Measuring Bottleneck Link Speed in Packet-Switched Networks

Robert L. Carter and Mark E. Crovella
Computer Science Department, Boston University
111 Cummington St., Boston MA 02215
{carter,crovella}@cs.bu.edu

BU-CS-96-006
March 15, 1996

Abstract

The quality of available network connections can often have a large impact on the performance of distributed applications. For example, document transfer applications such as FTP, Gopher and the World Wide Web suffer increased response times as a result of network congestion. For these applications, the document transfer time is directly related to the available bandwidth of the connection. Available bandwidth depends on two things: 1) the underlying capacity of the path from client to server, which is limited by the bottleneck *link; and 2) the amount of other traffic competing for links on the path. If measurements of these quantities were available to the application, the current utilization of connections could be calculated. Network utilization could then be used as a basis for selection from a set of alternative connections or servers, thus providing reduced response time. Such a dynamic server selection scheme would be especially important in a mobile computing environment in which the set of available servers is frequently changing.*

In order to provide these measurements at the application level, we introduce two tools: bprobe*, which provides an estimate of the uncongested bandwidth of a path; and* cprobe*, which gives an estimate of the current congestion along a path. These two measures may be used in combination to provide the application with an estimate of available bandwidth between server and client thereby enabling application-level congestion avoidance.*

In this paper we discuss the design and implementation of our probe tools, specifically illustrating the techniques used to achieve accuracy and robustness. We present validation studies for both tools which demonstrate their reliability in the face of actual Internet conditions; and we give results of a survey of available bandwidth to a random set of WWW servers as a sample application of our probe technique. We conclude with descriptions of other applications of our measurement tools, several of which are currently under development.

1 Introduction

For many applications, the quality of network connections can have a large impact on performance. One important characteristic of a network connection is the bandwidth available to clients using that connection. In particular, for document transfer applications (*e.g.* FTP, Gopher and the World Wide Web) higher available bandwidth implies faster document transfer time. Available bandwidth depends on two things: 1) the underlying capacity of the path between client and server which is limited by the slowest (or *bottleneck*) link, and 2) the presence of competing traffic (*congestion*).

One technique to minimize the time between document request and document arrival begins with an evaluation of the quality of a set of connections to candidate servers. The application may then choose to retrieve the document from the server with the highest quality connection. In this context, the quality of the network connection is determined by an estimate of network utilization which can be used to assess potential transfer time. Utilization, in turn, depends on both the bandwidth of the path (limited by the bottleneck link) and the presence of other traffic competing for the path.

However, neither of these two useful pieces of information are readily available to applications. In order to discover this information we developed two tools: *bprobe*, which measures the uncongested bandwidth of the bottleneck link of a connection; and *cprobe*, which estimates the current congestion along the bottleneck link of the path. When applications are provided with measurements of the current network state, application-level congestion avoidance becomes possible. By application-level congestion avoidance we mean that applications can use estimates of traffic volume to actively avoid paths that are currently heavily loaded.

The measurement of bottleneck link speed involves sending a series of ICMP echo packets from source to destination and measuring the inter-arrival times between successive packets. Under certain conditions analysis of the distribution of the inter-arrival times leads to a straightforward, yet reliable, calculation of the bottleneck link speed of the connection. *Bprobe* is specifically designed to make the necessary conditions occur and performs the analysis required to make a robust estimate of bottleneck link speed.

The method of measurement of competing traffic used by *cprobe* also relies on echo packets. An estimate of congestion can be made based on the elapsed time between the return of the first and last of a series of packets and the amount of data sent by the probe tool. *Cprobe* is designed to make a reliable estimate of competing traffic under real network conditions.

In the following sections we motivate and describe *bprobe* and *cprobe*. We present the theory behind the tools and explain the obstacles to making reliable measurements in practice. We then explain how the the probe tools were designed to overcome these obstacles. We also describe the validation process in which we compared the measurements made by our tools to known link capacities on local, regional and wide-area networks. We present the results of a survey of WWW servers and discuss the implications of our findings for replication and server selection. We conclude with a discussion of ongoing and future work.

1.1 Related Work

In [2], the author gives a model for packet travel through links and routers along a connection in the Internet. The bandwidth of the bottleneck link is studied by comparing the round-trip times (rtt) of adjacent packets from a sequence sent at regular intervals. The rtt's of successive packets are plotted against each other (rtt_i vs rtt_{i-1}). The inverse of the slope of a line fitted to this data gives the bandwidth and the intercept gives the latency. The line fitting is done by hand and a very large number of packets are needed to make manual fitting possible. An expanded version of that paper gives preliminary suggestions as to how this process can be used to determine the makeup of interfering traffic [1]. However, many obstacles exist to the automatic application of this idea in real networks. Our work overcomes these obstacles and results in a robust procedure that can calculate the bottleneck link speed with minimal overhead.

The author in [6], defines the notion of a *packet-pair*, two back-to-back packets that travel from source to destination and result in two acknowledgment packets returning. The calculation of bandwidth based on the returning ACKs is similar to ours but in that work, a network of Rate-Allocating Servers (RAS) is assumed. With RAS each incoming stream of packets is served in turn with one packet processed from each queue in its turn if any packets are present. This in effect shares the packet processor equally among all incoming lines. Thus, the actual bandwidth available to the source can be accurately measured since the inter-arrival time of two packets from the same source will be increased by the processing time of one packet from each competing input line in use. In contrast, we assume the conditions prevalent in the current Internet, which are much less tractable, and we require only that packets not be reordered in transit.

Treno [3] is a tool from Pittsburgh Supercomputer Center that emulates a TCP connection including slow-start and necessary retransmissions. They have set up a Web-based server that measures the path back to the calling client. Using a scheme similar to *traceroute*, it sends UDP packets with increasing TTL along the path from the server to the invoking client. Then it reports the available bandwidth that TCP would find along each of the path prefixes to the destination. In order to get the full effect of TCP, it is recommended that the tool be run for at least 10 seconds of continuous traffic. In contrast, our *cprobe* tool measures the estimated available bandwidth without regard to any particular higher-level protocol while our *bprobe* tool measures the underlying bottleneck link speed. In addition, we find that our measurement process is typically faster than the 10 seconds necessary for *treno* to reliably measure the bandwidth available using TCP.

2 Measuring Bandwidth and Congestion at the Application Level

We use the term *base bandwidth* of a connection to mean the maximum transmission rate that could be achieved by the connection in the absence of any competing traffic. This will be limited by the speed of the bottleneck link, so we also refer to this as the *bottleneck link speed*. However, packets from other connections may share one or more of the links along the connection we want to measure; this competing traffic lowers the bandwidth available to our application. We use the term *available bandwidth* to refer to the estimated transfer rate available to the application at any instant. In other words, the portion of base bandwidth which is not used by competing traffic. We define the *utilization* of a connection as the ratio of available bandwidth to base bandwidth, that is, the percentage of the base bandwidth which should be available to the application.

If a measure of current utilization is available, applications can make informed resource allocation decisions. For the specific problem of WWW server selection, knowledge of current network utilization allows better prediction of information transfer time from the candidate servers.

With this objective in mind, we set out to develop tools to measure the current utilization of a connection. These measurements are necessarily done from a distance and in a unknown environment. Therefore, we refer

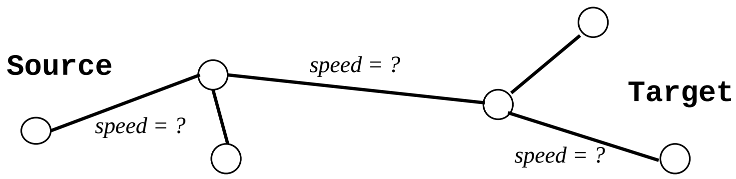

Figure 1: Path from Source to Target. Initially, the link speeds are unknown.

to our measurement process as *probing* and the tools as *probes*. Our design goals for the probes were that they:

- should work at the application level; *i.e.* by usable by any user program;
- should have a minimal impact on both the network and the server we are probing;
- should not rely on cooperation from the network or the server;
- should be robust and perform properly under varying conditions; and
- should be as accurate as possible while providing timely information.

The following sections describes the design and validation of BPROBE and CPROBE.

2.1 Bprobe: Measuring Base Bandwidth

2.1.1 Measurement Theory

In the following discussion we assume a network like the Internet. In such a network, a *path* between any two hosts in the network is made up of one or more *links*. Between each set of links along the path is a *router* which examines the destination address of each packet and forwards it along the appropriate outgoing link. Our main requirement of the network is that packets are not frequently reordered in transit. We also assume that the path is stable, by which we mean that the path packets take at any instant will not change for the next few seconds, at least. For the Internet, routing table updates are infrequent enough that this is a reasonable assumption. We further assume that the bottleneck in both directions is the same link, although this assumption could be relaxed in a different design.

Recall that the output of BPROBE is a measurement of the base bandwidth of a connection: the speed of the bottleneck link. The situation is as presented in Figure 1. In order to compute utilization, it would be most useful to know the minimum speed among the links along the path from the Source to the Target. The method used by BPROBE is to send a sequence of ICMP ECHO packets from the source to the target and measure the inter-arrival times of the returning packets.

Figure 2 illustrates the journey of a pair of packets along the round-trip path from source to target and back. When the packets depart from the Source host, the inter-departure gap is measured as $(d_2 - d_1)$. As the packets flow through intermediate routers, the inter-packet gap may change as the packets flow over links of various

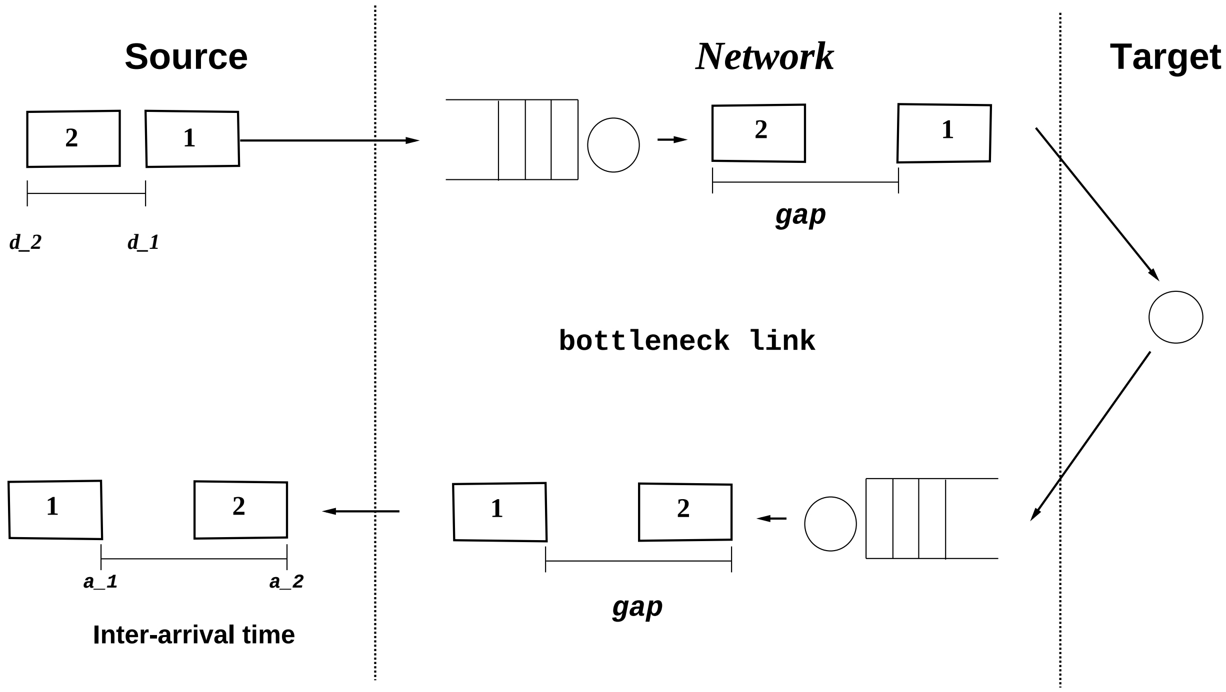

Figure 2: Flow of BPROBE packets from Source to Target and back. The inter-packet gap is stretched by the bottleneck link. The inter-arrival time measured at the Source can be used to calculate the bottleneck link speed.

capacities. In general, the size of the gap varies inversely with the capacity of the link. Figure 2 shows that the gap between the packets is stretched when the packets flow through the bottleneck link. On the return trip the packets flow again over the bottleneck link and the gap is unchanged. When the packets return to the Source, the inter-arrival time $(a_2 - a_1)$ reflects the speed of the bottleneck link.

The essential idea behind the probe, then, is this: if two packets can be caused to travel together such that they are queued as a pair at the bottleneck link, with no packets intervening between them, then the inter-packet spacing will be proportional to the processing time required for the bottleneck router to process the second packet of the pair. This well-known effect is illustrated in the familiar diagram shown in Figure 3 (adapted from Van Jacobson [5]). In addition, Bolot describes the basic effect in [1, 2] and Keshav has used a similar method in networks of Rate-Allocating servers [6].

The goal of the BPROBE tool is to create this condition and to use it to make reliable measurements. In other words, if packets from the probe tool alone are queued at the bottleneck link, then the inter-arrival time of those pairs that were queued can be used at the endpoint of the path to estimate the base bandwidth of the bottleneck link. Under ideal conditions, the receiver can use this information to measure the bottleneck link speed as follows: the trailing edge of the first of a pair of packets marks the time when the router started processing the second packet of the pair and the trailing edge of the second packet records the time when the router finished processing that packet. Given a packet of size $\mathcal{P}$, and the inter-arrival time (or *gap*), we can estimate the base bandwidth

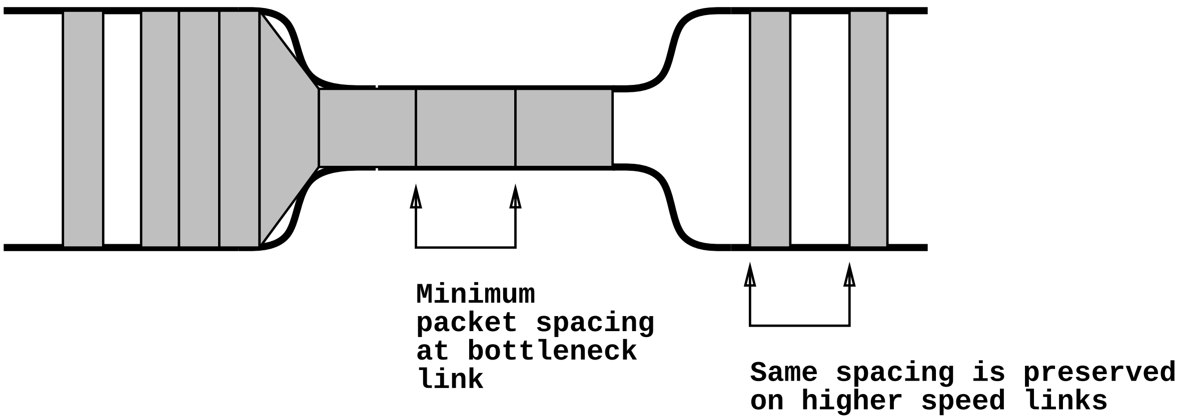

Figure 3: Illustration of packet flow through a bottleneck link.

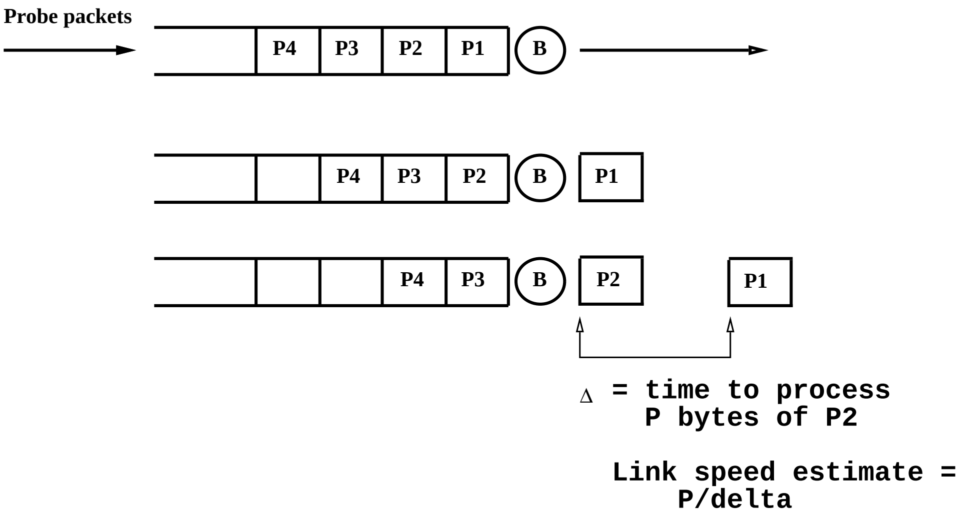

Figure 4: Ideal case: probe packets alone queued at router.

of the bottleneck link, $\mathcal{B}_{bls}$, as follows

$$\mathcal{B}_{bls}\ \textbf{bytes/second} = \frac{\mathcal{P}\ \textbf{bytes}}{gap\ \textbf{seconds}}.$$

This is illustrated in Figure 4 which shows the situation when probe packets alone are queued at a router.

2.1.2 Obstacles to Measuring Base Bandwidth in Practice

However, in contrast to the the network architecture assumed in [6], in our experimental environment (the current Internet) this ideal behavior is not easily achieved. There are several problems that arise in practice:

- **Queuing failure:** Probe packets may not queue at the bottleneck link.

- **Competing traffic:** Competing traffic along the path may intervene between probe packets.

- **Probe packet drops:** Packets sent by the probe may be dropped.
- **Downstream congestion:** Congestion at routers downstream from the bottleneck may invalidate the results.

Each of these problems can be the cause of spurious estimates of base bandwidth. The major obstacle to implementation, then, is this: given a set of inter-arrival time measurements, how can the probe tool decide which will result in valid bandwidth estimates? The challenge was to start from the intuitive idea captured in Figure 3 and design an accurate, robust and low-impact tool to measure bandwidth. We now present our solutions to these problems.

2.1.3 Solutions to Implementation Problems

Both BPROBE and CPROBE are built on top of the ICMP ECHO mechanism. Because of our use of ICMP ECHO, a client can send packets to a host and receive replies without installing new software at the remote site, which affords wide utility of our tools.

There are two principal techniques with which we attack these problems. First, the use of multiple packets of varying sizes; second, the tool uses a careful filtering process which discards inaccurate measurements.

Queuing failure: The first problem to overcome is that the client sending the probe packets may not happen to send data fast enough to cause queuing at the bottleneck router. In order to ensure such queuing, we send a number of packets and we use packets of varying sizes. Larger packets will naturally take more processing time at routers and increase the possibility of queuing. Therefore, we want to use as large a packet as can negotiate the round-trip path. The upper limit on packet size will vary from path to path, however, so we use packets of varying size.

Currently, the probe runs in distinct phases with each phase using packets of successively larger sizes. The first phase sends a number of packets (currently 10) of the smallest size that will be used (currently 124 bytes). The next phase uses even larger packets and this process continues until we reach the maximum packet size which our test client can send (approximately 8000 bytes). In this way, we adapt to the maximum feasible size for each connection.

Competing traffic: The second problem is the interference of other traffic sharing a link along the path. In the ideal case, no non-probe packets intervene. One or more intervening packets will cause an increase in the inter-arrival time and therefore an underestimate of the bottleneck link speed. This is illustrated in Figure 5 which shows a non-probe packet queued between two probe packets. The resulting inter-arrival time measures the processing time for the $\mathcal{P}$ bytes of packet P2 as well as the unknown number of bytes contained in the intervening packet. Therefore, the estimated link speed will be an underestimate.

The solution to this problem is two-fold. By sending a large number of packets, we increase the likelihood that some pairs will not be disrupted by competing packets. Even when many pairs are disrupted, the number

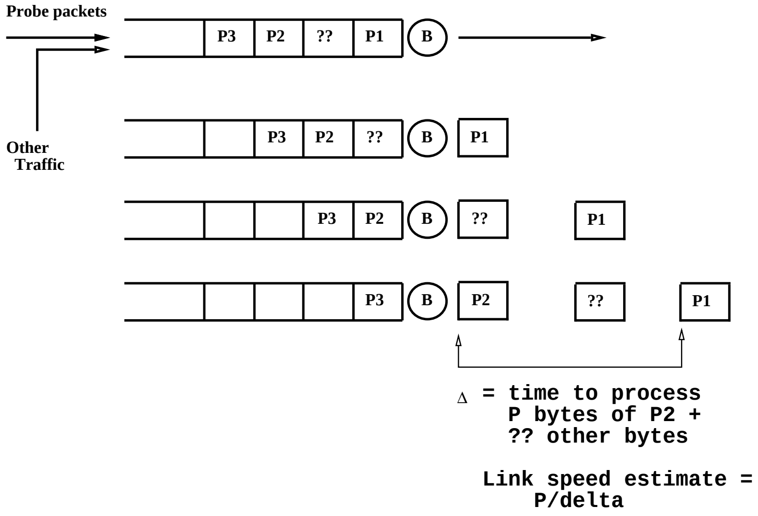

Figure 5: Competing packets intervening between probe packets cause an underestimate of link speed.

of intervening bytes will often vary from pair to pair. This results in differing bandwidth estimates which may then be filtered to determine the correct estimate. We explain the filtering process below.

As a further measure, we increase the packet size by alternating factors of 150% and 250%. This ensures that no two packet sizes are integer multiples of each other. If we simply doubled the packet size, the bandwidth estimated when one packet of size x intervenes between probe packets of size y would be the same as the bandwidth estimated when two packets of size x intervene between probe packets of size $2y$. As will be clear from the discussion of our filtering method, this could easily result in an erroneous final estimate. The use of alternating increments diminishes the probability of a bad estimate.

Probe packet drops: The third problem that must be addressed is dropped probe packets. Some packets are simply lost, but large packets are more likely to cause buffer overflow and be dropped. Large packets will be fragmented and this also increases the likelihood of dropped packets due to fragment loss. (Aside from this effect, fragmentation of large packets has not been a problem for our tools.) We avoid this problem by sending packets of varying sizes. Sending many packets of each size also serves to make the probe tolerant of a few packet losses regardless of the cause.

Downstream congestion: The fourth problem occurs when queuing occurs on the return trip, after passing through the bottleneck link. That is, even if all goes well from client to the server and back through the bottleneck link, congestion at intermediate servers downstream from the bottleneck can invalidate an estimate. Consider a pair of packets whose inter-packet gap was properly set by the bottleneck link. If these packets subsequently get queued, the inter-packet gap will be reset, thus measuring the wrong link. Since this subsequent queuing is unlikely, we can alleviate this problem during filtering. Some of the pairs may, in fact, measure the wrong link but if enough of the pairs make it back without further queuing, the erroneous estimates will be filtered out.

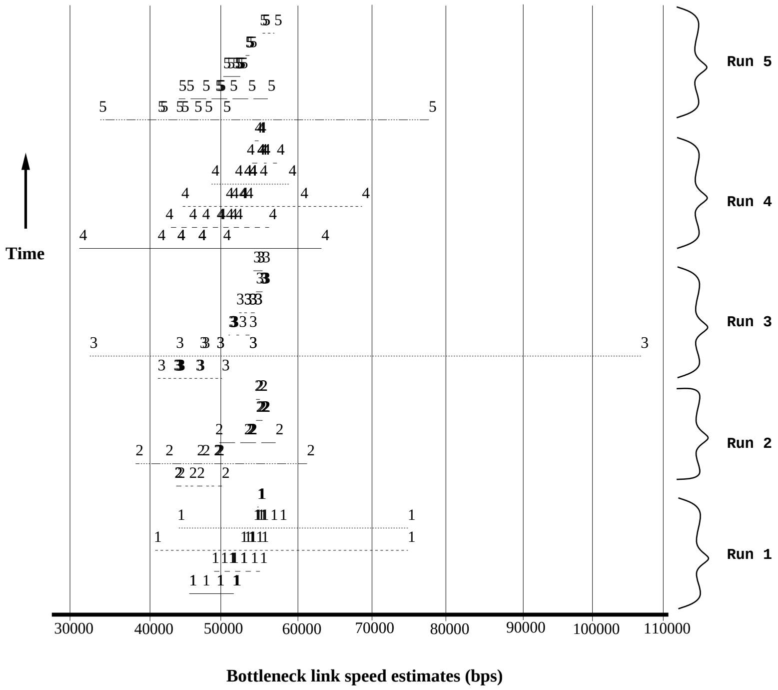

Figure 6: 5 BPROBE experiments to local 56Kbps hosts

The Filtering Process: The most difficult problem that must be addressed by the probe is identification of those estimates that should be used to determine the base bandwidth from those that should be discarded due to lack of queuing, subsequent queuing or interference from competing traffic. Each phase of 10 probe packets results in at most 9 inter-arrival measurements. Thus, at most 7 sets of at most 9 measurements each are the input to the filtering process. Given the packet size (and some intelligent guesses as to protocol headers and link level headers and trailers) each pair of packets generates one raw estimate of the bandwidth *assuming queuing* as defined before.

For example, Figure 6 shows the raw estimates for 5 invocations of the probe tool. All of the data points belonging to one invocation are shown by plotting a numeral representing the invocation number at each data point. The abscissa of a data point is the bandwidth estimate, and the ordinate increases with packet size. Thus, all the measurements for the first invocation occur on the bottom 5 line segments of the graph, all those for the second on the next higher 5 lines and so on. Within each invocation, packet size is increasing from bottom to top. Notice the clustering of estimate values as packet size increases; this shows that, in general, larger packet sizes yield more consistent results.

The multi-phase variable-packet-size design of the probe results in 1) correlation among correct estimates and 2) lack of correlation among incorrect estimates. For example, suppose all intervening packets are of size x, and the probe packet sizes for two probe phases are $p1$ and $p2$ with $p1 \neq p2$. The estimates produced by the packet sequences $p1 - x - p1$ and $p2 - x - p2$ will both underestimate the base bandwidth, but the important characteristic is that the two estimates will *differ*. On the other hand, sequences like $p1 - p1$ and $p2 - p2$ will produce agreeing correct estimates. Other incorrect estimates such as produced by the sequence $p1-x-x-x-p1$ will be even less correlated with the correct estimates and also uncorrelated with other. It is these properties of the data, which are evident in Figure 6, that we seek to exploit using non-linear filtering.

We have tested two filtering methods both of which rely on computing an "error interval" around each estimate, and subjecting these intervals to further set operations as explained below. The error interval can be expanded as necessary until a satisfactory estimate of the speed of the bottleneck link is determined. The *intersection* filtering method finds overlaps between estimate intervals and computes the intersection, with the idea being that we want to find the estimate that occurs in all sets. Since we observed that larger packets provide better estimates we start intersecting using the estimates from the largest packets and iteratively intersect with estimates derived from successively smaller packets. The *union* filtering method combines overlapping intervals using set union, and selects an interval only if enough sets contribute to it. Both methods produce one interval as the final result and the midpoint of this interval is returned as the final estimate.

The two methods are illustrated in Figure 7, in which two sets of estimates are combined using either Union or Intersection. In the top of the figure are histograms representing the two sets. The horizontal axis gives intervals of estimated bottleneck bandwidth (increasing to the right) and the vertical axis gives the number of measurements in each interval. The lower portion of the figure represents the histograms resulting from combination using the Union or Intersection operations. For the union result, the height of the histogram bar, h, and the number of sets which contribute to it, s, is shown under each interval using the notation: (h, s). For example, the fourth interval has a total of 7 measurements which originate in 2 sets. In this case the only interval that has members from more than one set is the $(7, 2)$ interval and the midpoint of this interval would be the resulting estimate of bottleneck bandwidth.

In our experience the union approach shows less dispersion of estimates than intersection and we have adopted it as the filtering method currently used by BPROBE.

2.1.4 Validation of BPROBE

To validate the base bandwidth probe tool, we performed three sets of experiments. We tested BPROBE between a fixed host and hosts on our local network, on our regional network, and on the wide-area network. For each of the networks we used available maps of link capacities to determine the bottleneck link speed for each of the target servers. Therefore, we were able to compare the results of the tool against the known bottleneck link speeds.

Local validation: First, we tested the probe to a set of 16 hosts on our local network, 9 of which were connected by Ethernet and 7 of which were connected over 56Kbps lines. Each minute over a period of 4 days we ran the probe tool and recorded the bandwidth estimate. The results of these tests are summarized in Figure 8 and Figure 9 which presents histograms of BPROBE bandwidth estimates for the two classes. In both figures, the histogram bins are labeled with bottleneck link speeds in bits-per-second. Figure 8 shows the histogram

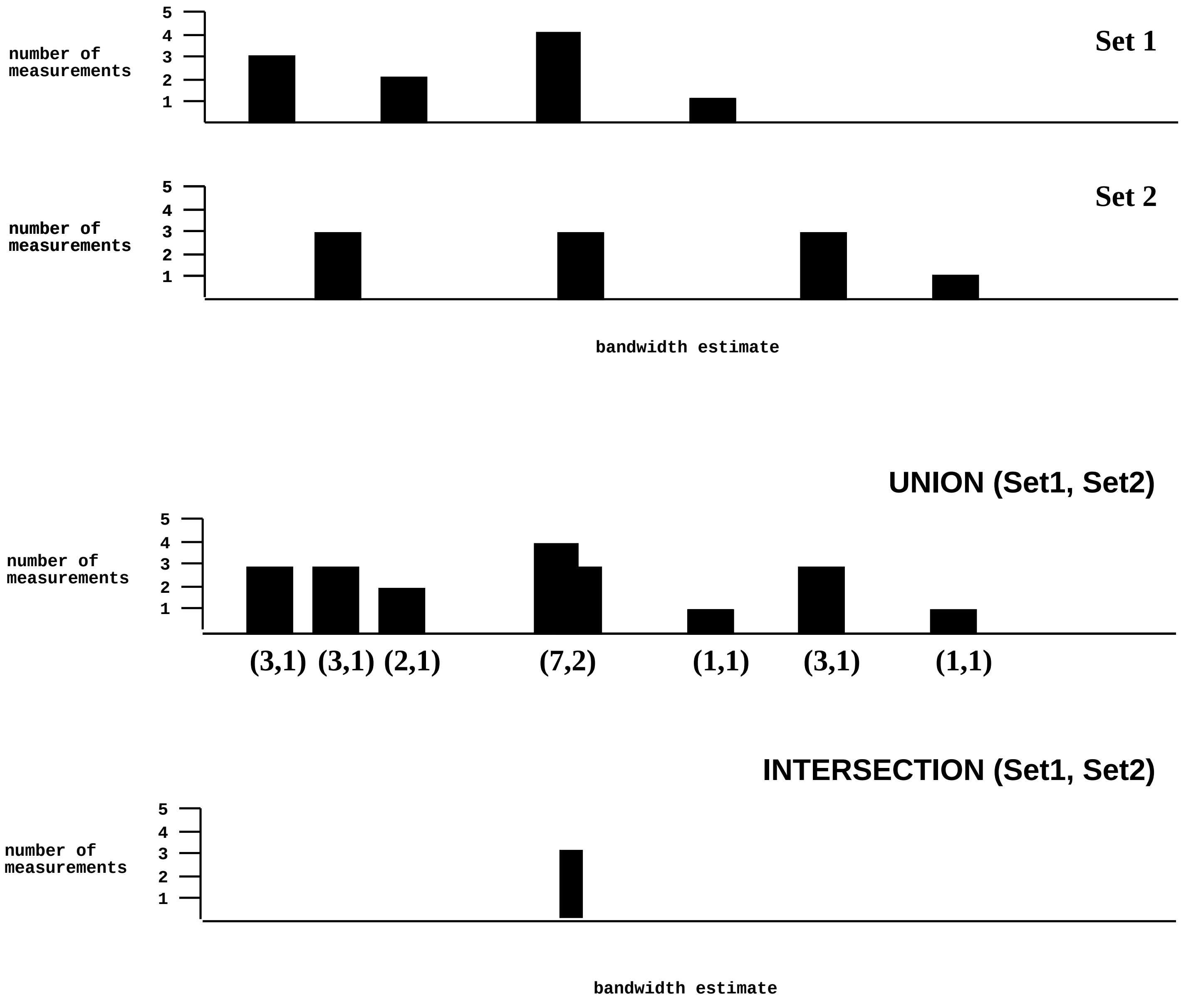

Figure 7: Filtering process: Union and Intersection operators

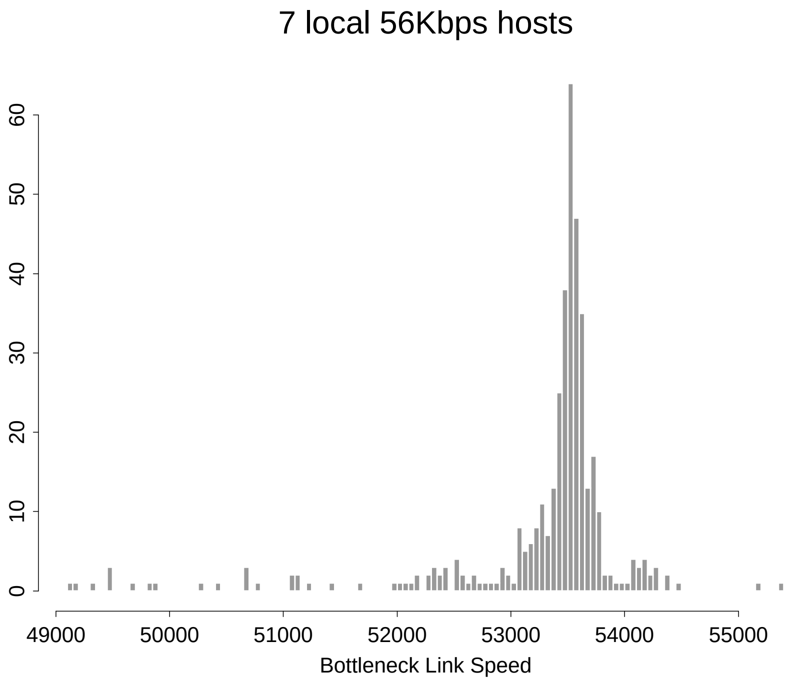

Figure 8: Histogram of BPROBE results: local 56Kbps hosts

of the estimated bottleneck link speeds for the 7 56Kbps hosts; Figure 9 shows the histogram of the estimated bottleneck link speeds for the 9 Ethernet (10Mbps) hosts. Both figures show the majority of the estimates closely clustered about the expected values, even though the expected values of the two host sets differ by two orders of magnitude.

Regional validation: The next set of tests was performed within NearNet, the regional network to which our site belongs.

Here we were able to add a third category of bottleneck capacity: T1. For the regional network we have sets of hosts with bottleneck link capacities of 56Kbps, 1.544Kbps (T1) and 10Mbps (Ethernet). Once again the measurements were made periodically over a few days and the results are presented as histograms of bottleneck link speed estimates.

Figure 10 shows the histogram of bottleneck bandwidth estimates for the hosts known to have a 56Kbps bottleneck. The histogram of bandwidth estimates for 681 probes of the 6 hosts shows a very tight grouping of estimates. Although there is a consistent underestimate, we still find 73% of the estimates within 5% of the theoretical capacity, 87% of the estimates within 10% of the theoretical capacity and 92% of the estimates within 20% of the theoretical capacity.

For the hosts with a T1 bottleneck, Figure 11 gives the histogram of 1156 probes of the 9 hosts. The picture is a little less clear for this case as there appear to be several minor peaks in the data. However, we still find most of the estimates close to the rated capacity. In particular, we find: 23% of the estimates within 5% of the

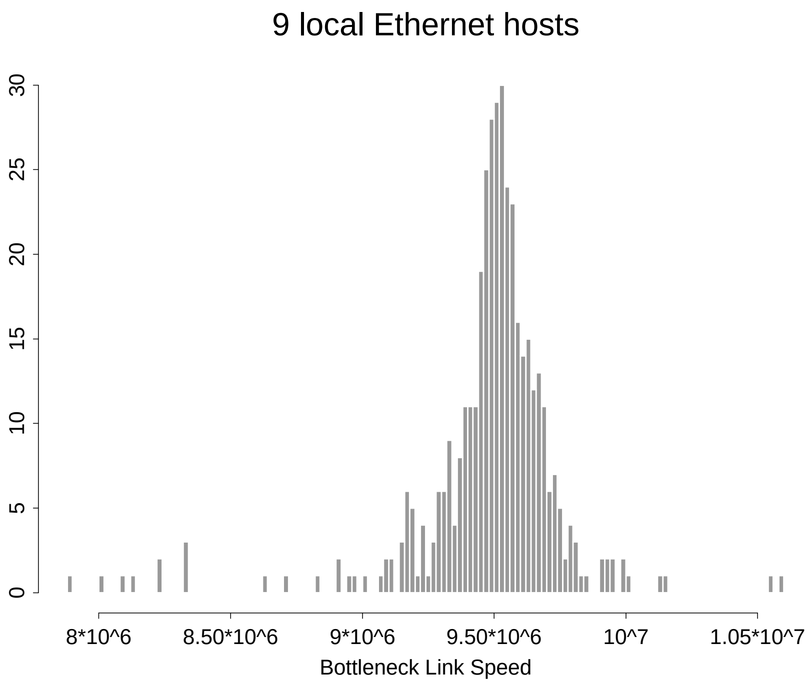

Figure 9: Histogram of BPROBE results: local Ethernet hosts

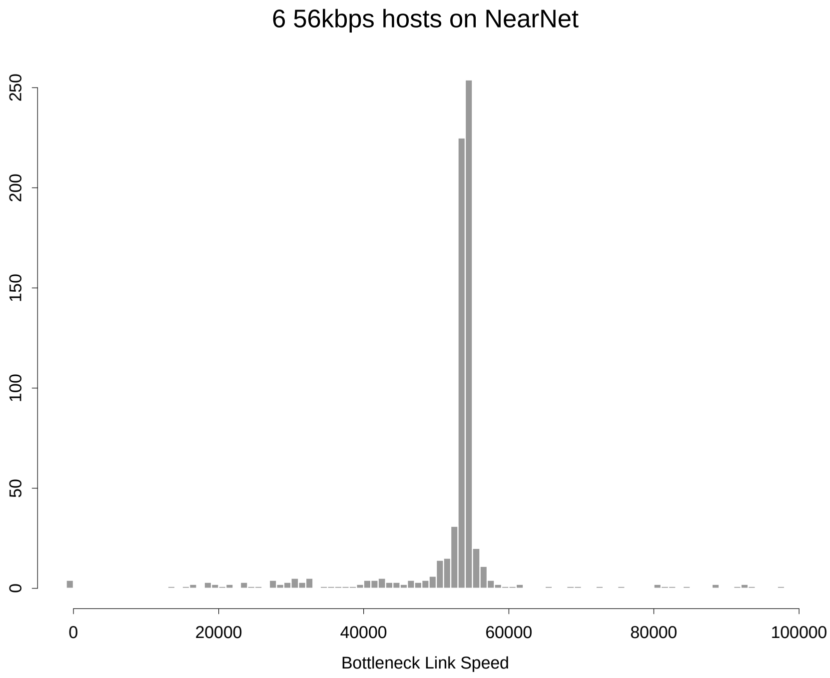

Figure 10: Histogram of BPROBE results: Nearnet 56Kbps hosts

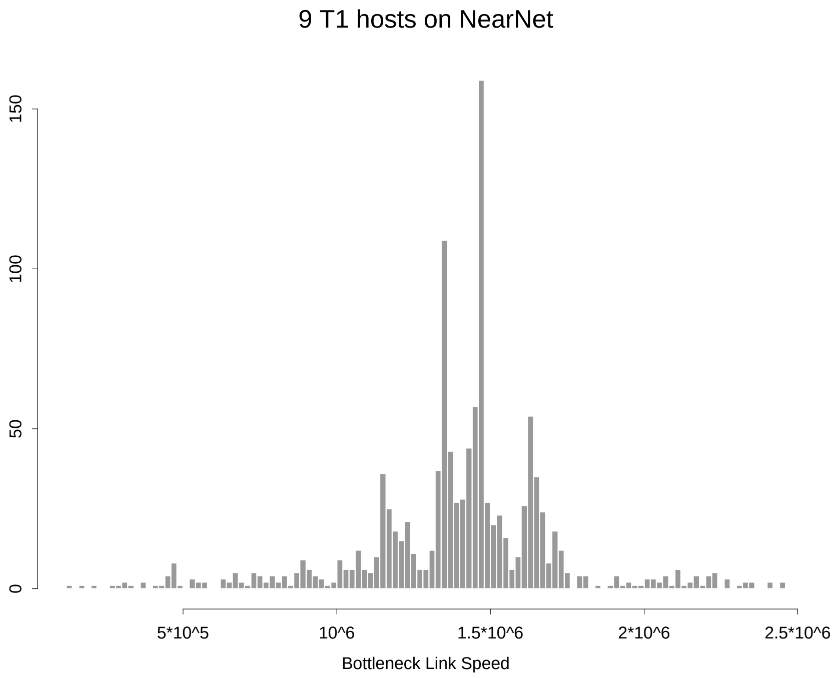

Figure 11: Histogram of BPROBE results: Nearnet T1 hosts

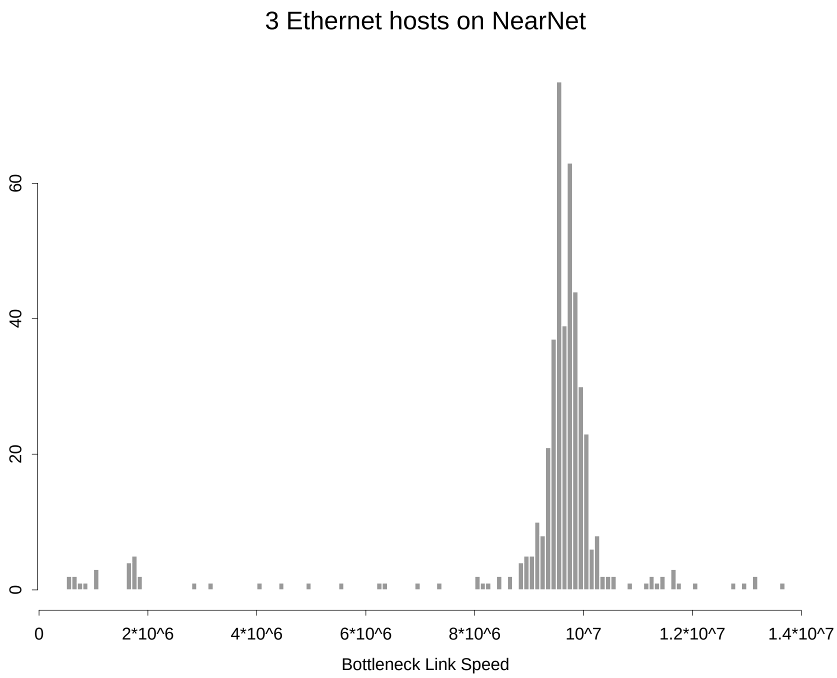

Figure 12: Histogram of BPROBE results: Nearnet Ethernet hosts

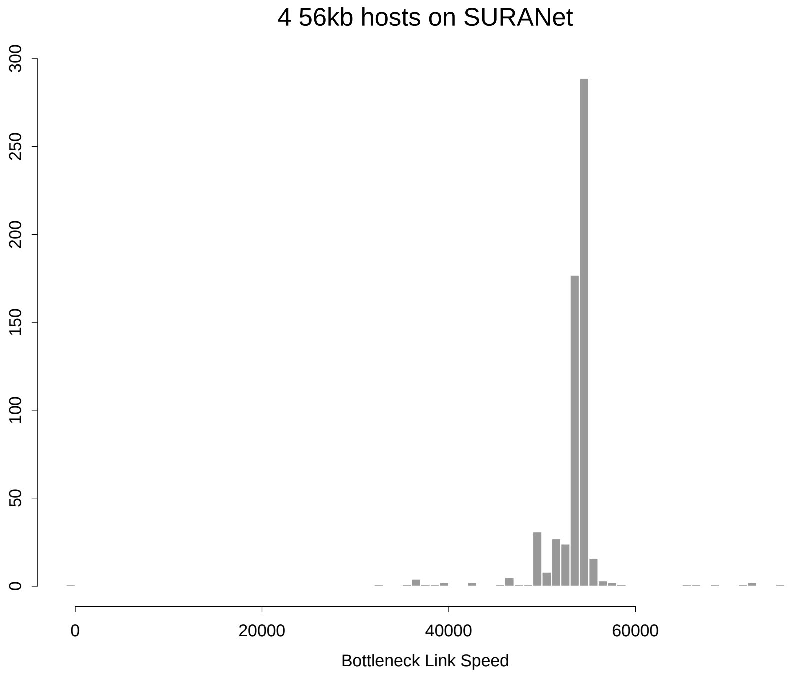

Figure 13: Histogram of BPROBE results: SURANET 56Kbps hosts

rated capacity, 52% of the estimates within 10% of the rated capacity and 75% of the estimates within 20% of rated capacity.

Finally, Figure 12 shows the histogram for regional hosts connected by Ethernet. The 397 probes of the 3 Ethernet hosts resulted in another tight grouping, with fractiles comparable to the results for the 56Kbps hosts. For the 10Mbps hosts we find: 68% of the estimates within 5% of the 10Mbps capacity, 86% of the estimates within 10% of the 10Mbps capacity and 92% of the estimates within 20% of the 10Mbps capacity.

Wide-area validation: The final validation test was done using a set of hosts on SURANET, a regional network in the Baltimore–Washington, D.C. metropolitan area. Hosts on this regional net are approximately 16 hops away from our test client including 7 MCI backbone hops. Once again, we were able to obtain capacity maps which let us independently verify the capacity of the bottleneck link to each of the selected hosts.

Figure 13 shows the histogram of bandwidth estimates for the 611 probes of the 4 56Kbps hosts. Once again, the histogram shows a tight grouping of estimates near the known bottleneck speed.

Figure 14 shows the histogram of bandwidth estimates for the 459 probes of the 3 T1 hosts. In this case, as in the regional case, we find minor peaks around the main peak at the expected value. We suggest an explanation in the Discussion section below.

Figure 15 shows the histogram of bandwidth estimates for the 475 probes of the 3 Ethernet hosts. Here we find a fair amount of serious underestimates, but the bulk of the data is still clustered around the known value.

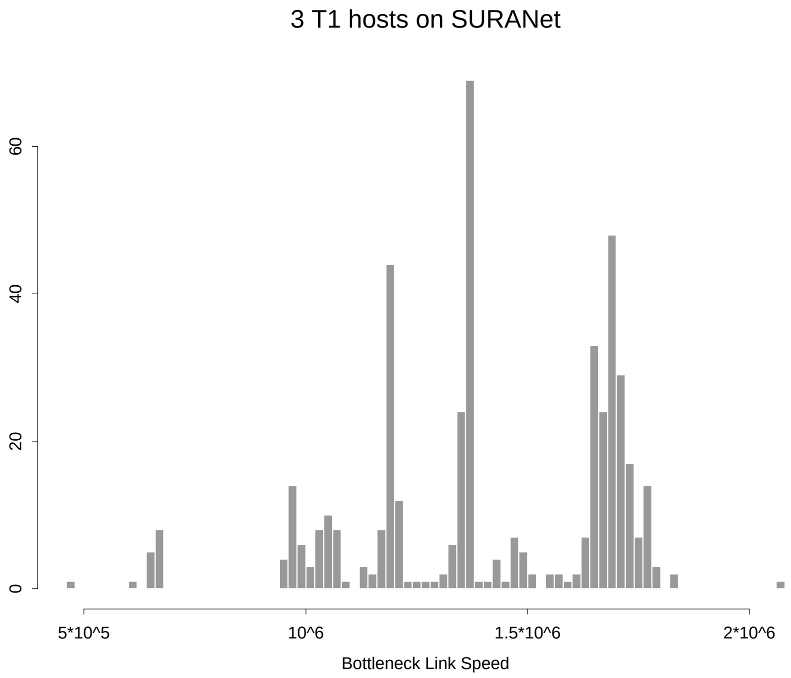

Figure 14: Histogram of BPROBE results: SURANET T1 hosts

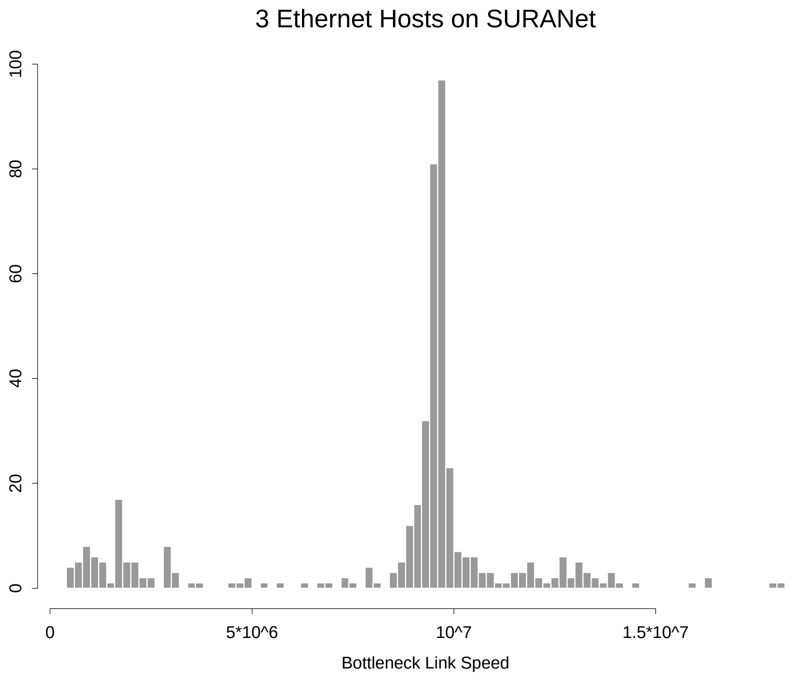

Figure 15: Histogram of BPROBE results: SURANET Ethernet nodes

Summary and Discussion Relative error values for the three sets of hosts are given in Table 1, which presents a more quantitative view of the validation results. The columns show percentages of error relative to the rated capacity of the bottleneck link; each row gives measurements for a particular subset of hosts. Each entry in the table gives the percentage of the bandwidth estimates that fall within a given relative percentage of the rated capacity. For example, 96% of the bandwidth estimates of local 56Kbps hosts were within ±10% of 56Kbps. As is evident from Figure 8 there is a systematic underestimation error due to unaccounted for link-level encapsulation overhead. In practice, this should not be a significant problem. For example, this could be addressed using a table of commonly available capacities to round the bandwidth estimates to the nearest likely value.

The results for T1 connections at the regional and wide-area levels were surprising. In contrast to the single large peaks found in the histograms for the other link capacities, for the two T1 host sets we find multiple peaks in the histograms (Figures 11 &14). An immediate suggestion was that the peaks correspond to individual hosts and represent underlying hardware differences. However, this is not the case. Bottleneck link speed estimates for any particular host were found in many of the peaks.

Our hypothesis is that consistent cross-traffic consisting of small packets of nearly equal size could cause this effect. Such traffic might be generated by telnet sessions, for example. This competing traffic would result in underestimation of bottleneck bandwidth as explained in section 2.1.3. If the source of the cross-traffic was regular enough, many of the resulting underestimates produced by BPROBE would agree and thus defeat our filtering process. The resulting estimate of bottleneck link speed for the host would then be an underestimate. Nevertheless, as measured by the fractile statistics, the estimates in these cases are still fairly accurate.

			Relative Error		
Network	Rated Capacity	No. Hosts	±5%	±10%	±20%
Local	56Kbps	7	55%	96%	99%
	10Mbps	9	80%	97%	100%
Regional	56Kbps	6	73%	87%	92%
	1.54Mbps	9	23%	52%	75%
	10Mbps	3	68%	86%	92%
Wide-area	56Kbps	4	84%	93%	97%
	1.54Mbps	3	12%	54%	82%
	10Mbps	3	31%	53%	63%

Table 1: Measurements of BPROBE accuracy for local, regional and wide-area host sets.

The results of these three sets of experiments convince us of the accuracy of the probe tool. For three very different classes of hosts we were able to accurately measure the base bandwidth of links using BPROBE. As we move from local to regional and then wide-area hosts, the tool shows only a minor loss in estimate accuracy.

Considering now our stated goals, we find that BPROBE does provide accurate base bandwidth estimates in spite of the difficult environment it measures; it operates without explicit cooperation from the network or servers; and it provides reliable estimates without excessive time overhead. At the current time, a reasonable concern is the probe's impact on the network. The current implementation consumes too much network bandwidth. Two approaches are under development: First, we believe we can achieve nearly the same accuracy and reliability with fewer packets; and second, it is anticipated that the base bandwidth estimate of a path will be cached, so that redundant measurements of a path will be unnecessary.

2.1.5 Survey of Bottlenecks to WWW Servers

As a further demonstration of the use of BPROBE we surveyed a set of 5825 randomly selected WWW servers to get an idea of the distribution of base bandwidth of connections to servers on the WWW. The servers were chosen uniformly from a list obtained from [7].

Previously, we performed a similar survey primarily concerned with the distribution of hops and latency to WWW Servers [4]. During the latest survey we gathered these statistics in addition to bandwidth estimates. For the set of 5825 servers, Figure 16 presents the distribution of hops (measured by *traceroute*) and Figure 17 gives the distribution of latencies (measured by *ping*). As discussed in [4] the striking difference between the two distributions has implications for replica placement and server selection. In particular, the lack of correlation between the two measures suggests that hops is a bad predictor of latency, and thus a bad metric for distance in an internetwork. What is needed is a dynamic measurement of current network conditions. This observation motivated us to design BPROBE and CPROBE.

Figure 18 shows the histogram of bottleneck link speed estimates from this survey of WWW servers. Notice the distinct peak at 10Mbps. There are no higher estimates because the survey was run from a machine on a 10Mbps local network. Since BPROBE measures the bottleneck link, readings higher than this should not be expected. Another concentration of estimates appears around 1 Mbps.

Inspecting the data, we find that of the 5825 servers surveyed, 2586 have bottleneck bandwidth estimates greater than 5Mbps. In other words, about 44% of the servers were reachable at Ethernet speeds. However, nearly 56% of the servers surveyed had a bottleneck link speed less than half of Ethernet speed and nearly 40% of them exhibit a bottleneck less than 20% of Ethernet speed.

In Figure 19 we restrict the histogram to estimates smaller than 200Kbps. This range includes servers with a low-speed modem as the bottleneck as well as other low-capacity connections. There are 659 (about 10%) of the servers with bottlenecks in this range. There is a clear peak at 56Kbps, a value which we have established as a common bottleneck speed in our validation experiments. Another peak appears near the ISDN rate of 128Kbps.

Figure 20 gives estimates in the range 200Kbps to 2Mbps, a region which includes T1 connections, fractional T1 and other values. This region includes 2351, or about 40% of the servers surveyed.

What are the implications of this distribution for replication and server selection? If we assume a uniform distribution of copies of popular documents over this set of servers, it becomes clear that measurement of the

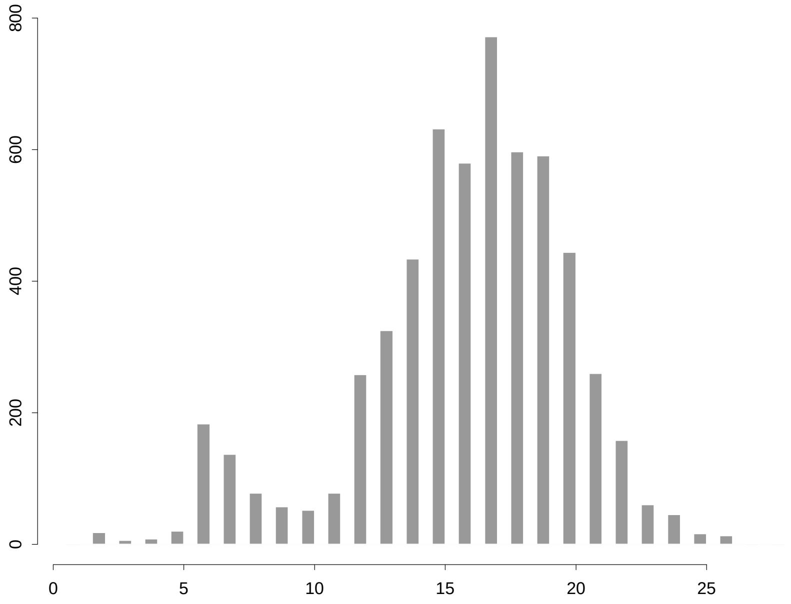

Figure 16: Histogram of hops to 5825 WWW Servers

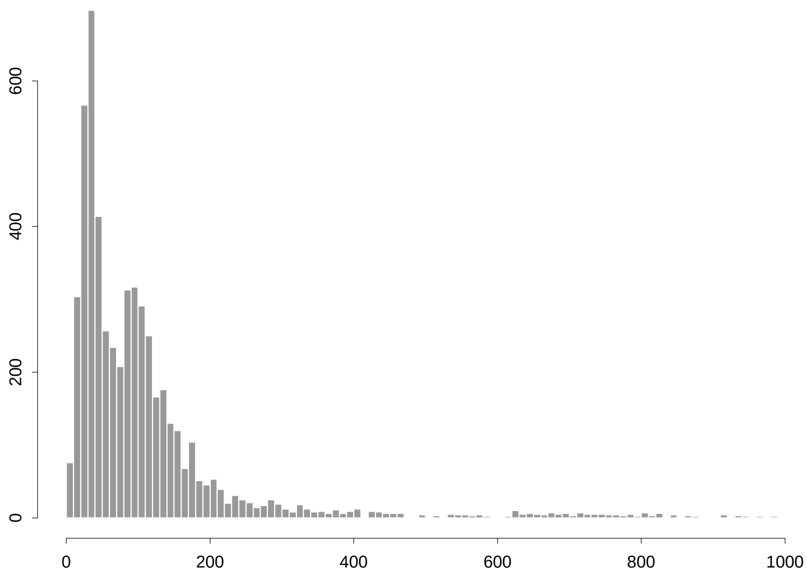

Figure 17: Histogram of latencies to 5825 WWW Servers

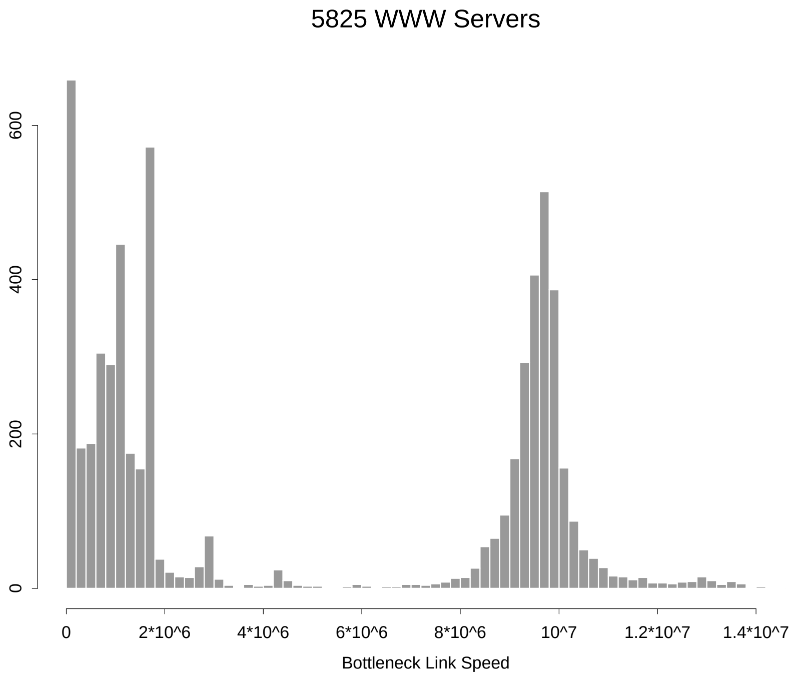

Figure 18: WWW survey results: histogram of bottleneck link speeds to 5825 WWW servers

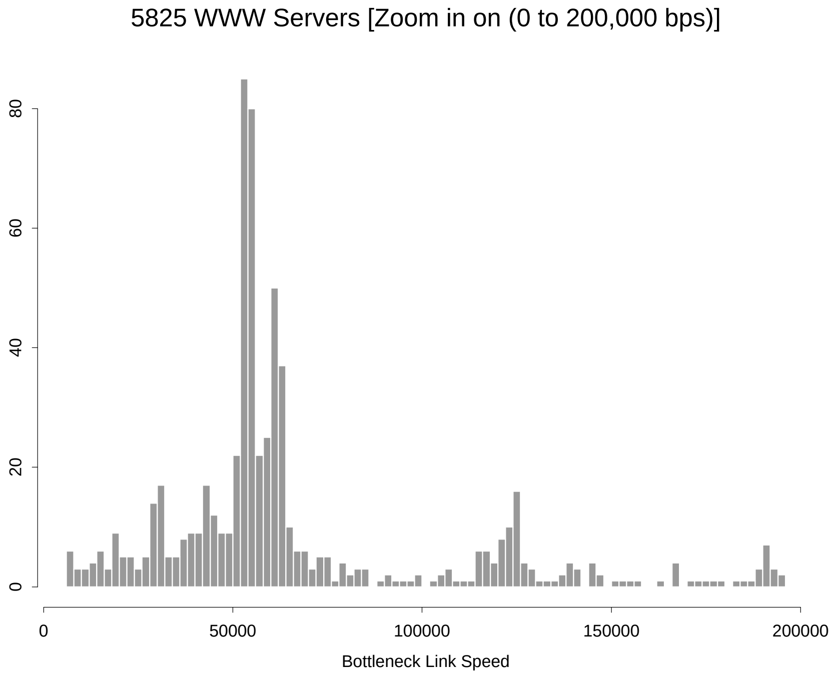

Figure 19: WWW survey results: subset (10%) of servers with bottleneck link speeds less than 200Kbps

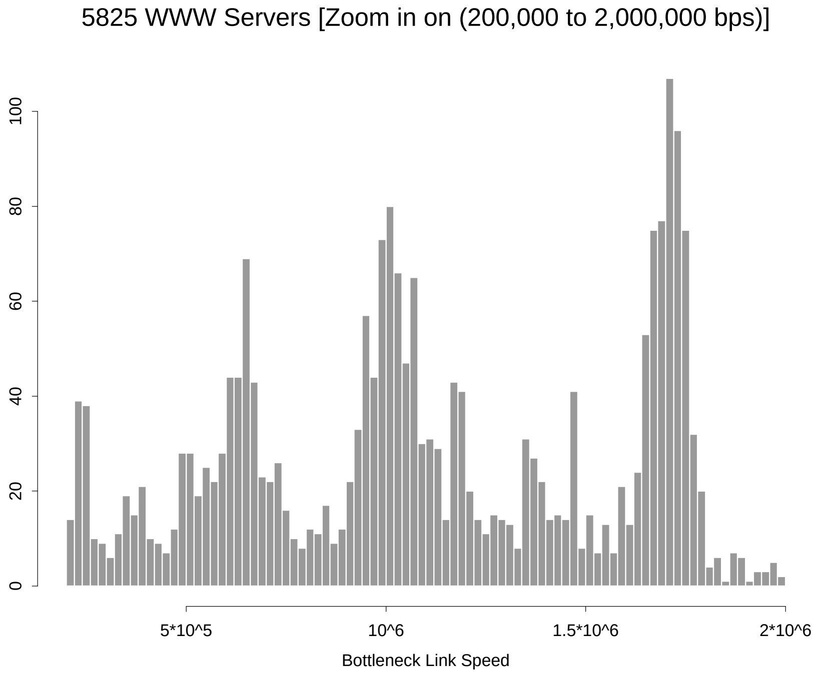

Figure 20: WWW survey results: subset (40%) of servers with bottleneck link speeds between 200Kbps and 2Mbps

bottleneck link speed is of prime importance when a choosing among replicas. For this distribution, choosing a server at random gives a better than even chance of getting a slow link. Clearly, a choice based on available bandwidth should give better performance. More precise evaluation of the amount of improvement to be gained by dynamic server selection policies are the focus of our current work.

2.2 Cprobe: Measuring Available Bandwidth

In the previous section we described our bandwidth probing tool which gives a fairly reliable estimate of the bottleneck link speed for a path between two hosts on the Internet. We now consider estimating available bandwidth.

To measure available bandwidth, we developed a tool called *cprobe*. Cprobe's technique is straightforward: by bouncing a short stream of echo packets off of the target server and recording the time between the receipt of the first packet and the receipt of the last packet, we can measure the presence of competing traffic on the bottleneck link. Dividing the number of bytes sent by this time yields a measure of available bandwidth, $\mathcal{B}_{avail}$, that represents the actual throughput achieved by the probe bytes. As long as we can send packets at a higher rate than the bottleneck link speed (which we can measure using bprobe) this effect should occur. Any additional time lag between the first packet and the last one represents delay due to intervening non-probe bytes (competing traffic).

The utilization of the bottleneck link can then be computed as the ratio of available bandwidth measured by

CPROBE to the bottleneck link speed as estimated by BPROBE:

$$\mathcal{U}_{probe} = \frac{\mathcal{B}_{avail}}{\mathcal{B}_{bls}}.$$

In practice, we discovered a complication during testing of CPROBE. Occasionally, the sending host would be momentarily delayed due to operating system effects, delaying the returning flow of packets. This resulted in an unusually long delay between one pair of packets which then caused an overestimate of competing traffic, since the delay was wrongly attributed entirely to traffic. In addition, scheduling effects on the receiving side can cause the inter-packet time to be unrealistically short. To eliminate these two kinds of erroneous readings from our data we discard the highest and lowest inter-arrival measurements when calculating $\mathcal{B}_{avail}$. We have found that this improves the accuracy of the measurements significantly. In order to tolerate packet drops and possible re-ordering of packets (which invalidate inter-arrival time measurements), we use the results of four separate 8-packet streams when calculating the available bandwidth.

As in the case of BPROBE, care must be taken to ensure that the CPROBE's results are valid. We validated the individual inter-arrival measurements using a packet tracing tool running on a local Ethernet. The experimental set-up consisted of the probe client and the probe target; a host running the packet trace tool; and other hosts between which FTP sessions were run to provide a background load. While varying the background load we ran several repetitions of the probe tool. We then compared the probe's measurements of packet inter-arrival times with the log of the packet traces. The probe's measurements of elapsed time for packet transfers were generally accurate to within $\pm 10\%$ relative to the packet trace measurements. We then compared CPROBE's estimate of available bandwidth with that derived from the packet trace log. Using the log, we calculated the time difference between the first and last reply, and divided by the amount of data sent, duplicating the calculation done by CPROBE. The measurements are quite accurate as can be seen from the fractile results in Table 2. Three quarters of the measurements are within 5% of the actual value and all are with 25% of the actual value. Of course, these results are limited to the local-area network where we can control the cross-traffic and measure all probe and non-probe packets. Beyond the local network validation becomes very difficult.

Relative error Fractile	Percentage of measurements within Fractile
5%	74%
10%	81%
15%	88%
20%	92%
25%	100%

Table 2: Fractile quantities for CPROBE's available bandwidth estimates.

Evaluating CPROBE against our design goals we find that it operates without relying on support from servers or the network; that it's impact on the network is not too great and that the measurement of available bandwidth is quite accurate. An open question is the predictive ability of the measurements of available bandwidth provided by CPROBE. Can the measurements be used to reliably predict the available bandwidth in the future? If so, how far into the future and with what degree of accuracy?

3 Future Directions and Conclusions

We have plans to extend this work in several directions.

Our primary interest is using BPROBE and CPROBE in addition to *ping* to gather current information about the state of the network and using this information as input to dynamic server selection algorithms. We plan to evaluate the contribution of each of these measurement techniques and combinations of them to determine which are of use in selecting a server dynamically.

Currently, we are building a version of the Mosaic WWW browser which uses the BPROBE tool to inform the user of the relative expected transfer speeds of links on a Web page. The hypertext anchor is color-coded with a measure of the current network state. For instance, faster expected transfers are coded green while slower ones are red.

Another browser extension we are considering is support for multiple URLs per link in a document. This list of alternate servers can be used as input to dynamic server selection by the browser. This represents a step along the way to integrated support for replication in which the browser would have to *obtain* a list of replicas. Such a mechanism might be particularly attractive to heavily loaded sites.

We also plan to package the probe tools as a daemon that can be placed strategically throughout a network to allow probing of conditions in remote parts of the network. This would also allow measurement of a link as a part of several paths providing confirmation of probe estimates.

In conclusion, we have introduced and validated two tools useful for measuring network conditions at the application level: BPROBE, which uses ECHO packets to measure the bottleneck link speed of paths between hosts in the Internet; and CPROBE, which measures the presence of competing traffic on the bottleneck link. We have presented validation results for each tool showing that accurate and reliable measurements of bottleneck link speed can be made under real network conditions. As an example application of our tools we presented a study in which we used BPROBE to measure the base bandwidth to a set of WWW servers. The results of this survey illustrate the potential for performance improvements based on dynamic measurement of current network conditions such as provided by BPROBE and CPROBE.

References

[1] Jean-Chrysostome Bolot. Characterizing End-to-End packet delay and loss in the Internet. *Journal of High Speed Networks*, 2(3):305–323, 1993.

[2] Jean-Chrysostome Bolot. End-to-End Packet Delay and Loss Behavior in the Internet. In *Proceedings of SIGCOMM 1993*, pages 289–298. ACM SIGCOMM, August 1993.

[3] Pittsburgh Supercomputer Center. About the PSC Treno Server. Available at `http://www.psc.edu/ pscnoc/treno_info.html.`, November 1995.

[4] Mark E Crovella and Robert L. Carter. Dynamic server selection in the internet. In *Third IEEE Workshop on the Architecture and Implementation of High Performance Computer Systems'95*, pages 158–162, Mystic, Connecticut, August 1995.

[5] Van Jacobson. Congestion Avoidance and Control. In *Proceedings SIGCOMM '88 Symposium on Communitations Architectures and Protocols*, pages 314–329, Stanford, CA, August 1988.

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

[7] net.Genesis Corporation. Comprehensive list of sites. Available at `http://www.netgen.com/cgi/comprehensive.`, April 1995.