CapProbe: A Simple and Accurate Capacity Estimation Technique

Rohit Kapoor	Ling-Jyh Chen	Li Lao	Mario Gerla	M. Y. Sanadidi
Qualcomm	UCLA	UCLA	UCLA	UCLA
rohitk@cs.ucla.edu	cclljj@cs.ucla.edu	llao@cs.ucla.edu	gerla@cs.ucla.edu	medy@cs.ucla.edu

ABSTRACT

We present a new capacity estimation technique, called CapProbe. CapProbe combines delay as well as dispersion measurements of packet pairs to filter out samples distorted by cross-traffic. CapProbe algorithms include convergence tests and convergence speed-up techniques by varying probing parameters. Our study of CapProbe includes a probability analysis to determine the time it takes CapProbe to converge on the average. Through simulations and measurements, we found CapProbe to be quick and accurate across a wide range of traffic scenarios. We also compared CapProbe with two previous well-known techniques, pathchar and pathrate. We found CapProbe to be much more accurate than pathchar and similar in accuracy to pathrate, while providing faster estimation than both. Another advantage of CapProbe is its lower computation cost, since no statistical post processing of probing data is required.

Categories and Subject Descriptors

C.2.3 [**Network Operations**]: Network Monitoring

General Terms

Measurement, Experimentation, Performance

Keywords

Network capacity, Bottleneck bandwidth, Packet pair dispersion

1. INTRODUCTION

Estimating the capacity of an Internet path is a fundamental problem that has received considerable attention in the last few years. Knowledge of the capacity of a path can be put to good use in various scenarios. Using such information, multimedia servers can determine appropriate streaming rates while ISPs can keep track of the characteristics of their own links. Further, recent research in overlay networks and application layer multicast can benefit from such capacity information in better structuring their overlays and trees.

Work on estimating path capacity has been based on monitoring either delays of packet pairs and trains or their dispersion. Tools adopting the former approach include pathchar [8, 5]. Pathchar uses ICMP replies from routers to estimate link capacities based on the variation of the round-trip delay as the probing packet size is increased.

"Packet dispersion" techniques rely on probing the path with a series of "packet pairs" or "packet trains", and statistically processing the probing packets' dispersion induced by the path of interest. An early packet dispersion approach, bprobe [3], was proposed by Carter and Crovella. Lai [11] filtered packet pair measurements, whose potential bandwidth, derived from the minimum separation of the packet pair at the sender, was less than the measured bandwidth. Packet Tailgating was another technique proposed by Lai [12]. Recently, Vicat in [6] proposed another filter based technique called tracerate. Paxson [14] identified multi-channel links as a failure case of packet pairs and presented the Packet Bunch Modes (PBM) technique to overcome this limitation. Dovrolis [4] presented the most detailed and revealing analysis of the capabilities and limitations of packet dispersion techniques. He also introduced the well-known capacity estimation tool pathrate, which first uses packet pairs and if this yields a multimodal distribution, then uses packet trains of increasing length. Further details on previous work are presented in Section 8.

All the techniques mentioned above relied either only on delay [8, 5] or only on dispersion [4, 12] of probe packets. CapProbe, the tool we present and study in this paper, combines dispersion and delay measurements of probing packet pairs. CapProbe is based on a simple and fundamental observation: a packet pair that produces either an overestimation or an under-estimation of capacity must have suffered cross-traffic induced queuing at some link (similar queuing observations have also been made by the authors in [4, 7]). CapProbe filters out such distorted measurements by tracking packet pair delays. It only uses packet pairs with minimal end-to-end delays. As we will show in this paper, CapProbe is accurate across a wide range of path and network traffic parameters. The only scenario where the tool fails to consistently estimate capacity correctly is when cross-traffic is both intensive and non-reactive (like constant rate UDP traffic).

In this work, we present and study in some detail CapProbe algorithms and their performance. We assess the ac-

SIGCOMM'04, Aug. 30–Sept. 3, 2004, Portland, Oregon, USA.

curacy and convergence speed of CapProbe, and using measurements on the Internet, we compare its performance to other tools such as pathrate and pathchar. CapProbe algorithms include filtering for minimal queuing time among packet pairs, convergence speed-up by varying probing parameters, and convergence determination tests that improve the chances of the tool to reach an accurate capacity estimate at termination, while minimizing traffic overhead and the time to convergence.

An active probing version of CapProbe can be implemented using ICMP messages or UDP messages as probing packet pairs. In such cases, we say that "active probing", or "out-of-band" probing is used to determine the path capacity. ICMP messages are a fine alternative, provided destination hosts are not discarding such messages. And UDP messages are a fine alternative provided it is feasible to implement the tool on both ends of a path. In case of difficulties with either ICMP, or a two end implementation of UDP probing, it would be advantageous to use "passive probing" or "in-band probing". As an example of passive probing, one might consider the use of CapProbe techniques within TCP. We have designed a minimal modification to the TCP sender protocol that allows accurate estimation of path capacity without sending any special messages; relying only on the normal packet flows within the TCP connection. Details of this technique, which we call TCPProbe, will be reported in future work due to space limitation.

This paper also includes a probability analysis to identify the time it takes CapProbe to converge on average, and a mathematical analysis of the fundamental property of dispersion and delay that CapProbe relies on.

The remainder of the paper is organized as follows. Section 2 describes the packet pair technique on which a number of capacity estimation techniques including CapProbe are based. Section 3 discusses the main idea underlying CapProbe. Section 4 discusses the effect of probing packet size on the probability of queuing of a packet pair. Section 5 presents an analysis of the probability of obtaining a sample not affected by queuing. Section 6 presents algorithms to detect and speed-up convergence of CapProbe. In Section 7, we present results of simulations and measurements showing the performance of CapProbe and comparing it with two well-known capacity estimation techniques, pathchar and pathrate. Previous works on capacity estimation are discussed in Section 8. Section 9 concludes the work and discusses avenues for future work.

2. BACKGROUND

The basic Packet Pair algorithm [9, 2] relies on the fact that if two packets sent back-to-back are queued one after the other at the narrow link, they will exit the link with *dispersion* T given by:

$$T = L/B$$

where L is the size of the second packet, and B is the bandwidth of the *narrow link*, i.e., the capacity limiting link.

If the two packets have the same size, their transmission delays are the same. This means that after the narrow link, a dispersion of T will be maintained between the packets even if faster links are traversed downstream of the narrow link. This is shown in Figure 1(a), where S is the source, D is the destination, and link A-B is the narrow link. The narrow link capacity can then be calculated as:

$$B = L/T$$

The Packet Pair algorithm assumes that the packets will queue next to each other at the narrow link. The presence of cross-traffic can invalidate this assumption.

Previous researchers have noted that capacity estimates resulting from Packet Pair dispersion can be inaccurate. This inaccuracy can be caused either by interference from cross-traffic or by the end systems' inability to measure dispersion accurately. Below, we discuss the effects of cross-traffic, which can cause compression or expansion of dispersion. Compression results in over-estimation of capacity, while expansion results in under-estimation.

Capacity over-estimation occurs when the dispersion between the packet pair at the destination is smaller than what would be introduced by the narrow link. This may happen whenever the narrow link is not the last link on the path, i.e., when so-called post narrow links are present. If the first packet of a pair queues at a post-narrow link, while the second experiences queuing for a shorter time than the first (for example, when no cross packets are injected between the pair), the dispersion between the packets decreases. Figure 1(b) shows how dispersion can decrease at a post-narrow link. When dispersion of a packet pair sample is compressed resulting in capacity over-estimation, the first packet of the probing pair will have queued at a post-narrow link due to interference from cross-traffic. Compression and its effect have been found to be more pronounced when probe packets are smaller than cross-traffic packets [4].

Capacity under-estimation occurs when the dispersion between the packet pair at the destination is larger than what would be introduced by the narrow link in the absence of cross-traffic. This increase of dispersion happens due to cross-traffic packets being served (transmitted) in between packets of a packet pair probe. Such expansion can happen anywhere on the path, before, at, or after the narrow link. Figure 1(c) shows how under-estimation of capacity can occur. When under-estimation occurs, the second packet of the probing pair will have queued due to interference from cross-traffic. We emphasize that queuing of the second packet behind the first packet does not lead to expansion of the probing pair dispersion. Capacity under-estimation has been shown to be more pronounced when the size of cross-traffic packets is smaller than that of probing packets [4].

3. CAPPROBE

The main idea underlying CapProbe is that at least one of the two probing packets must have queued if the dispersion at the destination has been distorted from that corresponding to the narrow link capacity. This means that for samples that estimate an incorrect value of capacity, the sum of the delays of the packet pair packets, which we call the *delay sum*, includes cross-traffic induced queuing delay. This delay sum will be larger than the *minimum delay sum*, which is the delay sum of a sample in which none of the packets suffer cross-traffic induced queuing. The dispersion of such a packet pair sample is not distorted by cross-traffic and will reflect the correct capacity. Based on this observation, CapProbe calculates delay sums of all packet pair samples and uses the dispersion of the sample with the minimum delay sum to estimate the narrow link capacity.

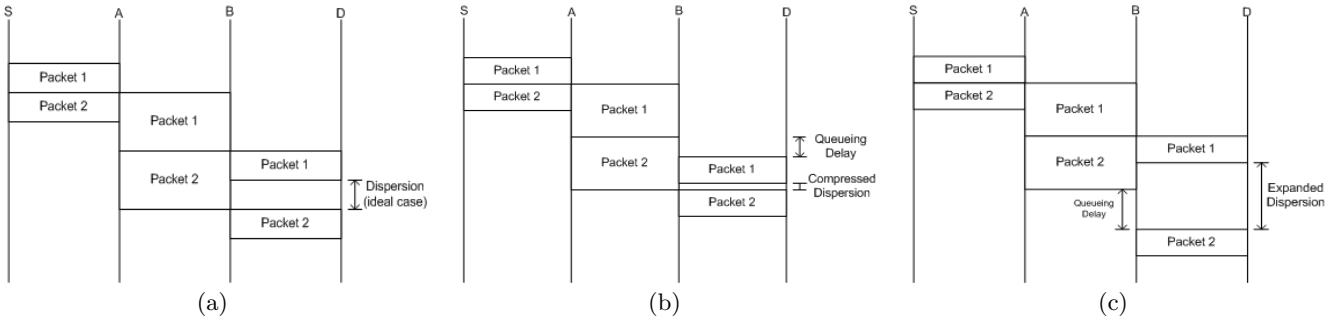

Figure 1: Packet Pair dispersion. (a) Ideal case. (b) Over-estimation of capacity. (c) Under-estimation of capacity.

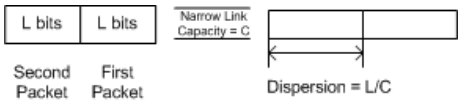

Figure 2: Packet Pair arriving at the narrow link of capacity C.

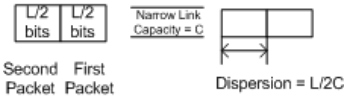

Figure 3: Dispersion reduced due to smaller packet size.

While pathchar used only packet delays and other schemes [4, 12] used only packet pair dispersion, CapProbe combines dispersion and delay measurements of packet pair probes. Searching for the pair with the minimum delay sum implies no post processing of probing pair data. Thus CapProbe promises lower computation costs, and faster capacity estimation. Such features may allow the use of CapProbe in an "on-line" mode, in cases where the capacity to be estimated changes relatively frequently. This might be the case in wireless links in which the quality of the link varies frequently. For example, in the CDMA-based 1xRTT cellular technology, the system design employs varying transmission bit rates to optimize correct reception probability.

CapProbe also requires minimal storage since only a search for a minimum is performed, as compared to gathering data for a histogram to be post-processed. For CapProbe to accurately estimate the narrow link capacity, at least one packet pair sample with the minimum delay sum must be received at the destination. In a network such as the Internet in which the traffic intensity has ample fluctuations including lull periods due to reactive TCP flows, there is a good likelihood of obtaining one or more of the desired samples. In fact, our experiments below encountered very few cases that are deprived of such samples. The cases in which these samples are sometimes not obtained correspond to a highly congested (almost 100% congested), UDP-predominant (i.e., non-reactive) network.

4. EFFECT OF PROBING PACKET SIZE

In this section, we discuss the effect of the size of probing packets on the accuracy of CapProbe estimation. Our discussion makes use of the well-known queuing theory result that the probability of queuing of a packet depends only on the traffic load in the network [10] and is independent of the size of the packet or cross-traffic packets (assuming that the packet probe rate does not significantly change the traffic load on the network). We also mention here that, as previous authors have discussed, for packet pair dispersion to measure the narrow link capacity, the two packets should have the same size.

For CapProbe to estimate accurately, it is sufficient that neither packet of the packet pair has suffered any queuing. We first discuss the effect of packet size on the queuing probability of the second packet. Consider the packet pair shown in Figure 2, where both packets have a size of L bits each. Assume these packets arrive back-to-back at a narrow link with capacity C bps. Note that the arrival of a packet occurs when the last bit of the packet is received.

Assuming that no cross-traffic interferes with the packets at the narrow link, the second packet departs L/C time units after the first packet, and this is also equal to the dispersion T between the packets. This dispersion is the time in which cross-traffic packets can interfere with the packet pair at a post-narrow link. That is, a cross-traffic packet "arriving" in this time will be served between the two packets, causing the second packet to queue. This consequently causes expansion of dispersion. We refer to this time as the "vulnerability window" of the packet pair.

The probability of queuing of the second packet can be reduced by decreasing the size of the packets. As shown in Figure 3, the size of the packets is reduced to $L/2$ bits. The "vulnerability window" in this case is $T = L/2C$, which is smaller than L/C. Thus, by reducing the packet size, the chances of the second packet being queued are reduced. This reduces the chances of capacity under-estimation.

We now consider the effect of packet size on probability of queuing of the first packet of a probing pair. We observe that the queuing behavior of the first packet is similar to that of an independent single packet. Since the probability of queuing of a single packet is independent of its size, this also holds true for the first packet of a packet pair.

Thus, a smaller packet size decreases the chances of queuing of the second packet, while the probability of queuing of the first packet is independent of the packet size. Consequently, we conclude that smaller packet sizes have a larger probability of going through without suffering any queuing.

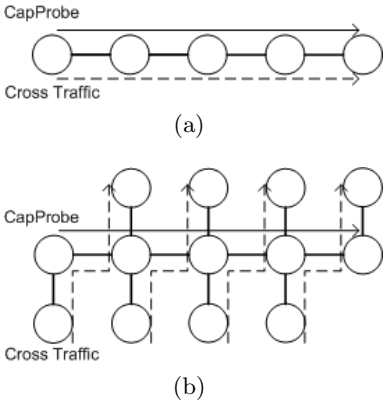

Figure 4: (a) Path Persistent (b) Non-persistent cross-traffic.

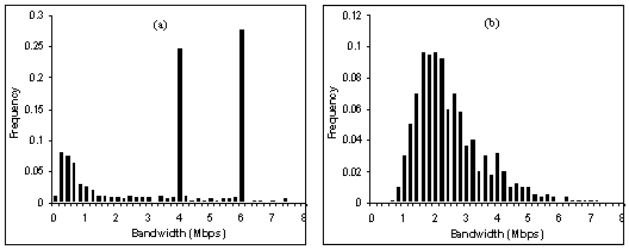

Figure 5: Frequency of occurrence of bandwidth samples when packet size of probes is (a) 100 and (b) 1500 bytes.

Yet, previous authors [4] have shown that while small packet sizes can reduce under-estimation, they can also increase over-estimation. So, is our observation in conflict with these earlier observations? Our observations are actually entirely consistent with these earlier observations. According to our observation, the second packet has a smaller probability of being queued when the packet size is decreased. Though this decrease in packet size does not change the probability of queuing of the first packet, the "relative" probability of queuing of the first packet with respect to the second packet is increased. This increases the probability of over-estimation. Note though that decreasing the size reduces the queuing probability of the packet pair and thus leads to a higher probability for the packet pair to go through without being queued. Increasing the packet size has the opposite effect.

Another effect of decreasing the packet size is that the "magnitude" of over-estimation is also increased. To explain this, suppose the first packet suffers more queuing than the second, leading to compression of the packet pair. Clearly, the compression ratio will be larger when the original dispersion is smaller, i.e., when packet sizes are smaller.

To explain these observations further, we present results of a simulation consisting of a 6-hop path with link bandwidths $= 10, 7.5, 5.5, 4, 6, 8$ Mbps. Cross-traffic was path-persistent and long-range dependent (LRD). Path-persistent and non-persistent stand for the two extreme cases of cross traffic routing as shown in Figure 4. The path persistent cross-traffic packets follow the same path as the packet pair

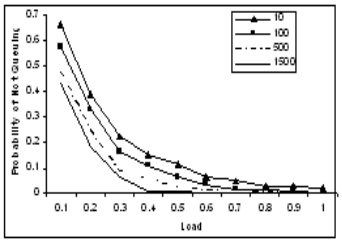

Figure 6: Probability that a sample does not suffer queuing when cross-traffic packet size is 550 bytes.

(Figure 4(a)), whereas the non-persistent cross-traffic packets exit one hop after they enter the path (Figure 4(b)). In order to model LRD traffic, we used a number of Pareto sources with shape parameter $\alpha = 1.9$ [16]. NS-2 [1] was used as the simulation environment. The cross-traffic packet size was uniformly distributed between 40 and 1500 bytes. The cross-traffic created a traffic load of 50% on the narrow link. This is similar to the scenario used by Dovrolis et al [4]. Figure 5(a) shows the frequency of occurrence of a bandwidth sample when the probe packet size is 100 bytes. Figure 5(b) shows the same when the probe packet size is 1500 bytes.

In Figure 5(a), over-estimated capacity occurs with a high frequency, while in Figure 5(b), under-estimation is predominant. Clearly, smaller probing packet sizes lead to a higher chance of over-estimation. Yet, in Figure 5(a), the capacity mode (i.e., the capacity with highest frequency of occurrence) occurs with a relative frequency of almost 25%, while in Figure 5(b), the capacity mode occurs with a frequency of less than 4%. Thus, even though smaller probing packets increase the chances of over-estimation relative to chances of under-estimation, smaller probe packets have higher chances of accurate estimation.

We also found that the probability of a packet pair not suffering queuing was around 13% when the probing packet size was 100 bytes. Note that this probability is smaller than the frequency of occurrence of the correct capacity, since the latter can include some samples where both packets suffered queuing delay by the same amount, and thus dispersion reflected the right capacity. When packet size was 1500 bytes, this probability was around 1.5%.

We thus see that decreasing the packet size of the pair leads to a higher probability of obtaining a sample that does not suffer any queuing. This means that CapProbe should work best when the probing packet size is the smallest possible. Due to the impact of operating system clock granularity, there are practical limits to how much the probing packet size can be reduced. The authors in [11] and [4] have also addressed this issue. Since dispersion of a packet pair is a function of the packet size of the second packet, the larger the second packet's size, the easier it will be for clocks to measure dispersion accurately. We discuss the issue of probe packet sizing further in Section 6.

We studied via simulation the effect of probing packet size on the probability of obtaining a queuing-free packet pair sample. Using the same topology as above, we performed simulations for LRD cross traffic. Probing packet pairs were sent every 200msec. The size of probing packets and the traffic load on the links was varied. Figure 6 shows the

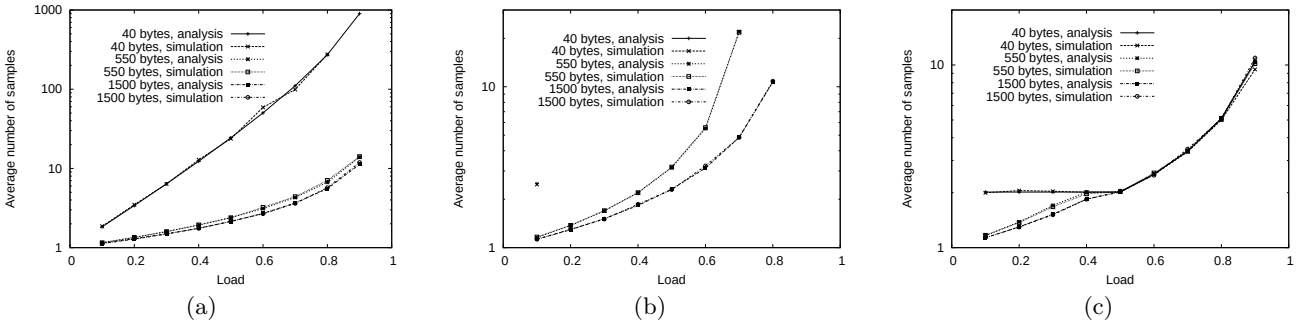

Figure 7: Analytical and simulation results of average number of samples for a single link. (a) Poisson CT. (b) Deterministic CT. (c) Pareto ON/OFF CT.

probability of obtaining queuing-free probing packet pairs when cross-traffic packet size is 550 bytes. The legend shows probing packet sizes. The graph shows that packet pairs with smaller packet size have a higher probability of not hitting any queues, which is consistent with our explanation above.

5. DERIVATION OF NON-QUEUING PROBABILITY

CapProbe estimates capacity accurately once a packet pair goes through the path with no queuing due to interfering traffic. The probability of obtaining such a sample, which we call a "good" sample, is an indication of how quickly CapProbe converges to a correct capacity estimate. Below, we determine the probability of obtaining a good sample and from that, the average number of samples required for convergence.

We present a queuing model that can predict the probability of a "good" sample for a single link in the case of Poisson, Deterministic and Pareto On/Off cross traffic (CT) with deterministic packet size, since these distributions produce packet flows with significantly different variance in inter-arrival time and thus pose different challenges to CapProbe. We use simulations to extend our evaluation to the case of LRD traffic, which does not lend itself easily to analytic modeling. Finally, we extend our evaluation to a path consisting of multiple links with cross-traffic being path-persistent, non-persistent, and a combination of these two.

An assumption we make here is that packet pairs arrive according to a Poisson distribution, and thus they take what amounts to "a random look" at the link. This assumption becomes more accurate as the path length increases, and the time interval between successive packet pairs is randomly affected by various delays along the path. We also assume that packet pairs do not constitute a significant load on the network since they are sent infrequently as a non-intrusive probe. Finally, we assume that buffers are large enough; that is, queues are effectively infinite.

We now calculate the probability that a packet pair sample is not affected by cross-traffic at a link. There are two ways in which cross-traffic can affect a sample: i) Cross-traffic is present upon arrival of the first packet; ii) Cross-traffic packets arrive between the packet pair, specifically, between the arrival instants of the two packets. We simplify the analysis by disregarding the case when cross-traffic packets arrive between a packet pair but do not cause the second packet to queue, and thus our analysis is rather conservative and tends to over-estimate the number of samples N required for convergence.

5.1 Poisson Cross Traffic

For condition (i) to be false, no cross-traffic packets should be found upon arrival of the first packet. The probability p_1 that the first packet arrives to an empty system, is given by:

$$p_1 = 1 - \lambda/\mu$$

where λ and μ are the traffic arrival rate to the link and the service rate of the link, respectively.

For condition (ii) to be false, there should be no arrivals between the packet pair. If the dispersion of the packet pair is denoted as τ, the probability p_0 of no arrivals during τ is given by [10]:

$$p_0 = e^{-\lambda\tau}$$

The probability p_{link}, of no queuing and therefore no distortion at the link is equal to the product of p_1 and p_0:

$$p_{link} = p_0 p_1$$

The expected number of samples, N, needed to obtain a good sample on a given link is given by:

$$N = \sum_{k=1}^{\infty} k p_{link}(1 - p_{link})^{k-1} = \frac{1}{p_{link}}$$

We perform NS-2 simulations to evaluate the values of N for a single link of capacity 4Mbps. Studies have shown that Internet traffic consists of primarily 3 packet sizes, 40, 550 and 1500 bytes [13]. We evaluate N for cross-traffic composed only of one of these packet sizes. The probing packet size is fixed at 500 bytes and the dispersion τ is set to 0.4msec, corresponding to link capacity of 10Mbps. Note that all simulation results presented in this section are the averages of 10 runs with different seeds for random number generation. Unless otherwise specified, these parameters are used in all the simulation studies presented in this section.

In Figure 7(a), we show analytical and simulation results for values of N when cross traffic is Poisson. The curves represent cross traffic of different packet sizes, and the packet sizes are shown in the legend. We found that simulation

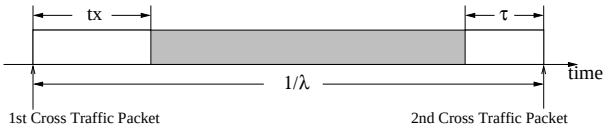

Figure 8: An illustration of the time interval between the arrivals of two deterministic CT packets.

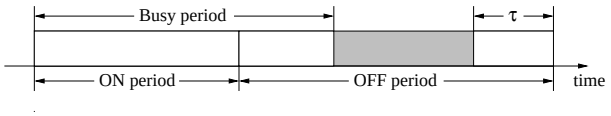

Figure 9: An illustration of an Pareto ON/OFF cycle for the third case.

results for Poisson cross-traffic match the analytically obtained curves very well. We observed that smaller cross-traffic packets tend to reduce the probability of obtaining good samples, since at the same link load, cross traffic with smaller packets has higher packet arrival rate λ, which results in a higher chance of the second packet being queued. Considering that the Internet consists of a combination of small and large packets, we expect CapProbe to be able to converge rather quickly when the arrivals of cross traffic packets is closer to Poisson distribution.

5.2 Deterministic Cross Traffic

Deterministic cross traffic packets arrive at the link periodically, so we consider one inter-arrival period of length $1/\lambda$. As shown in Figure 8, if we denote the transmission time of a cross-traffic packet as tx, the arrival of the packet pair must occur after the transmission of the first cross-traffic packet to avoid the queuing of the first probing packet. In addition, since the dispersion of the packet pair is τ, the first probing packet must arrive early enough before the arrival of the next cross-traffic packet to allow the second probing packet to start transmission without being queued. In other words, the first probing packet must arrive during the shaded time interval in Figure 8. Therefore, the probability of no queuing is calculated as follows:

$$p_{link} = max(0, 1 - \frac{tx + \tau}{\frac{1}{\lambda}}) = max(0, 1 - \lambda(tx + \tau))$$

The analytical and simulation results for Deterministic cross traffic are shown in Figure 7(b). It is clear that the analytical results match simulation results very well. Note that when the cross-traffic packet size is 40 bytes, good samples can only be obtained at very light load of 0.1. When link load increases, the higher arrival rate and thus shorter inter-arrival time of the cross-traffic packets prevent the packet pair from going through the link without being queued. However, as cross-traffic packet size increases, a good sample can normally be obtained in less than 20 samples.

5.3 Pareto On/Off Cross Traffic

For Pareto on/off cross traffic, the source enters ON and OFF state alternately. In the ON state, the source transmits packets at a deterministic rate, whereas in the OFF state, the source does not transmit any packets. The time for a source to remain in ON or OFF state follows Pareto distributions. In our model, we assume the ON and OFF periods are independently identically distributed, i.e., the probability density function (pdf) $f(t)$ of the ON/OFF time periods t and its mean $\bar{t}$ are given by:

$$f(t) = \frac{\alpha k^\alpha}{t^{\alpha+1}} \qquad 0 < k \leq t, 1 < \alpha \leq 2$$

and

$$\bar{t} = \frac{\alpha k}{\alpha - 1}$$

where α and k are the shape parameter and scale parameter of the Pareto distribution, respectively. Since the mean arrival rate of the cross traffic packets is λ and the mean length of ON periods is equal to that of the OFF periods, the arrival rate of the cross traffic during ON periods should be 2λ.

Based on the relationship of $1/2\lambda$ (the inter-arrival time of cross traffic during an ON period), tx (the transmission time of cross-traffic packets), and τ (the dispersion of a packet pair), we derive the non-queuing probability of the packet pair in the presence of one Pareto cross-traffic flow. Three cases need to be considered, depending upon whether the link has idle time in ON/OFF periods, and the length of that idle time.

1. If $tx < 1/2\lambda < tx + \tau$, the inter-arrival time of cross-traffic packets is larger than their service time, so there will be idle time during an ON period. However, the idle time is not long enough for a packet pair to arrive during the idle time and not queue. In this case, a good sample can only arrive during an OFF period. In the OFF period, a packet pair does not suffer queuing if and only if the residual lifetime of the OFF period when the first probing packet arrives is longer than the dispersion τ. Given the distribution of the length of the OFF period, we can find the distribution of the residual lifetime when the renewal process is Pareto. Thus, the probability of obtaining a good sample can be obtained from the probability that the residual lifetime Y is larger than the dispersion τ (the details are shown in Appendix):

$$p_{link} = \frac{1}{2}P[Y \geq \tau] = \begin{cases} \frac{1}{2\alpha}(\frac{k}{\tau})^{\alpha-1} & \text{if } \tau \geq k, \\ \frac{1}{2}[1 - \frac{(\alpha-1)\tau}{\alpha k}] & \text{if } \tau < k. \end{cases}$$

2. If $1/2\lambda > tx + \tau$, the link has idle time in both ON and OFF periods, and the idle time in ON periods is long enough for a packet pair to arrive without suffering queuing. In other words, good samples can occur in both ON and OFF periods. Considering ON and OFF periods separately, the conditional probability of obtaining a good sample given that the source is in ON state can be obtained in a similar way as the Deterministic cross traffic case shown in Section 5.2, whereas the conditional probability given that the source is in OFF state is equal to $P[Y \geq \tau]$ in case 1. Thus, by conditioning and un-conditioning on the state of the source, the probability of obtaining a good sample can be formulated as:

$$p_{link} = \begin{cases} \frac{1}{2}[1 - 2\lambda(tx + \tau)] + \frac{1}{2\alpha}(\frac{k}{\tau})^{\alpha-1} & \text{if } \tau \geq k, \\ \frac{1}{2}[1 - 2\lambda(tx + \tau)] + \frac{1}{2}[1 - \frac{(\alpha-1)\tau}{\alpha k}] & \text{if } \tau < k. \end{cases}$$

3. If $1/2\lambda < tx$, the inter-arrival time of cross-traffic packets is not long enough to serve one packet, thus the queue will build up in ON periods and decrease during OFF periods. Consequently, good packet pair samples can only survive in OFF periods with idle time longer than the dispersion τ. Due to the difficulty of computing the distribution of the length of the idle time in OFF periods, we simplify

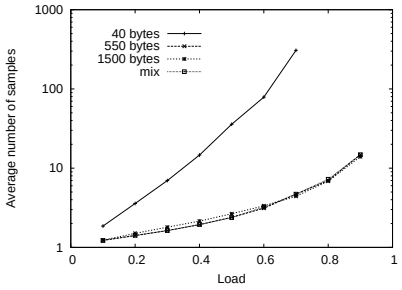

Figure 10: Simulation results of average number of samples for a single link; LRD cross-traffic.

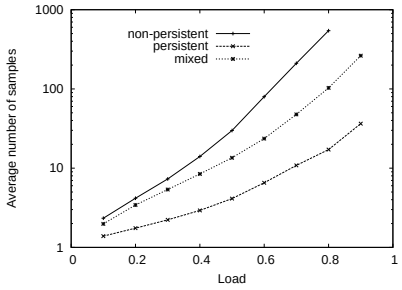

Figure 11: Average number of samples for the 6-link path with persistent, non-persistent, and mixed LRD cross traffic.

the analysis by assuming that every OFF period has idle time longer than τ. Figure 9 shows a typical ON and OFF period, in which the busy period of the link extends to the OFF period, and the first packet of a good sample can arrive any time in the shaded region in the graph. Therefore, the probability of a good sample pair is equal to the proportion of the shaded region in a complete ON and OFF cycle, that is:

$$p_{link} = 1 - \frac{\lambda}{\mu} - \frac{\tau}{2\bar{t}}$$

Figure 7(c) plots the results obtained for one Pareto source. Even under conditions with heavy load and small cross-traffic packets, a good sample can be obtained within approximately 10 packet pairs. Intuitively, these results are reasonable because good packet pairs have more chances to go through due to the presence of OFF periods.

In summary, the analytic model and simulation results for Poisson, Deterministic and Pareto ON/OFF cross traffic show that the average number of samples N for CapProbe to obtain a correct capacity estimation is affected by link load, cross-traffic packet size, and the pattern of the cross traffic. However, under normal conditions in which links are not extremely heavily loaded and the cross-traffic packet sizes are not too small, a good sample can be obtained rather quickly, irrespective of the traffic patterns of the cross traffic. To further demonstrate this observation, we present some simulation results for more realistic LRD cross traffic in the following subsection.

5.4 Long Range Dependent Cross Traffic

Figure 10 shows simulation results for LRD cross-traffic. We evaluate N for cross-traffic composed only of one of the three packet sizes (40, 550, and 1500 bytes) and also a combination of these sizes with the percentages 50%, 25% and 25% respectively, similar to the measured Internet traffic [13]. In this figure, we observe a similar trend as before: the number of required samples is large when cross-traffic packets are 40 bytes, since smaller packets can interfere more easily with probe packets, whose size is 500 bytes. In addition, mixing the cross-traffic with different packet sizes gives results similar to those for 550- and 1500-byte cross-traffic packets.

We now extend the results to the same 6-hop path used in Section 4. Note that we do not study this longer path through analysis owing to the more intractable nature of such analysis. We mention here that developing a more sophisticated analytical model to better study the impact of path length on the probability of an unqueued sample is part of our future work.

We show results only for LRD cross-traffic consisting of a mix of different packet sizes in the same ratio as earlier. The cross-traffic can be path-persistent (Figure 4(a)), non-persistent (Figure 4(b)), or a more realistic mix of both. In this last case, around 50% of the traffic at every link is path-persistent. All links on the path were equally loaded.

In Figure 11, we show the average number of samples needed to obtain a good sample for the 6-hop path obtained using simulations. Under non-persistent cross-traffic, it is more plausible that the packet pairs encounter independent queues. On the other hand, under persistent cross-traffic, a correlation among the successive queues is more likely. Thus, probe packets that are not queued at one link, have a high chance of not being queued at subsequent links. In accordance with this intuition, we found, as shown in Figure 11, that persistent cross-traffic easily allows the passage of probe samples without queuing, while non-persistent cross-traffic does not. The number of samples required when cross-traffic is a mix lies between the two extremes of only persistent and only non-persistent traffic. We expect Internet traffic to be a mix of persistent and non-persistent sources. For mixed LRD traffic, even for loads as high as 80%, a good sample can be obtained in around 100 samples. Based on this observation, we choose 100 as the value of a parameter in Section 6.

6. TECHNIQUES FOR CONVERGENCE DETECTION AND SPEED-UP

A question that arises when using CapProbe is: how do we know whether the capacity estimated by CapProbe is accurate or not? In other words, how can we determine whether the minimum delay sum in a set of packet pair samples is actually equal to the no-queuing (minimum) delay sum. At the beginning of an experiment, the current minimum delay sum in a set of packet pair samples is still likely to contain some amount of queuing delay. The higher the congestion, the higher this probability, and the longer the time to obtain a correct sample. Thus, in a sense, we would like to be able to: 1) Use some indicators to establish confidence in the capacity estimated by CapProbe; and 2) Stop sending probes based on the above indicators.

We begin by observing that by summing the delays of the packet pair packets, CapProbe was able to identify and eliminate samples whose dispersion was distorted. On the other hand, the summing of delays also leads to the loss of potentially useful information: the individual delays of the

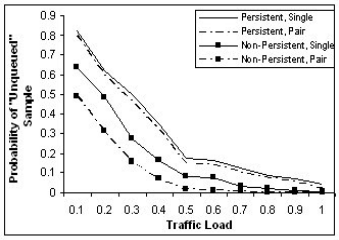

Figure 12: Probability of an unqueued sample for pairs and single packets.

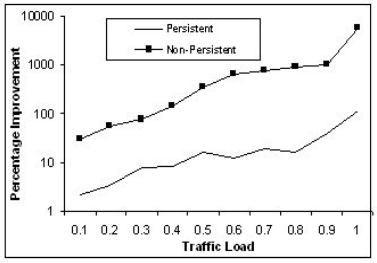

Figure 13: Percentage increase in probability of unqueued sample when using single packets instead of packet pairs.

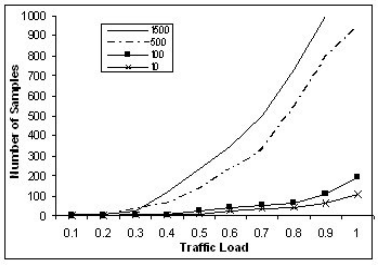

Figure 14: Number of samples required to satisfy minimum delay sum condition.

first and second packets. Our aim thus, is to use individual delays to determine convergence soon after CapProbe has indeed achieved it.

Consider the set of packet pair probes, $i = 0, 1, 2, ...$, let d_i^1 and d_i^2 represent the delays of the first and second packets of the ith probe. Let the minimum delay sample be the jth sample. Thus, $min\{d_1 + d_2\}$ occurs at the jth sample.

Now consider the minimum among the delays of the first packet of all packet pair samples, i.e., $min\{d_1\}$. Let $min\{d_2\}$ represents the corresponding quantity for the second packet.

In the set of packet pair samples, it is not necessary that $min\{d_1 + d_2\} = min\{d_1\} + min\{d_2\}$. In other words, it is not necessary for the minimum delay sum to be equal to the sum of the minimum delays. It could happen, for instance, that the delay of the first (or the second) packet in the minimum delay sum is greater than the minimum delay of the first (or the second) packet among all samples. In such a situation, $min\{d_1 + d_2\} > min\{d_1\} + min\{d_2\}$.

Therefore, if, in a set of packet pair samples, $min\{d_1 + d_2\} > min\{d_1\}+min\{d_2\}$, then we can deduce the following:

- If the delay of the first (or the second) packet in the minimum delay sum sample is greater than $min\{d_1\}$ (or $min\{d_2\}$), we conclude that this packet suffered some queuing delay.
- In either case, the minimum delay sum is clearly greater than the no-queuing delay sum and thus, the dispersion obtained from the minimum delay sum could have been a distorted one.

In this way, the additional information relating individual minimum delays with the minimum delay sum helps in weeding out incorrect minimum delay samples. We refer to this relation as the *minimum delay sum* condition.

6.1 How much does this extra information improve detection of convergence?

The probability of a single packet going through without queuing is much higher than the probability of a packet pair suffering the same fate. Thus, by comparing the delay of the first (second) packet in the minimum delay sample with the minimum delay of the first (second) packet, we can identify incorrect minimum delays sums with the same probability as that of the first (second) packet not being queued in the network.

We evaluated by simulations the probability of a packet pair as well as a single packet going through without being queued under different conditions. A 6-hop linear topology, similar to the one used in earlier sections, was used for the simulations. Cross-traffic was LRD. The cross-traffic packets were a mix of three packet sizes as described in previous sections. The probing packets had a size of 500 bytes.

Simulations were performed for both path-persistent and non-persistent cross-traffic sources. Figure 12 shows the probability of obtaining a sample which does not suffer any queuing in the case of packet pairs and single packets, for different values of narrow link traffic load. Figure 13 shows the percentage increase in the probability of obtaining a sample which does not suffer any queuing when using single packets compared to using packet pairs. The increase is clearly very large when the traffic load increases. Also, for non-persistent traffic, the increase is larger than when traffic is path-persistent. The more than 10 times increase for highly loaded non-persistent traffic is very significant.

6.2 More on Minimum Delay Sum

In order to show the effect of using the minimum delay sum condition with CapProbe, we show some simulation results. The simulation parameters are similar to the ones used in Section 6.1, except for that the size of probe packets was varied. Figure 14 shows the number of samples required to satisfy the minimum delay sum condition. The index in Figure 14 shows different probing packet sizes.

For low loads, the minimum delay condition is satisfied after a small number of samples, while higher loads require a larger number of samples. Also, packet pairs using smaller packets satisfy this condition much faster than those using larger packets.

To avoid situations where the minimum delay sum is equal to the sum of the minimum delays but these minimum delays are not the no-queuing delays (this can easily happen, for instance, in the first few samples, either of the packets of the pair may not have gone through without queuing), we continue sending a few samples even after the minimum delay sum is equal to the sum of the minimum delays. Thus, the minimum delay sum condition is said to be *satisfied* when

the minimum delay sum is equal[1] to the sum of the minimum delays for the n previous samples and the minimum delay sum and minimum delays have not changed during these n samples. Through experiments, we found 40 to be a good value for n.

6.3 Algorithm

Based on the observations in the last two sections, we outline an algorithm to detect and speed-up convergence. We again bring to the attention of the readers our result of Section 5 that for a 6-hop path with up to 80% traffic load, a sample not affected by queuing could be obtained within 100 samples. This value of 100 is used as the number of packet pair samples in each phase of the algorithm described below.

We also found while conducting experiments that when the packet size was very small such that the operating system could not measure the dispersion accurately, the bandwidth obtained from the dispersion of samples varied quite a lot. In our algorithm outlined below, we need to determine whether the variance in bandwidth estimated from packet pair samples is caused due to operating system not being able to measure dispersion accurately. We used a simple test: if the ratio of maximum to minimum bandwidth estimated by samples > 50, it is likely to be due to measurement errors. The value of 50 was obtained solely through experimentation.

In the algorithm, each "run" is defined to consist of packet pair samples, all having the same packet size. The "run" stops either if the minimum delay sum condition is satisfied or 100 samples have been sent. In the beginning, we select two initial values of packet sizes, $p_1 = 700$ bytes and $p_2 = 900$ bytes. These initial values are chosen since they lie between very small values which cause problems in measurement and large values (such as 1500 bytes, which is the typical MTU value) which have a higher chance of suffering expansion. These are also values suggested by previous authors [4].

1. The first run uses p_1 as the packet size. If the minimum delay sum condition is not satisfied in this run, then:
   (a) If the bandwidth estimated varies a lot across samples (if the ratio of maximum to minimum bandwidth estimated by packet pair samples is greater than 50), it is an indication that the operating system clock is unable to measure dispersion accurately. Thus, the packet sizes p_1 and p_2 are increased by 20% and the algorithm goes back to Step 1. The upper bound on p_1 and p_2 is 1500 bytes, which is the largest packet that does not suffer fragmentation.
   (b) If the bandwidth estimated does not vary significantly across samples, it is an indication that no ample could go through without queuing. Since decreasing packet size leads to a higher chance of obtaining an unqueued sample, the packet sizes p_1 and p_2 are decreased by 20% and the algorithm goes back to Step 1.
2. If the minimum delay sum condition is satisfied in the previous run with packet size p_1, another run with packet size p_2 is employed.

[1]Since operating system measurements can have some error, we say that the minimum delay sum is equal to the sum of minimum delays when their difference is less than 1%.

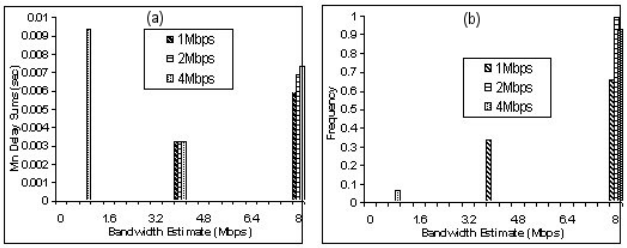

Figure 15: (a) Minimum delay sums and (b) frequency of occurrence when cross-traffic is TCP and packet size of probes is 200 bytes.

   (a) If the minimum delay sum condition is not satisfied in this run, then packet sizes p_1 and p_2 are decreased or increased according to the rules in 1(a) and 1(b) and the algorithm goes back to Step 1.
   (b) If the minimum delay condition is satisfied
      i. If the capacities resulting from the two runs are within 5%, the algorithm stops, yielding the average of the two runs as the capacity.
      ii. Else, the algorithm goes back to Step 1.

Thus, the CapProbe algorithm continues to run till capacities obtained from two consecutive runs (using different packet sizes) are similar and the minimum delay sum condition is satisfied in each of these runs.

7. RESULTS

In this section, we present results of simulations and Internet measurements to evaluate the performance of CapProbe. We compared CapProbe with two previously proposed well-known capacity estimation schemes, pathchar [8] and pathrate [4] (for a brief description, see Section 8).

We first show results of simulations experiments to test CapProbe. The network topology is the same 6-hop linear path used in previous sections. The capacity is measured at the destination. The cross-traffic on the path can be either persistent (Figure 4(a)) or non-persistent (Figure 4(b)). The different traffic types we used for the cross-traffic were TCP, CBR and LRD. 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 simulation time was 100 sec. The size of cross-traffic packets was 500 bytes. We study below the packet pair delay sum statistics, in particular, the minima of such delays and corresponding bandwidth estimate distributions obtained for various bandwidth estimates. We show some of the results from these simulations.

Figure 15(a) shows the minimum packet pair delay sums when packet size of probes is 200 bytes and cross-traffic is path-persistent. The index in the figures shows maximum cross-traffic rates[2]. Figure 15(b) shows the frequency distribution of bandwidth estimates from packet dispersion.

We make the following observations from the graphs: the value of minimum packet pair delay sums is smallest at the point corresponding to the narrow link capacity, which is 4Mbps. We note here that when cross-traffic rate is 4Mbps,

[2]TCP rates fluctuate, but are limited to between 1Mbps and 4Mbps, depending upon the scenario.

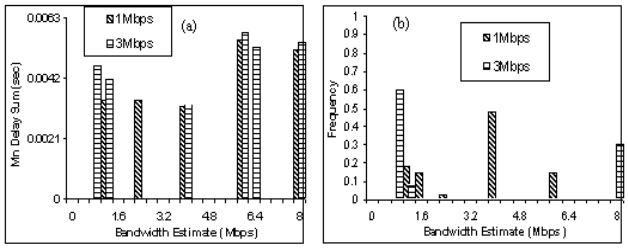

Figure 16: (a) Minimum delay sums and (b) frequency of occurrence when cross-traffic is TCP and packet size of probes is 500 bytes.

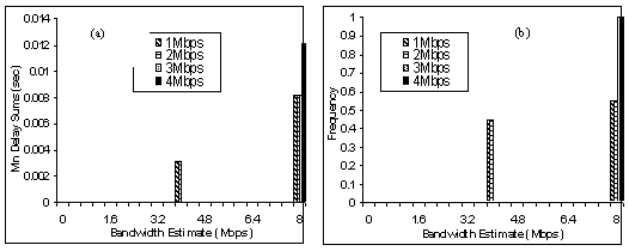

Figure 17: (a) Minimum delay sums and (b) frequency of occurrence when cross-traffic is UDP.

i.e., equal to the narrow link capacity, CapProbe still works. Thus, when cross-traffic is TCP, CapProbe measures the right capacity even for highly congested links. Looking at Figure 15(b), the strongest mode always occurs at 8Mbps. This mode is the Post-Narrow Capacity Mode (PNCM) introduced in [4] and represents compression.

We now show results when size of packet pair packets is increased to 500 bytes. Figure 16(a) shows the minimum packet pair delay sums. Figure 16(b) shows the frequency distribution of bandwidth estimates from packet dispersion. From Figure 16(b), when the size of packet pair packets is similar to that of cross-traffic packets, results show the emergence of the ADR (Asymptotic Dispersion Rate), which has been noted in [4]. The strongest mode is the ADR when cross-traffic is 3Mbps. From Figure 16(a), the CapProbe scheme still estimates the right capacity for different cross-traffic values. Thus, even when the strongest mode is the ADR, CapProbe is able to estimate the correct capacity.

We now show simulation results when the cross-traffic is path-persistent and CBR using UDP as transport layer protocol. Figure 17(a) shows the minimum packet pair delay sums. Figure 17(b) shows the frequency distribution of bandwidth estimates from packet dispersion. The UDP cross-traffic results are different from TCP results since UDP is not reactive to congestion.

From Figure 17(a), CapProbe predicts the correct capacity till the UDP cross-traffic is 2Mbps, i.e., till a load of 50% on the narrow link. For higher cross-traffic rates, no samples corresponding to the correct capacity are obtained. In fact, all samples for cross-traffic of 3Mbps or 4Mbps have a packet dispersion corresponding to an estimated bandwidth of 8Mbps (Figure 17(b)). This is basically the PNCM mode described in [4]. Also, the strongest mode is an overestimation for all cross-traffic loads. It should be noted here that high-rate UDP is a worst-case for CapProbe (and in general, for all packet pair-based techniques), and it is also not very realistic. Our basis for experimenting with such cross-traffic is to identify cases in which our technique fails.

For all other combinations of cross-traffic type (TCP, CBR, LRD) and nature (persistent, non-persistent), CapProbe was able to estimate the correct capacity. We do not show these results due to space constraints.

We now show results of measurement experiments. We evaluated the performance of CapProbe with respect to its speed and accuracy. CapProbe was implemented using ICMP PING packets, sent in pairs. This meant that we had to be careful in choosing our test paths, since PING packets are blocked by some Internet nodes (firewalls etc). It should be noted that CapProbe can just as well be implemented as a UDP-based client-server application, similar to pathrate. This has the disadvantage that a module needs to be installed at the destination, similar to pathrate.

A Pentium 4 2.5GHz machine at our university was used as the source machine, running CapProbe, pathrate and pathchar. A number of destination machines on the Internet, Abilene and CalREN networks were chosen to provide different types of paths (the names of these destination machines are shown in Table 1). Abilene is a high-speed 10Gbps national backbone connecting a number of universities. Abilene provides a different environment for testing than the commercial Internet since it is very high-speed. CalREN is also a high-speed network connecting universities in California. We had knowledge of capacities of all links on certain paths on Abilene and CalREN.

The path to the UCLA-2 machine involved a wireless hop, with 802.11b being the wireless technology. In all other paths (except WLSH where the narrow link capacity was 1.5Mbps), the narrow link was 100 Mbps. Since pathrate requires the installation of a module at the destination and we did not have access to all destination machines, results for pathrate could not be obtained for all test paths.

Table 1: Destinations used in our experiments

Host	Location
BERKELEY	University of California, Berkeley
CMU	Carnegie Mellon University
MIT	Massachusetts Institute of Technology
NTNU	National Taiwan Normal University
UA	University of Alabama
UCLA	University of California, Los Angeles
UCLA-2	University of California, Los Angeles (including a wireless hop)
UCLA-3	University of California, Los Angeles
UCONN	University of Connecticut
UCSD	University of California, San Diego
WLSH	National Wuling Senior High School, Taiwan
YAHOO	Yahoo.com

Table 2: Convergence time and capacity (in Mbps) estimated over Internet

	YAHOO 66.218.70.49		WLSH (1.5Mbps) 210.70.26.17	
	time	C	time	C
1	0'03	97	0'13	1.50
2	0'01	93	0'56	1.53
3	0'03	90	0'13	1.50
4	0'03	99	0'13	1.53
5	0'01	98	0'13	1.50

Table 3: Convergence time and capacity (in Mbps) estimated over Abilene

	MIT 18.181.0.31		CMU 128.2.11.43		UCONN 137.99.29.54	
	time	C	time	C	time	C
1	0'04	95	0'04	99	0'13	98
2	0'07	98	0'04	97	0'13	99
3	0'07	97	0'02	99	0'04	98
4	0'07	96	0'02	98	0'05	93
5	0'03	97	0'04	99	0'25	85

Table 4: Convergence time and capacity (in Mbps) estimated over CalREN

	UCLA 169.232.56.135		UCSD 132.239.50.184		BERKELEY 169.229.131.109	
	time	C	time	C	time	C
1	0'02	99	0'01	97	0'02	99
2	0'02	93	0'02	97	0'01	99
3	0'01	91	0'08	96	0'03	97
4	0'02	97	0'01	97	0'03	98
5	0'01	97	0'05	98	0'05	95

Table 2, 3 and 4 show the speed and accuracy of CapProbe in 5 runs on various test paths on the Internet, Abilene and CalREN networks respectively. We found that CapProbe is able to estimate capacity with a high degree of accuracy typically within a few seconds. The quick and accurate estimation of CapProbe indicates that it may be suitable for "online" capacity estimation. This can prove useful over wireless links, in which the changing quality of the link can cause the capacity to vary frequently.

Table 5 compares the time taken (in minutes and seconds) and the capacity estimated (in Mbps) by CapProbe, pathchar and pathrate for different test paths. We found both CapProbe and pathrate to be quite accurate in most scenarios. On the other hand, pathchar estimates were incorrect in most scenarios, most likely due to accumulation of errors as estimation proceeds along the path. Also, pathchar required a large amount of time to yield capacity estimates.

In order to get a fair comparison with pathrate, we set the probing rate (in bps) of CapProbe equal to that of pathrate in these experiments. We found that if we did not make the rates equal, CapProbe estimated faster than the results shown in Table 5. We believe that by making the bandwidth consumed by the probing streams equal, we obtained a fair comparison between CapProbe and pathrate.

While pathrate typically required on the order of minutes to yield its estimate, CapProbe was able to typically estimate in a minute or sometimes even in a few seconds. Thus, CapProbe proved to be similar in accuracy to pathrate, but was typically able to achieve this in a smaller time. On the UA path, CapProbe is sometimes inaccurate by around 20% and pathrate by around 30%. The path to UA is a long and extremely congested path, and as a result, obtaining samples not affected by queuing is difficult. This caused CapProbe to produce slightly inaccurate estimates in a few runs.

Table 5: Comparison of convergence time and capacity (in Mbps) of CapProbe, pathrate and pathchar

	UCLA-2 131.179.33.171		UCLA-3 131.179.136.151		UA 130.160.47.35		NTNU 140.122.77.6	
	time	C	time	C	time	C	time	C
CapProbe	0'03	5.5	0'01	96	0'02	98	0'07	97
	0'03	5.6	0'01	97	0'04	79	0'07	97
	0'03	5.5	0'02	97	0'17	83	0'22	97
	0'07	5.6	0'01	98	0'09	98	0'04	99
	0'03	5.6	0'02	99	0'09	95	0'04	96
pathrate	6'10	5.6	0'16	98	5'19	86	0'29	97
	6'14	5.4	0'16	98	5'20	88	0'25	97
	6'5	5.7	0'16	98	5'18	133	0'25	97
	6'14	6.8	0'16	98	5'19	88	0'26	97
	6'20	5.8	0'16	98	5'19	132	0'25	97
pathchar	21'12	4.0	22'49	18	3 hr	34	3 hr	34
	21'21	4.0	22'53	18	3 hr	31	3 hr	35
	21'45	4.0	22'48	18	3 hr	32	3 hr	34
	20'43	3.9	27'41	18	3 hr	34	3 hr	35
	21.18	4.0	29'47	18	3 hr	30	3 hr	35

8. RELATED WORK

Previous work on estimating the capacity of a path has mostly been based on using dispersions of packet pairs or trains. In [3], Carter proposed bprobe, in which filtering methods are applied on packet pair measurements, consisting of different packet sizes. In [11], Lai used packet pair measurements, but filtered them through a kernel density estimator [15]. The kernel density algorithm is known to be statistically valid and is relatively simple and fast to compute. The underlying assumption of all these techniques is that the distribution of measurements obtained from packet pair samples is unimodal, i.e., the sample with the maximum frequency of occurrence corresponds to the capacity.

Paxson showed in [14] that this distribution can be multimodal. He 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 [4] 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 to an over-estimate of the capacity. Underestimation occurs when the network is heavily congested, while over-estimation occurs, to various degrees, when the narrow link is followed by links of higher capacity, referred to as Post-Narrow Links. They also observed that a packet train of N packets is most useful for estimating capacity when $N = 2$, corresponding to a packet pair, since interference from cross-traffic is likely to increase as N increases. Finally, 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 [8]. This technique, based on the generation of ICMP replies from routers, is known to have scalability problems. Pathchar tries to estimate the capacity of a link by sending sets of packets (not packet pairs) to a link, with each set having a different packet size. It assumes the minimum delay in each set to be the no-queuing delay for the particular packet size. The minimum delays for different packet sizes yield a set of

linear equations that are solved to obtain the bandwidth of the link. This procedure is repeated for each link on the path and the minimum among the link bandwidths is chosen as the path capacity. Pathchar is known to consume a significant amount of bandwidth [11]. Also, we tested pathchar and found that its estimation accuracy fell as the path length was increased, mainly due to accumulation of estimation errors. Clink [5] differs from pathchar in the manner it generates its interval capacity estimates. Pchar is based on a similar concept as pathchar and uses regression to determine the slope of the minimum RTT versus the probing packet size. The key difference between pathchar and CapProbe is that whereas pathchar uses packets delays to estimate capacity, CapProbe uses packet delays only as an indicator of which sample's dispersion to choose for estimating capacity. In [2], the authors study another technique based on probing with variable packet sizes.

Packet tailgating is another technique proposed by Lai [12]. 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.

9. CONCLUSIONS

This paper presented and studied a new capacity estimation technique, called CapProbe. CapProbe relies on a novel scheme that uses packet delays to filter out packet pairs with distorted dispersion. Simulations showed that CapProbe is able to estimate capacity correctly except when cross-traffic is both intensive and non-reactive (like UDP). We also compared CapProbe, using measurements, with two well-known capacity estimation methods, pathchar and pathrate. We found that the accuracy of CapProbe is similar to that of pathrate. Pathchar was found to be less accurate. In terms of speed, CapProbe out-performed both pathchar and pathrate.

We are hopeful that new and emerging applications can make use of capacity estimates and this is one direction of our future work. Another direction for the future is to build and study more sophisticated analytical models to gain further insight on the convergence speed of CapProbe in different network conditions.

10. ACKNOWLEDGMENTS

We are grateful to the following people for their help in carrying out CapProbe measurements: Jun-Hong Cui (University of Connecticut), Xiaoyan Hong (The University of Alabama), Yi-Wen Jiu (National Wuling Senior High School, Taiwan) and Che-Chih Liu (National Taiwan Normal University). We also want to thank the anonymous reviewers for their valuable comments and suggestions.

11. REFERENCES

[1] Network simulator ns-2. *http://www.isi.edu/nsnam/ns*.

[2] J. C. Bolot. Characterizing end-to-end packet delay and loss in the internet. In *Proceedings of ACM SIGCOMM*, pages 289–298, September 1993.

[3] R. Carter and M. Crovella. Measuring bottleneck link speed in packet-switched networks. *Performance Evaluation*, 27(8):297–318, October 1996.

[4] C. Dovrolis, P. Ramanathan, and D. Moore. Packet dispersion techniques and capacity estimation. *submitted to IEEE/ACM Transactions of Networking.*

[5] A. B. Downey. Using pathchar to estimate internet link characteristics. In *Proceedings of ACM SIGCOMM*, pages 241–250, September 1999.

[6] M. Goutelle and P. Vicat-Blanc/Primet. Study of a non-intrusive method for measuring the end-to-end capacity and useful bandwidth of a path. In *Proceedings of ICC*, June 2004.

[7] N. Hu and P. Steenkiste. Evaluation and characterization of available bandwidth techniques. *IEEE JSAC Special Issue in Internet and WWW Measurement, Mapping, and Modeling*, 21(6):879–894, August 2003.

[8] V. Jacobson. Pathchar: A tool to infer characteristics of internet paths. *ftp://ftp.ee.lbl.gov/pathchar/*.

[9] S. Keshav. A control-theoretic approach to flow control. In *Proceedings of ACM SIGCOMM*, pages 3–15, September 1991.

[10] L. Kleinrock. *Queueing Systems, Volume I: Theory*. Wiley, 1975.

[11] K. Lai and M. Baker. Measuring bandwidth. In *Proceedings of IEEE INFOCOM*, pages 235–245, March 1999.

[12] K. Lai and M. Baker. Measuring link bandwidth using a deterministic model of packet delay. In *Proceedings of ACM SIGCOMM*, pages 283–294, August 2000.

[13] S. McCreary and K. Claffy. *Trends in Wide Area IP Traffic Patterns*. Technical Report, CAIDA, February 2000.

[14] V. Paxson. *Measurements and Dynamics of End-to-End Internet Dynamics*. Ph.D. Thesis, Computer Science Division, Univ. Calif. Berkeley, April 1997.

[15] D. Scott. *Multivariate Density Estimation: Theory, Practice and Visualization*. Addison Wesley, 1992.

[16] M. S. Taqqu, W. Willinger, and R. Sherman. Proof of a fundamental result in self-similar traffic modeling. *ACM/SIGCOMM Computer Communications Review*, 27(2):5–23, April 1997.

APPENDIX

To calculate residual lifetime when renewal process is Pareto distribution, we define the following random variables: X is the lifetime of a typical interval with Pareto renewal processes, and Y is the residual lifetime of the selected interval when the first packet in a packet pair probe arrives. Let the residual life have a distribution $\hat{F}(x) = P[Y \leq x]$ with density $\hat{f}(x) = \frac{d\hat{F}(x)}{dx}$, and let the typical lifetime X have a pdf $f(x)$ and cumulative distribution function (CDF) $F(x)$ where $F(x) = P[X \leq x]$ and $f(x) = \frac{dF(x)}{dx}$.

Based on the renewal theory [10], given that $x \geq k$ (according to the definition of Pareto distribution), we obtain

$$\hat{f}(y) = \begin{cases} \int_{x=y}^{\infty} \frac{f(x)}{m_1} dx, & \text{if } y \geq k, \\ \int_{x=k}^{\infty} \frac{f(x)}{m_1} dx, & \text{if } y < k. \end{cases}$$

where m_1 is the mean time between renewals, i.e., mean time of the Pareto distribution $\bar{t}$. Therefore, by integrating the right-hand side of the equations, we obtain the pdf of the residual lifetime distribution:

$$\hat{f}(y) = \begin{cases} \frac{1-F(y)}{m_1} = \frac{\alpha-1}{\alpha k}(\frac{k}{y})^{\alpha}, & \text{if } y \geq k, \\ \frac{1-F(k)}{m_1} = \frac{1}{m_1} = \frac{\alpha-1}{\alpha k}, & \text{if } y < k. \end{cases}$$

Finally, we can compute the complementary cumulative distribution function (CCDF) $P[Y \geq t]$ as follows, given that $1 < \alpha \leq 2$:

$$P[Y \geq t] = \begin{cases} \int_t^{\infty} \hat{f}(y)dy = \frac{1}{\alpha}(\frac{k}{t})^{\alpha-1}, & \text{if } t \geq k, \\ 1 - \int_0^t \hat{f}(y)dy = 1 - \frac{t}{m_1} = 1 - \frac{(\alpha-1)t}{\alpha k}, & \text{if } t < k. \end{cases}$$