

Accuracy of Link Capacity Estimates using Passive and Active Approaches with CapProbe

Rohit Kapoor, Ling-Jyh Chen, M. Y. Sanadidi, Mario Gerla
UCLA Computer Science Department, Los Angeles, CA 90095, USA
{rohitk, cclljj, medy, gerla}@cs.ucla.edu

Abstract

CapProbe is an inexpensive and accurate means to estimate capacity. CapProbe combines both dispersion and end-to-end delay to estimate the capacity of the narrowest link on a path. We evaluate in this paper the accuracy of CapProbe estimation, and its dependence on end systems speed, packet sizes, narrow link speeds, and other system parameters. We test kernel and user level implementations of CapProbe and find the kernel implementation to be much more accurate. We also evaluate through experiments the effect of probing packet size on the accuracy of CapProbe estimation. Finally, we explore the idea of a “passive CapProbe” within the context of a TCP flow. Passive here means that the dispersion and delay observed for the TCP flow data and ACK packets, without introducing any additional probing packets. We test active and passive versions of CapProbe with TCP. The active version is found to produce more accurate capacity estimates than the passive version.

1. Introduction

Estimating the capacity of an Internet path in an end-to-end manner is a fundamental problem. Considerable efforts have been devoted to this problem over the last decade or so. Capacity of a path refers to the minimum physical link capacity¹ among all links on the path.

Knowledge of the capacity of a path can be put to good use in many applications. Multimedia servers would be able to determine appropriate streaming rates, while ISPs would be able to determine the capacity of their installed links, no matter how remote they may be from an operation center. Routing protocols and multicast application level protocols can use capacity information to build optimal routes/trees. Congestion control is another obvious

¹ Capacity is different from available bandwidth, which is the minimum unused bandwidth among the links on a path. While the capacity is a fixed quantity for a path, the available bandwidth is time varying.

area of application for accurate, timely, and computationally inexpensive capacity estimation methods. TCP Westwood [13], in its version called TCPW ABSE (Adaptive Bandwidth Share Estimate) [14], computes a connection “Eligible Rate Estimate”, or ERE, from the dispersion of ACKs (acknowledgements) received at the sender. TCPW also estimates the current congestion level in a manner similar to TCP Vegas. TCPW ABSE adjusts its ERE based on its estimate of the current congestion level. The ERE is intended to range from the recently achieved connection throughput, up to the bottleneck link capacity. When the congestion level is high, packet losses are assumed to be due to congestion, and the ERE the scheme produced is more conservative, closer to the recently achieved throughput of the connection. On the other hand, when the congestion level is estimated to be low, a packet loss is presumed to be due to random error, and the ERE produced is more aggressive, potentially reaching up to the bottleneck link capacity, but ERE can indeed exceed such capacity. Knowledge of the accurate capacity of the bottleneck link can be helpful in fixing such approximations in TCPW. Further, entirely different methods from the current TCPW strategies may in the future be feasible when an accurate capacity estimate is available.

CapProbe, a simple and accurate capacity estimation scheme, was presented in [6]. It sends packet pair probes and combines their dispersion and delay measurements to estimate capacity. Its estimation has been shown to be quick and accurate in almost all scenarios. Other capacity estimation schemes, such as bprobe [2], pathchar [5], pathrate [4] and others have been presented earlier, but they suffer from being either slow or inaccurate. In all cases, the previous schemes rely on statistical methods requiring the gathering of histograms of data, and processing such large amount of data after it has been gathered. This results in slow and computationally expensive methods. Unfortunately, at the end, the capacity estimates they produce may still be significantly inaccurate under certain conditions.

The novelty in CapProbe is that it combines the use of both dispersion measurements with end-to-end delay

measurements in an inexpensive scheme that produces accurate capacity estimates. In summary, a packet pair that is launched into the network back to back, is dispersed at the bottleneck link according to the bottleneck capacity. The smaller the capacity, the larger the dispersion. If such dispersion would arrive at a destination unperturbed, the destination can use the dispersion to identify the bottleneck capacity. Of course the dispersion can easily be perturbed by cross traffic on the path links. The dispersion perturbation is the result of queuing by either the first or second packets of the probing pair. CapProbe method eliminates the queuing perturbation by looking for the packet pair with the minimum end-to-end delay. The dispersion of such pair is likely to be an accurate reflection of the bottleneck link capacity. The accuracy of the results however depends on many factors including the path length, the variations of the capacity of the path links, the cross traffic rates, packet sizes of the probes and of those of the cross traffic, time resolutions at the senders and receivers, the mechanism used to launch the packet pair into the network and collect it from the network.

In this work, we study the dependence of the accuracy of CapProbe estimation on accuracy of operating system measurement. The larger the probing packet size used by CapProbe, the lower is the resolution required by the operating system to measure accurately. We show that the use of larger packet sizes can enable CapProbe estimations to be more accurate, particularly when the capacity of the narrow link on the path is large. We also show that a kernel mode implementation of CapProbe can be much more accurate than a user level implementation. We then evaluate the use of CapProbe TCPW. We compare two versions of CapProbe for use with Westwood, an active version which sends probing packets and a passive version, which uses TCP data and ACK packets for estimation.

The remainder of this paper includes Section 2 which provides a recap of CapProbe, and presents related work on capacity estimation; Section 3 in which we study the dependence of the accuracy of CapProbe on operating system measurement accuracy; Section 4 which presents results comparing active and passive versions of CapProbe, when used with TCP Westwood; and Section 5 which concludes the work.

2. Capacity Estimation

2.1. CapProbe

In [6], a new technique for estimating narrow link capacity, CapProbe, was presented. CapProbe is based on a simple and fundamental observation: *A packet pair measurement corresponding to either an over-estimated or an under-estimated capacity suffers cross-traffic induced queuing at some link.* Exploiting this observation, CapProbe combines dispersion and delay measures to filter out packet pair samples “distorted” by cross-traffic. The two kinds of distortion are “expansion” of disper-

sion, leading to under-estimation and “compression” of dispersion, leading to over-estimation.

Compression of dispersion occurs when the narrow link is not the last one on the path, i.e., when so-called *post narrow* links are present. The presence of these links can reduce the packet pair dispersion created by the narrow link if the first packet of the pair queues at a post-narrow link, while the second packet experiences queuing for a shorter time than the first packet, as shown in Fig 1. This causes the dispersion between the packet pair departing from the post-narrow link to be smaller than that created by the narrow link, leading to an over-estimation of capacity. Note that the queuing of the first packet in this case is caused by interference from cross-traffic. This behavior is more pronounced when the probe packets are smaller than cross-traffic packets. In this case, the *first packet* of the packet pair will have queued at a post-narrow link *due to interference from cross-traffic*.

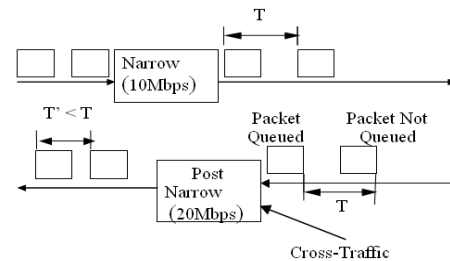


Figure 1: Over-estimation of Capacity

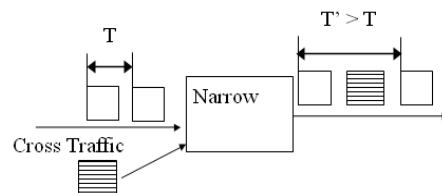


Figure 2: Under-estimation of Capacity

Expansion of dispersion occurs when cross-traffic packets are served (transmitted) in between packets of the packet pair. This increases the dispersion of the packet pair and leads to a lower capacity estimation. Fig 2 shows how under-estimation of capacity can occur. When dispersion under-estimates capacity, the *second packet* must have queued *due to interference from cross-traffic*.

Thus, whenever packet dispersion under-estimates or over-estimates capacity, at least one packet of the packet pair suffers cross-traffic induced queuing. This means that whenever an incorrect value of capacity is estimated, the sum of the delays of the packet pair packets, called the *delay sum*, includes cross-traffic induced queuing delay. Such delay sum is clearly larger than the delay sum of a packet pair sample in which both the packets do not suffer cross-traffic induced queuing. This delay sum, which does not include any cross-traffic queuing delay, is referred to as the *minimum delay sum*. The dispersion of such a packet pair sample is not distorted by cross-traffic and will measure the correct capacity. This sample can

easily be identified since its delay sum will be the minimum among delay sums of all packet pair samples.

CapProbe works by sending packet pair samples and calculating the delay sum for all samples. The dispersion measured from the sample corresponding to the minimum of all these delay sums reflects the narrow link capacity.

CapProbe is based on the assumption that at least one packet pair sample with the *minimum delay sum* is obtained. In a network such as the Internet in which the traffic intensity varies due to reactive TCP flows, there is very high likelihood of obtaining one or more of the desired samples. Results presented for CapProbe [6] have shown that it can estimate accurate capacity in almost all scenarios typically in a few samples.

2.2. Previous Work on Capacity Estimation

Previous work on estimating the capacity of a path has mostly been based on using dispersions of packet pairs or trains. In [2], Carter and Crovella proposed *bprobe*, in which filtering methods are applied on packet pair measurements, consisting of different packet sizes. In [7], Lai used packet pair measurements, but filtered them through a kernel density estimator.

Paxson in [10] identified multi-channel links as a failure case of packet pairs and presented the Packet Bunch Modes (PBM) technique to overcome this limitation. The PBM methodology consists of sending packet trains of different lengths in response to a distribution with multiple modes, treating multiple modes as corresponding to multi-channel links. Dovrolis et al [3] elaborated further on the occurrence of multiple modes. They showed that the strongest mode in the multimodal distribution may correspond to the capacity, or to an under- or over-estimate of the capacity. They presented a capacity estimation methodology, which first sends packet pairs. If this yields a multimodal distribution, then probing with packet trains with an increasing value of N is initiated. For some value of N , the distribution becomes unimodal, and the capacity is selected as the next highest mode after this mode in the multimodal distribution that was obtained from packet pairs.

A different technique, not based on dispersion of packet pairs, but rather on the variation of the round-trip delay as the packet size increases, was used by Jacobson in pathchar [5]. This technique, based on the generation of ICMP replies from routers, is known to have scalability problems. The overhead associated with this method can be quite high since it tries to estimate the capacity of every link on the path in order to estimate the path's capacity. Packet tailgating is another technique proposed by Lai [8]. This technique is divided into two phases: the Sigma phase, which measures the characteristics of the entire path, and the Tailgating phase, which measures the characteristics of each link individually. However, the reported measurements are still often inaccurate, possibly due to accumulation of errors in capacity estimates as measurements proceed along the path.

3. Experiments

In this section, we evaluate through measurements experiments the dependence of the accuracy of CapProbe estimation on end systems time resolutions, and thus their measurement accuracy. We also study the effect of using different probing packet sizes on accuracy of CapProbe estimation. We also show that a kernel mode implementation of CapProbe is much more accurate than a user level implementation.

3.1. User Mode and Kernel Mode

The experiment set-up consisted of a laptop at UCLA as the source of the CapProbe probing packets, connected to the UCLA campus backbone through a 100 Mbps connection. The experiments were conducted in three different environments: the Internet, and the Internet2 Abilene network [1]. Abilene is a high-bandwidth network connecting various universities and research labs at speeds up to 10 Gbps. Three different destinations were chosen in each environment: from the Internet (YAHOO: www.yahoo.com, NTNU: www.ice.ntnu.edu.tw, and WLSH: www.wlsh.tyc.edu.tw), and the Abilene network (MIT: www.mit.edu, CMU: www.cmu.edu, and UCONN: www.uconn.edu). Our knowledge of the capacity of the links along each path was derived either from the public Internet resources or from querying system administrators at different destinations.

Two machines of different CPU speeds are used in the experiments. The first system was equipped with Pentium II 500 MHz CPU and 512 MB RAM, and the second with Pentium IV 1.8 GHz CPU and 512 MB RAM.

The user level implementation of CapProbe is based on the *ping* [11] utility, which is included in the *IPUtils* open source software. We modified the *ping* program to send a packet pair instead of a single ping packet each time. From the ICMP replies, the capacity was calculated using the CapProbe technique. Since this implementation is in user mode, operating system overheads are introduced which can introduce inaccuracies in measuring packet pairs dispersion and delays. In order to understand the effect of end system speed on CapProbe estimation results, we performed similar user mode experiments on the slower (User mode 1, Pentium II 500 MHz CPU) as well as the faster machine (User mode 2, Pentium III 1.8 GHz CPU).

The kernel mode CapProbe is implemented in the Linux kernel, such that ICMP packets and replies were sent and received by a program running in the kernel. In the kernel mode, delay and dispersion measurements are less susceptible to operating system induced inaccuracies. The kernel mode is, as expected, more accurate. For experiments using the kernel mode implementation, we used only the slower machine (Pentium II 500 MHz CPU) as the CapProbe source.

Tables 1 and 2 show capacity estimation results when using CapProbe in the Internet and Abilene environments respectively with different probe packet sizes. Each run

consisted of 600 packet pair probes, with 2 seconds between successive probes. Results for 3 different runs of each experiment are reported.

From the measurement results, it is clear that CapProbe is very sensitive to the CPU speed of the measurement machine when operating in the user mode. When running on the slower machine (User mode 1), CapProbe has difficulties in estimating the correct capacities in many of the cases, except the WLSH path with its small capacity of 1.5 Mbps. The faster machine (User mode 2), on the other hand, is able to estimate the capacity correctly as long as probing packets are 3000 bytes or larger. The kernel mode CapProbe estimates accurately in most cases, except when the probe packet size is smaller, e.g. 500 bytes.

	PKSize (bytes)	YAHOO			NTNU			WLSH		
		1	2	3	1	2	3	1	2	3
User mode 1 (slow machine)	500	571.4	285.7	452.8	133.3	571.4	67.8	1.45	1.45	1.41
	1000	421.1	61.5	160.0	1000	381.0	216.2	1.51	1.47	1.5
	3000	338.0	237.6	452.8	79.2	827.6	444.4	1.47	1.48	1.47
	5000	231.2	156.2	239.5	209.4	210.5	243.9	1.48	1.47	1.47
User mode 2 (fast machine)	500	64.5	148.1	400	54.1	56.3	40.8	1.26	1.48	1.28
	1000	156.9	156.9	150.9	160	82.5	95.2	1.45	2.17	1.50
	3000	103.4	112.1	117.0	111.6	123.7	93.1	1.47	1.49	1.46
	5000	99.5	100.8	102.6	89.9	102.3	107.8	1.47	1.47	1.48
Kernel mode	500	9.2	102.6	10.2	89.4	90.9	74.0	1.39	1.42	1.5
	1000	69.6	79.3	78.6	93.0	85.1	91.6	1.48	1.44	1.47
	3000	92.0	86.3	90.0	96.2	98.4	95.6	1.45	1.46	1.45
	5000	95.0	86.8	91.3	93.0	96.4	96.4	1.45	1.47	1.46
Narrow Link Capacity		100			100			1.5		

Table 1: Capacity (Mbps) estimates under User mode and Kernel mode on Internet with different probing packet sizes

	PKSize (bytes)	MIT			CMU			UCONN		
		1	2	3	1	2	3	1	2	3
User mode 1 (slow machine)	500	1000	1000	444.4	117.6	666.7	800	16.7	75.5	56.3
	1000	363.6	400.0	666.7	1143	421.1	296.3	421.1	800	347.8
	3000	413.8	545.5	489.8	307.7	381.0	289.2	342.9	363.6	358.2
	5000	227.3	245.4	205.1	194.2	248.4	197.0	66.0	190.5	68.7
User mode 2 (fast machine)	500	444.4	333.3	400.0	444.4	444.4	333.3	129.0	166.7	71.4
	1000	166.7	163.3	173.9	156.9	153.8	163.3	145.5	114.3	156.9
	3000	119.4	118.2	111.1	117.6	119.4	113.7	114.3	114.3	121.8
	5000	108.7	107.2	105.3	109.0	108.1	106.4	101.3	85.4	88.0
Kernel mode	500	42.6	65.6	63.5	87.0	74.0	64.5	69.0	27.0	27.0
	1000	86.0	95.2	90.9	90.9	98.8	79.2	98.8	74.8	80.8
	3000	97.6	94.5	100.4	94.9	98.8	102.1	94.9	91.3	88.9
	5000	91.1	95.5	92.8	96.2	97.1	98.3	96.2	94.8	97.1
Narrow Link Capacity		100			100			100		

Table 2: User vs. kernel mode capacity estimates on Internet 2

We conclude that the accuracy of CapProbe estimation is highly dependent on the end system time resolution capability. Suppose the capacity of the narrow link is C (Mbps) and the probing packet size is P (bits), the dispersion time (and also the clock granularity needed for accurate estimation) that needs to be measured is $T=P/C$ (microseconds). Table 4 shows required clock granularity needed for different probing packet sizes and narrow link capacities.

For the widely deployed fast Ethernet network (100Mbps), either a high time resolution machine or a large probing packet size is necessary for accurate capacity estimation. For example, when the probing packet size is 1000 bytes, the measurement will not be accurate unless the machine supports a time resolution of smaller than 0.08ms. However, such fine resolution may not be

possible in the user mode, since the operating system introduces overheads due to scheduling and context switching. On the other hand, using a large packet size can increase the chances of cross traffic interference, causing expansion of dispersion. The kernel mode implementation, thus, provides a good solution for providing the fine time resolution required for accurate estimation.

Packet Size	Narrow Link Capacity		
	100 Mbps	10 Mbps	1 Mbps
500 bytes	0.04 ms	0.4 ms	4 ms
1000 bytes	0.08 ms	0.8 ms	8 ms
3000 bytes	0.24 ms	2.4 ms	24 ms
5000 bytes	0.40 ms	4 ms	40 ms

Table 4: Required time resolution for accurate estimation

4. Use of CapProbe in TCP

In this section, we evaluate the use of two different versions of CapProbe, an active and a passive version, with TCP. We investigate whether it is necessary to send packet pair probes as ICMP packets separately from the TCP data packets. Of course, it would be advantageous if it were possible to use the data and ACK stream and their dispersion to estimate the paths capacities. Such an approach would render TCP independent of other ICMP issues, such as whether a destination system rejects ICMP packets when busy, or for security purposes. Unfortunately, as we show below, using a “vanilla” TCP data and ACK stream may not be suitable for gathering dispersion and delay data to estimate capacity. Instead, it appears that it requires some, perhaps minor, modifications to the TCP sending protocol, to develop a “passive” capacity estimation scheme that does not require sending any separate probing packets. Below we carry out our experiments within the context of TCP Westwood, as we are quite familiar with its implementation details.

4.1. Active and Passive Versions of CapProbe

The active version of CapProbe is the same as that defined in Section 2 which sends PING packet pairs to estimate capacity. We use TCP Westwood, to evaluate a “passive” version of CapProbe which does not send its own probes, but uses TCP data and ACK packets instead. Unlike the active version, the passive version consumes no extra bandwidth to estimate capacity, and does not rely on whether hosts accept or reject pings.

For CapProbe to accurately estimate capacity passively in TCP, data and acknowledgement packets of TCP should satisfy the following two conditions:

- A. Some TCP data packets should be sent back-to-back as a packet pair. As we show later in the experiments section, we found that approximately 15-20% of TCP data packets are sent back-to-back in a typical TCP connection, leading to the generation of approximately 7-10% packet pair samples (two back-to-back packets gives one packet pair sample).
- B. A TCP acknowledgement packet should be sent immediately on receipt of a data packet. TCP receivers that send “Delayed Acknowledgements” would not satisfy such

conditions. If the above two conditions are satisfied, TCP data packets sent back-to-back and the acknowledgements sent in response to these form packet pairs. The passive version of CapProbe can then estimate capacity using these samples.

We used NS-2 [9] as the simulation environment. The network topology used in our simulations consists of a six-hop path with capacities {10, 7.5, 5.5, 4, 6 and 8} Mbps. The narrow link on the path is 4Mbps. A TCP (TCP Westwood) connection, whose source is at the 10Mbps link, and destination is at the 8Mbps link, are used. Delayed acknowledgments are disabled at the destination. The TCP receiver in our experiments does not send delayed acknowledgements.

For the active version of CapProbe, the packet pair probes were sent at intervals of 100 ms from the source to the destination. The packet size for the packet pair is 500 bytes, the same as that used for the TCP connection. The simulation time was 30 sec.

The different traffic types we used for the cross-traffic were TCP, UDP and Pareto. The Pareto traffic consisted of 16 Pareto sources with parameter alpha = 1.9 (this is known to result in Long Range Dependent traffic [12]). In each set of experiments, we increased the rate of the cross-traffic from 1Mbps to 4Mbps, which is the capacity of the narrow link. The size of the cross-traffic packet was also varied to show different effects on the dispersion. Smaller cross-traffic packets are likely to cause more expansion of dispersion, while larger cross-traffic packets are likely to cause more compression of dispersion. The sizes of cross-traffic packets we experimented with were 200 bytes (smaller than probing packet size) and 1000 bytes (larger than probing packet size). Cross-traffic links were used to limit the amount of cross-traffic on our test path.

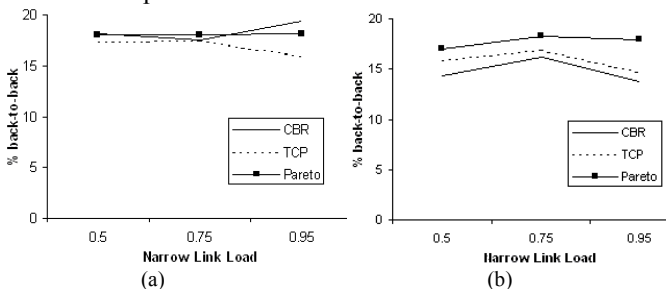


Figure 3: Fraction of TCP data packets sent back-to-back when cross-traffic packet size is (a) 200bytes (b) 1000 bytes

The metrics we studied were the accuracy of capacity estimation and the number of samples needed and time consumed for accurate estimation. For the passive version of CapProbe, it is also important to assess the fraction of TCP packets that are useful as packet pair samples. Such fraction of back-to-back packets vary according to the cross traffic characteristics, including their arrival statistics and their intensity, and the relative sizes of cross traffic and the TCP traffic.

Fig 3 (a) and (b) show the fraction of packets sent back-to-back by TCP. Such packets can be used as packet pair samples. The number of packet pair samples

is half of the number of such back-to-back packets. Fig 3 (a) shows the fraction of TCP packets that are back-to-back, for different types and intensities of cross-traffic, when cross-traffic packet size is 200 bytes. Fig 3 (b) shows the same when cross-traffic packet size is 1000 bytes.

From Figs 3 (a) and (b), approximately 15-20% of packets are sent back-to-back, forming around 7-10% packet pairs. We observed that most of the back-to-back packets were sent in the Slow Start phase of TCP. Very few TCP packets were sent back-to-back in the Congestion Avoidance phase.

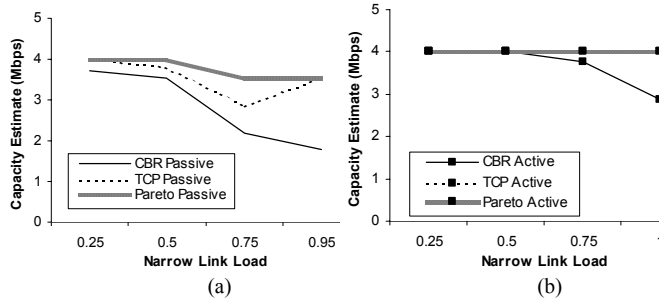


Figure 4: Capacity estimated by (a) passive and (b) active versions of CapProbe; cross-traffic packet size is 200 bytes

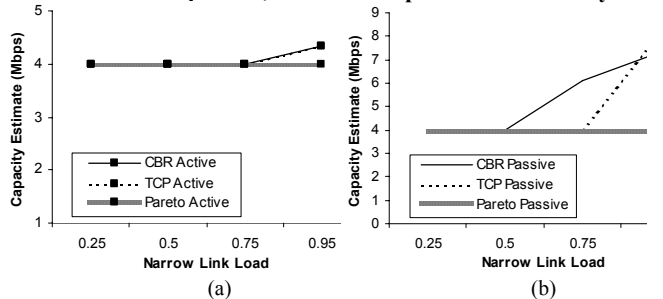


Figure 5: Capacity estimated by (a) passive and (b) active versions of CapProbe; cross-traffic packet size is 1000 bytes

Fig 4 (a) and (b) show the capacity estimated by the passive and active versions of CapProbe respectively for different kinds of cross-traffic (CBR, TCP, Pareto), and different narrow link loads, when cross-traffic packet size is 200 bytes. Fig 5 (a) and (b) shows the capacity estimated by the passive and active versions of CapProbe, respectively, when cross-traffic packets are 1000 bytes.

The following are the observations from the graphs:

- A. The active version of CapProbe is clearly more accurate than the passive version. The only case where it measures inaccurate capacity is when cross-traffic is CBR and intensive (> 70 % load). In fact, when cross-traffic is TCP or Pareto (i.e., long range dependent), active CapProbe always measures accurately.
- B. The passive version of CapProbe measures accurately till about 50% load on the narrow link, while its estimates are inaccurate for higher loads. The main reason behind the inaccuracy of the passive version is that it uses the TCP connection itself as the probing traffic. This leads to a lot of "internal" (as opposed to cross-traffic) interference from packets of the TCP connection. Since the typical behavior of TCP alternates between sending a bulk of packets and not sending much, the probe packets (which are the actual TCP packets here) get to "see" the network only at times

when TCP is sending bulk data. This bulk data offers interference and causes queuing of the back-to-back sent probe packets. Typically, very few probe packets get to see the network when TCP is not sending much.

C. The passive version measures reasonably accurately when cross-traffic consists of Pareto flows. More experimentation on the Internet is required to determine accuracy of the passive version in “real” traffic environments.

D. When cross-traffic packet size is 200 bytes, the inaccuracy is always an under-estimation of capacity. This is a result of the cross-traffic packet size being smaller than the size of probing packets. Smaller packets are more likely to be served between a probing packet pair, causing queuing of the second packet, expansion of the pair dispersion, leading to under-estimation. On the other hand, when cross-traffic packet size is 1000 byte, the estimation error is most often an over-estimation, as smaller probe packets are more susceptible to dispersion compression [9].

Table 5 (a) and (b) show the number of samples required by the passive and active versions of CapProbe to estimate within 10% of the narrow link capacity. The -1 values correspond to the estimated capacity not being within 10% of the narrow link capacity. The passive version typically needs a larger number of samples than the active version, yet the difference is not very large.

Table 6 shows the time (in seconds) taken by the passive version of CapProbe to estimate within 10% of the narrow link capacity. Typically, the time taken is small enough to merit considering the use of the passive version in long-lived TCP flows. We do not show the time taken by the active version since this is dependent on the sending rate of its probes and is clearly proportional to the number of samples required (shown in Table 5 (b)).

	Narrow Link Load	CBR, 200	TCP, 200	Pareto, 200	CBR, 1000	TCP, 1000	Pareto, 1000
(a)	0.5	78	136	71	162	215	106
	0.75	-1	-1	136	-1	68	166
	1	-1	143	122	-1	-1	166

	Narrow Link Load	CBR, 200	TCP, 200	Pareto, 200	CBR, 1000	TCP, 1000	Pareto, 1000
(b)	0.5	100	15	8	34	72	12
	0.75	69	123	35	86	45	34
	1	-1	109	126	186	198	144

Table 5: Number of samples required by (a) passive and (b) active CapProbe to estimate within 10% of actual capacity

Narrow Link Load	CBR, 200	TCP, 200	Pareto, 200	CBR, 1000	TCP, 1000	Pareto, 1000
0.5	2.354592	4.338308	3.475212	7.893789	7.951189	4.644695
0.75	-1	-1	7.159452	-1	2.039889	5.066789
1	-1	4.99854	6.526776	-1	-1	5.128721

Table 6: Time (in seconds) to capacity estimate within 10% of actual capacity

5. Conclusions

In this paper, we summarized the basic idea of CapProbe, an accurate and inexpensive capacity estimation

technique. We studied the dependence of CapProbe accuracy on end-system speed, implementation within a system, probe and cross traffic packet lengths, link speeds, etc. We found that a kernel mode implementation of CapProbe is much more accurate than a user mode implementation, as it is not subject to operating system overheads. For the user mode implementation, large probing packet sizes (such as 3000 bytes) are required to measure narrow link capacity of the order of 100 Mbps accurately.

We also studied the effectiveness of a passive version of CapProbe, where data and ACK packets of a TCP flow are relied upon with no additional probing packets sent into the path. We tested two versions of CapProbe, an active and a passive version. Results showed that the active version measures capacity more accurately than the passive version. In the future, we will investigate whether a slightly modified sending protocol for TCP would allow the development of a more accurate passive CapProbe method. More experimentation is also needed to evaluate the impact of the accurate CapProbe on many protocols, including TCP Westwood which currently can obtain only rough estimates of a path capacity.

6. References

- [1] Abilene Backbone Network, <http://abilene.internet2.edu/>
- [2] R. Carter and M. Crovella, "Measuring Bottleneck Link Speed in Packet-Switched Networks", Performance Evaluation, Vol. 27-8, pp. 297-318, Oct. 1996
- [3] C. Dovrolis, P. Ramanathan, and D. Moore, "Packet dispersion techniques and bandwidth estimation," http://www.cc.gatech.edu/fac/Constantinos.Dovrolis/Papers/ton_dispersion.pdf
- [4] C. Dovrolis, P. Ramanathan, and D. Moore, "What do packet dispersion techniques measure?" in Proceedings of IEEE Infocom'01, Anchorage, Alaska, April 22--26 2001.
- [5] V. Jacobson, "Pathchar: A tool to infer characteristics of Internet paths", <ftp://ftp.ee.lbl.gov/pathchar/>
- [6] R. Kapoor, L.-J. Chen, M. Y. Sanadidi, M. Gerla, "CapProbe: A Simple and Accurate Technique to Measure Path Capacity," Technical Report TR040001, UCLA CSD, 2004.
- [7] K. Lai, and M. Baker, "Measuring Bandwidth", In Proceedings of IEEE INFOCOM '99, March 1999.
- [8] K. Lai, and M. Baker, "Measuring Link Bandwidth Using a Deterministic Model of Packet Delay", SIGCOMM 2000
- [9] Network Simulator (NS-2), www.mash.cs.berkeley.edu/ns
- [10] V. Paxson, "Measurements and Dynamics of End-to-End Internet Dynamics", Ph.D. thesis, Computer Science Division, Univ. Calif. Berkeley, April 1997
- [11] Ping, <http://ftp.arl.mil/~mike/ping.html>
- [12] M.S. Taqqu, W. Willinger, R. Sherman, "Proof of a fundamental result in self-similar traffic modeling," SIGCOMM Computer Communications Review, 27: 5-23, 1997
- [13] TCP Westwood, <http://www.cs.ucla.edu/NRL/hpi/tcpw>
- [14] R. Wang, M. Valla, M. Y. Sanadidi, and M. Gerla, "Adaptive Bandwidth Share Estimation in TCP Westwood", In Proc. IEEE Globecom 2002, Taipei, Taiwan