a)), and by 0.6 for tight-unlike-narrow paths (figure 7(b)), then we achieve a better predictor for BTC. When scaled, the tcp rates in 53% of tight-like-narrow paths fall between 50% below the pathload estimate and 50% above. For tight-unlike-narrow paths, 77% occur in the $\pm$ 50% range.

Next we scale and then combine the pathload estimates, and arrive at the distribution shown in figure 8. With pathload estimates combined in this manner, 50% of TCP rates will fall within 20% of the pathload estimate range. However, for 80% certainty, TCP rates vary from one quarter the scaled pathload estimates to double the estimate.

Using squeezed pairs to refine pathload estimates and then picking the better path based on the scaled rates does not improve estimates enough to be useful. If choices are based on Pathload's average rates, the choice is correct in 79.1% of paths. Scaling the Pathload estimates as we have done improves this only to 80.4%. This change is so small because there are relatively few paths where the order of the rate estimates changes as a result of the squeezed pair

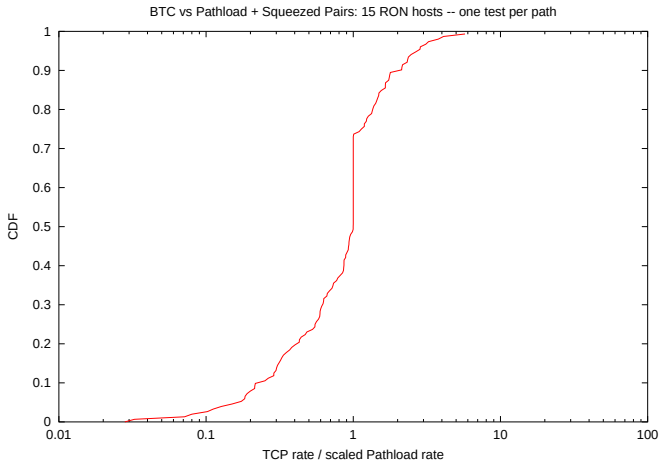


Fig. 8. Squeezed pairs with combined, scale Pathload

scaling.

2) Pathload with Loss Rate and Round Trip Time: We have examined using loss rates and round trip time to refine estimates from Pathload. None of these showed a significant improvement. On paths with a long round trip time, greater than 150 milliseconds, pathload estimates could be improved slightly by scaling estimates down. These cases are all examples of transcontinental paths. As the RON testbed only included 4 hosts not in the US, drawing conclusions based on only these few is questionable.

VI. Analysis

Our measurement results show that Pathload predictions are often significanly higher than average rates TCP connections on the same path achieve, indicating that BTC and available bandwidth often differ greatly.

We have found that Squeezed Pairs provide useful information about path conditions, and can successfully use this information to pick paths were Pathload results are closer to BTC measurements than other paths. Squeezed Pair tests group paths depending on where flows experience congestion. When congestion occurrs near the network endpoints, TCP is much slower than Pathload estimates. When congestion occurs away from narrow links, TCP rates are much closer to Pathload estimates.

We believe that there are two main causes behind these observations. One factor is the degree of sharing between flows. When limiting links are on slower paths near network edges, there are likely to be only a few competing flows, and thus a new TCP connection can capture a relatively larger fraction of the unused capacity. The other factor is the speed of the links in question. Paths with faster access types are more likely to have congestion occur on links with speed similar to the capacity limiting hop. TCP connections are not as efficient at utilizing spare capacity on faster paths.

Pathload estimates combined with squeezed pairs are similar to short TCP timing estimates in both quality and network resources consumed. We believe that there is considerable redundancy present in both Pathload and Squeezed Pair measurements. These properties encourage further refinement of both tools for estimation purposes.

Acknowledgments

Dina Katabi suggested using Squeezed Pairs as a method to measure available bandwidth. David Andersen provided us with access to the RON testbed.

References

[1] M. Jain and C. Dovrolis, "Pathload: A measurement tool for end-to-end available bandwidth," in *Passive and Active Measurements*, Fort Collins CO.

[2] David Andersen, Hari Balakrishnan, M. Frans Kaashoek, and Robert Morris, "Resilient overlay networks," in *Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP '01)*, Chateau Lake Louise, Banff, Canada, October 2001.

[3] C. Dovrolis, P. Ramanathanm, and D. Moore, "What do packet dispersion techniques measure?," in *INFOCOM*, 2001.

[4] M. Jain and C. Dovrolis, "End-to-end available bandwidth: Measurement methodology, dynamics, and relation with tcp throughput," in *ACM SIGCOMM*, Pittsburg, PA, 2002.

[5] Mark Allman, "Measuring End-to-End Bulk Transfer Capacity," in *ACM SIGCOMM Internet Measurement Workshop*, November 2001.

[6] M. Mathis and M. Allman, "A framework for defining empirical bulk transfer capacity metrics," RFC 3148.

[7] Van Jacobson, "Congestion avoidance and control," in *ACM SIGCOMM '88*, Stanford, CA, Aug. 1988, pp. 314–329.

[8] M. Allman, V. Paxson, and W. Stevens, "Tcp congestion control," RFC 2581.

[9] Jitedra Padhye, Victor Firoiu, Don Towsley, and Jim Krusoe, "Modeling TCP throughput: A simple model and its empirical validation," in *ACM SIGCOMM '98 conference on Applications, technologies, architectures, and protocols for computer communication*, Vancouver, CA, 1998, pp. 303–314.

[10] M. Goyal, R. Guerin, and R. Rajan, "Predicting TCP Throughput From Non-invasive Network Sampling," in *Proceedings of IEEE INFOCOM*, New York, June 2002.

[11] Kevin Lai and Mary Baker, "Nettimer: A tool for Measuring Bottleneck Link Bandwidth," in *USENIX Symposium on Internet Technologies and Systems*, March 2001.

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

[13] Robert L. Carter and Mark E. Crovella, "Dynamic server selection using bandwidth probing in wide-area networks," Tech. Rep. TR-96-007, Boston University Computer Science Department, 1996.

[14] Van Jacobson, "Pathchar," ftp://ftp.ee.lbl.gov/pathchar/.

[15] M. Mathis, "Treno bulk transfer capacity," draft-ietf-ippm-treno-btc-03.txt (Internet-Draft Work in progress).

[16] Tobias Oetiker and Dave Rand, "Multi router traffic grapher," http://people.ee.ethz.ch/ oetiker/webtools/mrtg/.

[17] Iris Baron and Jacob Strauss, "Relations between packet bursts and available bandwidth," MIT course 6.829 final project: Computer Networks.

[18] Y. Zhang, N. Duffield, V. Paxson, and S. Shenker, "On the Constancy of Internet Path Properties," in *ACM SIGCOMM Internet Measurement Workshop*, November 2001.

[19] J.-C. Bolot, "End-to-End Packet Delay and Loss Behavior in the Internet," in *Proceedings, ACM SIGCOMM Conference*, San Francisco, CA, August 1993.

[20] K. Claffy, G. Miller, and K. Thompson, "the nature of the beast: recent traffic measurements from an internet backbone," in *Proc. of INET'98*, 1998.