An Inline Measurement Method for Capacity of End-to-end Network Path

Cao Le Thanh Man, Go Hasegawa and Masayuki Murata
Graduate School of Information Science and Technology, Osaka University
E-mail: {mlt-cao, hasegawa, murata}@ist.osaka-u.ac.jp

Abstract— **We previously proposed a new version of TCP, called Inline measurement TCP (ImTCP), in [1]. The ImTCP sender adjusts the transmission intervals of data packets and then utilizes the arrival intervals of ACK packets for available bandwidth estimation. This type of active measurement is preferred for TCP connections because the obtained results are as accurate as those of other conventional types of active measurement, even though no extra probe traffic is injected onto the network. In the present research, we combine a new capacity measurement function with ImTCP in order to enable simultaneous measurement of both capacity and available bandwidth in ImTCP. The capacity measurement algorithm is a new packet-pair-based measurement technique that utilizes the estimated available bandwidth values for capacity calculation. This new algorithm promises faster measurement than current packet-pair-based measurement algorithms for various situations and works well for high-load networks, in which current algorithms do not work properly. Moreover, the new algorithm provides a confidence interval for the measurement result.**

I. Introduction

The capacity of an end-to-end network path, which is considered to be the smallest capacity of network links along a path, is the maximum possible throughput that the network path can provide. Traffic may reach this maximum throughput when there is no other traffic along the path. The available bandwidth indicates the unused bandwidth of a network path, which is the maximum throughput that newly injected traffic may reach without affecting the existing traffic. The two bandwidth-related values are obviously important with respect to adaptive control of the network. In addition, these two values are often both required at the same time. Although network transport protocols should optimize link utilization according to capacity, congestion should be also avoided through the use of available bandwidth information. For route selection or server selection in service overlay networks, information concerning both capacity and available bandwidth offers a better selection than either capacity or available bandwidth information alone. For example, when the available bandwidth fluctuates frequently and the transmission time is long, the capacity information may be a better criterion for the selection. However, when the available bandwidth appears to be steady during the transmission, the available bandwidth should be used for the selection. Moreover, the billing policy of the Internet service provider is based on both the capacity and the available bandwidth of the access link that they are providing to the customer.

Several passive and active measurement approaches exist for capacity or available bandwidth. Although active approaches are preferred because of their accuracy and measurement speed, sending extra traffic onto the network is a disadvantage that is common to all active measurement tools. For example, Pathload [2] generates between 2.5 and 10 MB of probe traffic per measurement. The average per-measurement probe traffic generated by Spruce [3] is 300 KB. For routing in overlay networks, or adaptive control in transport protocols, these measurements may be repeated continuously and simultaneously from numerous end hosts. In such cases, the probes will create a large amount of traffic that may degrade the transmission of other data on the network, as well as the measurement accuracy itself.

We therefore propose an active measurement method that does not add probe traffic to the network. The proposed method uses the concept of ”plugging” the new measurement mechanism into an active TCP connection (*inline measurement*). We previously introduced ImTCP (Inline measurement TCP) [1], a Reno-based TCP that deploys inline measurement. The ImTCP sender not only observes the ACK packet arrival intervals in the same manner as TCP Westwood [4], but also actively adjusts the transmission interval of data packets, in the same way that active measurement tools use probe packets. When the corresponding ACK packets return, the sender utilizes the arrival intervals to calculate the measurement values.

The *available bandwidth* measurement algorithm for ImTCP is described in detail in [5]. For each measurement, the ImTCP sender searches for the available bandwidth only within a given search range. The search range is a range of bandwidth that is expected to include the current available bandwidth and is calculated statistically from the previous measurement results. By introducing the search range, sending packets at an extremely high rate can be avoided. This also allows the number of packets for the measurement to be kept small, so that measurement is still possible when the TCP window size is relatively small. The search range is divided into multiple sub-ranges of identical width of bandwidth. For each of the sub-ranges of the bandwidth, the sender transmits a group of TCP data packets (a packet stream), the transmission rate of which varies to cover the sub-range. The sender then determines whether an increasing trend exists in the transmission delay of packets in each stream when the echoed (ACK) packets arrive at the sender host. The increasing trend indicates that the transmission rate of the stream is larger than the current available bandwidth of the network path. This fact allows the sender to infer the location of the available bandwidth in the search range. The simulation results show that the ImTCP sender can perform periodic measurements at short intervals, on the order of several RTTs and the measurements results reflect well the changes in the available bandwidth of the network.

In the present paper, we introduce an inline measurement algorithm for *capacity* for ImTCP. The proposed algorithm utilizes the arrival intervals of the ACK packets of packet pairs (PPs) that are sent back-to-back. In the existing PP-based capacity measurement algorithm [6-8], the PPs that are cut into by other packets from cross traffic at the bottleneck link causes incorrect capacity estimation and are therefore eliminated from the data used in the calculation. Unlike previous algorithms, the proposed algorithm in ImTCP can use these PPs for capacity measurement, which enables ImTCP to collect more information from PPs so that faster and more accurate measurement can be expected.

The main concept of the proposed capacity measurement algorithm of ImTCP is that the available bandwidth information, which can be yielded periodically due to the deployed available bandwidth measurement mechanism, is exploited. The available bandwidth information is used for estimation of the quantity of the cross traffic that is cut in PPs at the bottleneck link so that the interval of the PPs becomes usable for the capacity measurements. The proposed algorithm also uses statistic analysis to calculate the confidence interval of the delivered results.

Through simulation validations, we show that ImTCP can deliver measurement results quickly, independent of the characteristics of the network. In addition, we find that the capacity measurement algorithm works well in extremely high-load networks, in which current measurement algorithms do not work well.

The remainder of this paper is organized as follows. In Section II we discuss PP-based measurement techniques used for inline meaurement. In Section III, we introduce the proposed measurement algorithm for network capacity. In Section IV, we evaluate its performance through simulation experiments. Finally, in Section V, we present concluding remarks and discuss future projects.

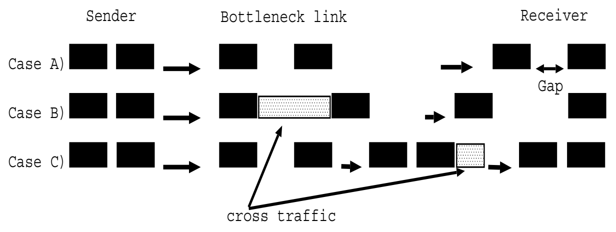

Fig. 1. 3 cases of how the spacing between a pair of packets changes as they travel along a path.

II. PACKET-PAIR-BASED CAPACITY MEASUREMENT ALGORITHMS

Currently there are various approaches for measuring the capacity of an end-to-end network path. Some of these approaches use packets of various size to probe the network and infer the network capacity from the difference in the transmission delays of packets of various sizes [9]. Other approaches use the probe packets in different TTLs to measure all link bandwidth, rather than just the capacity of the bottleneck link [10-12]. These approaches can not be used for inline measurement because changes in TCP data packet size for the purpose of measurement may cause severe deterioration in the data transmission throughput of TCP. We found that only packet-pair-based measurement can be used for inline measurement because no changes in packet size or TTL are required, whereas packets that are sent back-to-back can be created with the current ImTCP structure without requiring any changes.

A. Packet pair technique

The intuitive rationale of capacity measurement using packet pairs is that if two packets are sent close enough together in time to cause the packets to queue back-to-back at the bottleneck link, then the packets will arrive at the destination with the same spacing as when they left the bottleneck link [9].

The intuitive rationale of capacity measurement using packet pairs is that if two packets are sent close enough together in time to cause the packets to queue back-to-back at the bottleneck link, then the packets will arrive at the destination with the same spacing as when they left the bottleneck link [9]. The spacing will remain unchanged because no link downstream of the bottleneck link has a lower bandwidth than the bottleneck link, as shown in Case A of Figure 1, which is a variation of a figure taken from [13]. In this case, the capacity of the bottleneck link (C) can be calculated by the Equation:

$$C = \frac{P}{Gap} \quad (1)$$

where P is the size of the packet pairs, and Gap is the time spacing of the two packets when arriving at the receiver.

However, when a packet pair travels along the path, two more situations can occur. As shown by Case B in Figure 1, the two packets may be cut into by other packets from cross traffic at the bottleneck link. The result is that, the spacing between the two packets becomes larger than expected. In this case, Equation 1 leads to an under-estimation of the capacity. In another case, indicated by Case C in Figure 1, the packet pairs may pass back-to-back through the bottleneck link, but in a link downstream of the bottleneck link, the pairs again get in queue, and the spacing between the two packets is shortened. In this case, Equation 1 leads to over-estimation.

Current PP-based measurement techniques use only the PPs described in Case A to calculate capacity. These techniques have various mechanisms for determining the Case-A PPs from all of the received PPs. Some tools assume a high frequency of appearance of Case-A PPs and so search for these PPs from a frequency histogram (Pathrate [6]) or a weighting function (Nettimer [7]). CapProbe [8] repeatedly sends packet pairs until it discovers a Case-A PP, based on the transmission delay of the packets.

When the network path is almost empty, Case-A PPs may appear with the highest frequency. However, when other traffic appears on the network, the bottleneck link is often congested, and so the probability of a Case-B PP appearing is much higher than that of a Case-A or Case-C PP. In this case, CapProbe will spend an extremely long time for capacity searching, and Pathrate and Nettimer will deliver incorrect estimations.

We therefore propose a new technique by which to calculate capacity that can use either Case-A PPs or Case-B PPs. This is possible because of the available bandwidth information that is available in ImTCP.

B. Capacity calculation

Let us consider the timing of the arrival at the bottleneck link of a packet pair (Figure 2). We assume that the first packet arrives at t_1 and the second packet arrives at t_2. During the interval from t_1 to t_2, packets from other traffic may arrive at the bottleneck link. The second packet (P2) must wait in the queue for the processing of the packets from other traffic. Therefore, the time spacing (*Gap*) of the packet pair after leaving the bottleneck link is the total of the queuing time and the processing time of the second packet. That is:

$$Gap = \frac{P+L}{C} \tag{2}$$

where P is the size of the packet and L is the amount of the cross traffic that arrives at the bottleneck link during

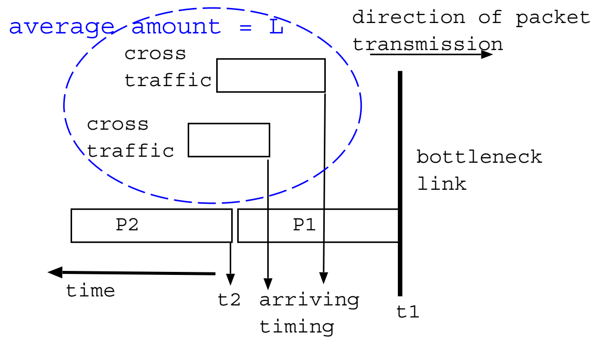

Fig. 2. Arrival time at the bottleneck link of packet pairs and cross traffic

the interval (t_1, t_2). Supposing that the bottleneck link of a network path is the link having the smallest available bandwidth, we can then calculate the total transmission rate of the cross traffic at the bottleneck link as: $C - A$, where A is the current available bandwidth. Let δ be the time spacing of the packet pair upon arrival at the bottleneck link ($\delta = t_2 - t_1$). Then, the average value of L is:

$$L = \delta(C - A) \tag{3}$$

from Equation (2) and (3), we can write:

$$C = \frac{P + \delta(C - A)}{Gap}, \tag{4}$$

or

$$C = \frac{P - \delta \cdot A}{Gap - \delta}. \tag{5}$$

Equation (5) enables the calculation of capacity from the PPs for both Case A and Case B. In the next section, we propose the new capacity calculation algorithm based on Equation (5).

III. INLINE MEASUREMENT ALGORITHM FOR CAPACITY

A. Implementation of packet pairs in ImTCP

As introduced in a previous study [1], a measurement program is inserted into the sender program of TCP Reno to create an ImTCP sender. The measurement program is located at the bottom of the TCP layer. When a new data packet is generated at the TCP layer and is ready to be transmitted, the packet is stored in an intermediate FIFO buffer. The measurement program waits until the number of packets in the intermediate buffer becomes sufficient and then decides the time at which to send the packets in the buffer in order to create measurement streams. When no measurement stream is needed, the program immediately passes all of the data packets to the IP layer. In the previous version of ImTCP [1], we decided that the program forms and sends one measurement stream for the available bandwidth in each RTT in order to maintain fairness with respect to traditional TCP Reno.

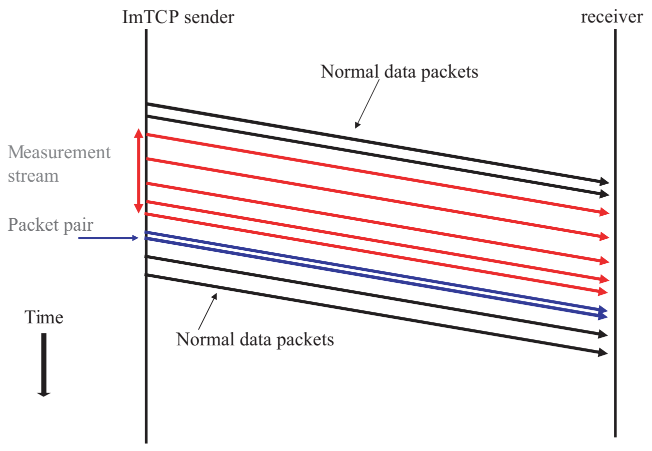

Fig. 3. Creation of packet pairs in ImTCP

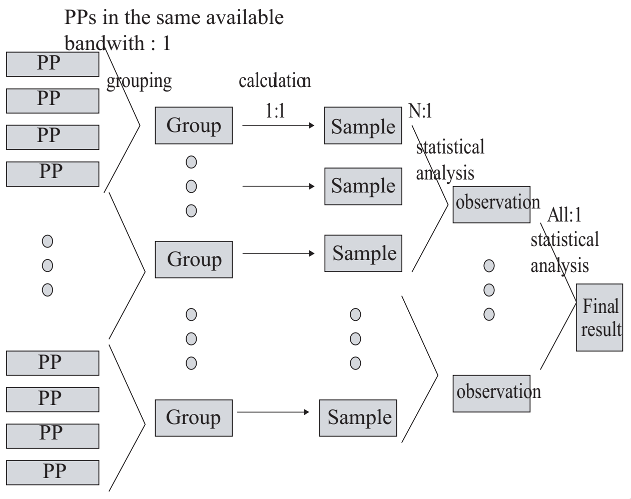

Fig. 4. Proposed algorithm

In ImTCP, 2–4 measurement streams are required for a result of available bandwidth. As mentioned above, each PP is formed and transmitted after each measurement stream. Therefore, 2–4 results of PP can be obtained during the interval of two consecutive measurement result of available bandwidth.

To the current system, we add the creation and transmission of a PP just after the transmission of each measurement stream, as depicted in Figure 3. Note that during the transmission of a measurement stream, some packets may arrive at an intermediate FIFO buffer, so that there are usually a number of packets available in the buffer just after the transmission of a stream [1]. Therefore, there is almost no waiting time required in the creation of a packet pair. Therefore, there is almost no effect on the performance of ImTCP by introducing the capacity measurement algorithm.

In ImTCP, 2–4 measurement streams are required in order to determine the available bandwidth. As mentioned above, each PP is formed and transmitted after each measurement stream. Therefore, 2 to 4 results for PPs can be obtained during the interval of two consecutive measurement results for available bandwidth.

B. Proposed measurement algorithm

Next, we explain the procedure for determining the capacity from the measurement results of PPs using Figure 4. The procedure involves the following steps:

- Grouping of packet pairs (PPs): PPs that sent when the measured available bandwidth remains unchanged are placed in the same `group`. The average arival interval of PPs in a group, denoted by $\overline{Gap}$, is then calculated. To obtain a good average value, the number of PPs in each group should be enough large i.e. larger than or equal to 3, as determined herein. Therefore, after grouping, a group having only one or two PPs will be merged with a nearby group.
- Calculation: Based on the $\overline{Gap}$ value of a group, a `sample` of capacity is calculated using the following function.
  If
  $$\frac{A}{P/\delta} > \lambda$$
  $$C = \frac{P}{\overline{Gap}} \tag{6}$$
  otherwise,
  $$C = \frac{P - \delta \cdot A}{\overline{Gap} - \delta} \tag{7}$$
  where A is the available bandwith of the group, and P is packet size.
  When the available bandwidth is approximately equivalent to the rate of the PPs upon arriving at the bottleneck link that is defined as P/δ, the packets may pass through the link without being cut into by other packets (Case A). In this case, Equation (5) is used. On the other hand, since when the arrival rate of the PPs is much higher than the available bandwidth, the probability is high that the PP is a Case-B PP, Equation (1) is used.
- Statistical analysis:
  - We form `obsevations`, each of which is the average value of N samples. N should be large enough so that each observation has high accuracy. But when N is too large, the time required to finish an observation is long. This means that the proposed algorithm can not deliver the measurement results quickly. In the present paper, based on empirical experiments, we recommend $N = 5$.
  - The average value of the observations are calculated as the “final result”. The 90% confidence interval is also calculated to show the degree of fluctuation of the capacity.

IV. Simulation experiments

In this Section, we examine the measurement results of the proposed capacity measurement algorithm through ns-2 simulations [14]. We also compare the proposed algorithm with two existing algorithms, CapProbe [8] and Pathrate[6].

We used the simulation topology shown in Figure 5. The transmission rate of Cross traffic 1 is fixed to 5 Mbps and that of Cross traffic 3 is fixed to 15 Mbps. The packet size distribution of cross traffic is set to the statistical results for the Internet traffic reported in [15]. The simulation time is 80 (s).

A. *Effect of parameters*

- Value of λ
  We set the bottleneck link capacity to 90 Mbps and the transmission rate of Cross traffic 2 to 5 Mbps and examined the measurement results when $\lambda = 0.9$ (Figure 6) and λ = 0.8 (Figure 7). These figures show the changes of the capacity measurement results as the number of PPs sent for the measurement increases. In this case, the load on the bottleneck link is low, so the Equation (1) should normally be used. The setting $\lambda = 0.9$ does not allow the Equation (1) to be used so frequently and therefore leads to a bad result. We see that in this case $\lambda = 0.8$ is a better setting.
  We next show the case when capacity is 80 Mbps and the rate of cross traffic 2 is 20 Mbps in Figure 8 ($\lambda = 0.6$) and 9 ($\lambda = 0.8$). In this case, the rate of the cross traffic is high so Equation (2) need to be used frequently. Therefore, a small value such as 0.6 gives incorrect results of the capacity, $\lambda = 0.8$ is again a good setting in this case.
  Thus, $\lambda = 0.8$ is a suitable setting for the two cases above. We also found that $\lambda = 0.8$ is a good setting in many other cases. Therefore, we use $\lambda = 0.8$ in the following simulations.
  We next show the case when the capacity is 80 Mbps and the rate of Cross traffic 2 is set to 20 Mbps in Figure 8 $\lambda = 0.6$) and 9 ($\lambda = 0.8$). In this case, the rate of the cross traffic is high, so Equation (2) should normally be used. Therefore, a small value, such as 0.6, gives incorrect results for the capacity, and, again, $\lambda = 0.8$ is a good setting in this case. Thus, $\lambda = 0.8$ is a suitable setting for the two cases above, and we found that it is a good setting in many other cases. Therefore, in the following simulations, we used $\lambda = 0.8$.

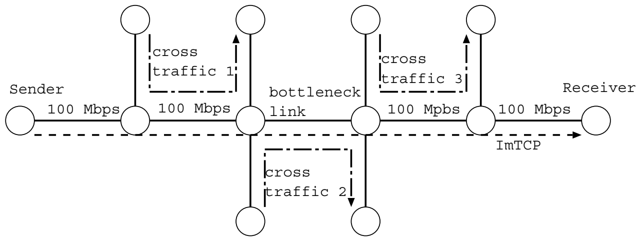

Fig. 5. Simulation topology

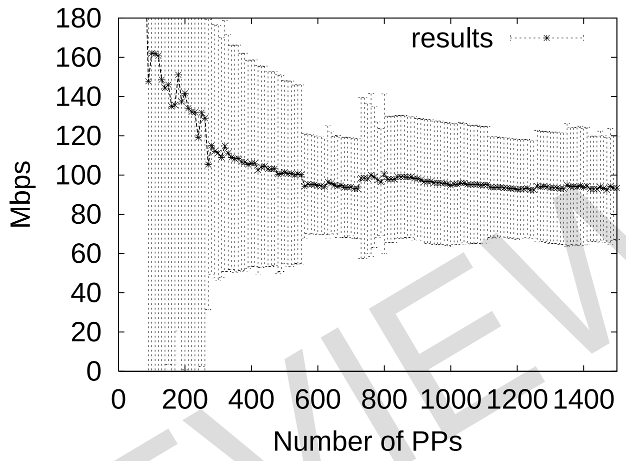

Fig. 6. Measurement results for the proposed algorithm. λ = 0.9. Correct Capacity value =90 Mbps

- Value of N
  Figure 10, 11 and 12 show the measurement results when N is set to 1, 50 and 5, respectively. The large confidence interval in Figure 10 indicates that a small value of N ($N = 1$) is not suitable. On the other hand, Figure 11 indicates that a value of N ($N = 50$) that is too large is unsuitable as well, because in this case the time required for the results to become stable is long. Figure 12 shows the results with the proposed setting ($N = 5$), which can provide fast and good results.

B. *Comparision with CapProbe*

We implement CapProbe algorithm into TCP to compare the performance as fairly as possible. The different point from the original algorithm of CapProbe proposed in [8] is that the packet size is unchanged over the “runs” in the algorithm. That is because in TCP connections, changing the data packet size may cause bad effect on the performance of TCP. In the following simulation results, we show that the restriction on the packet size could be the reason for the bad performance of CapProbe in the following simulations. This means that CapProbe is not suitable for inline measurement for capacity.

We implemented the CapProbe algorithm in TCP in order to compare the performance with the greatest possible impartiality. The difference from the original

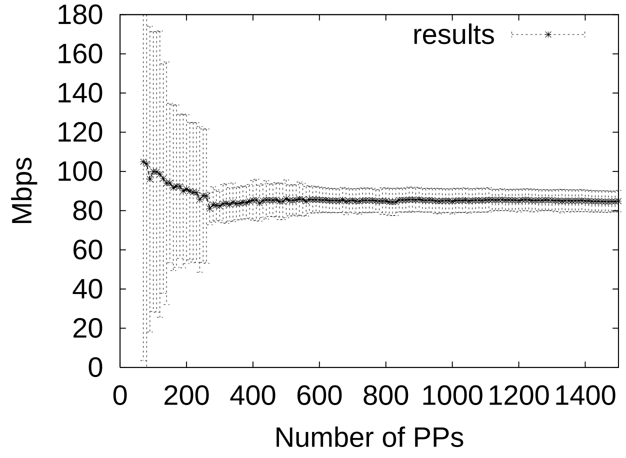

Fig. 7. Measurement results for the proposed algorithm. λ = 0.8. Correct Capacity value =90Mbps

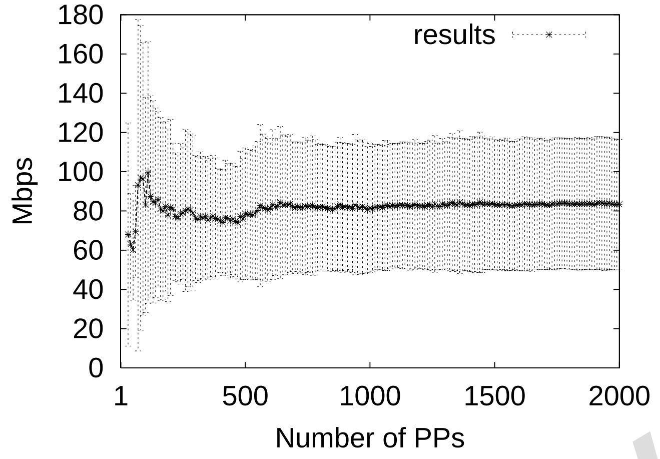

Fig. 10. Measurement results for the proposed algorithm. N = 1

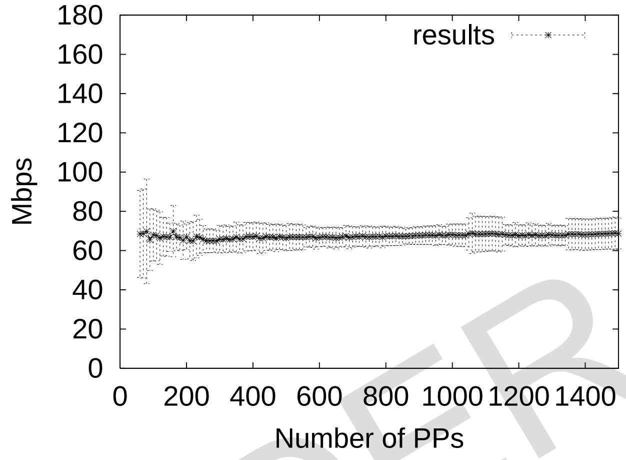

Fig. 8. Measurement results for the proposed algorithm. λ = 0.6. Correct Capacity value =80 Mbps (Measurement result is not correct)

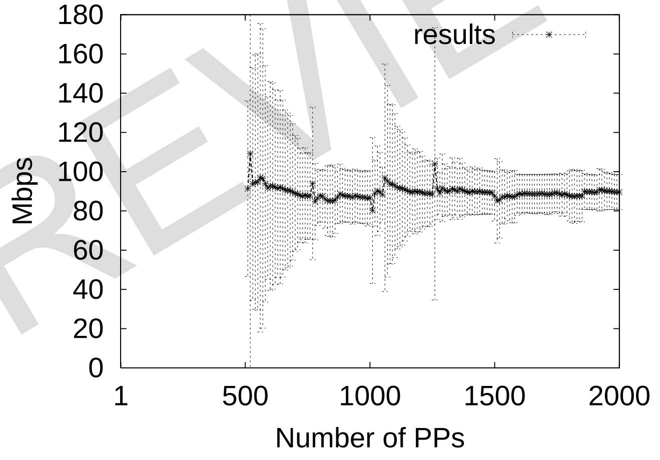

Fig. 11. Measurement results for the proposed algorithm. N = 50

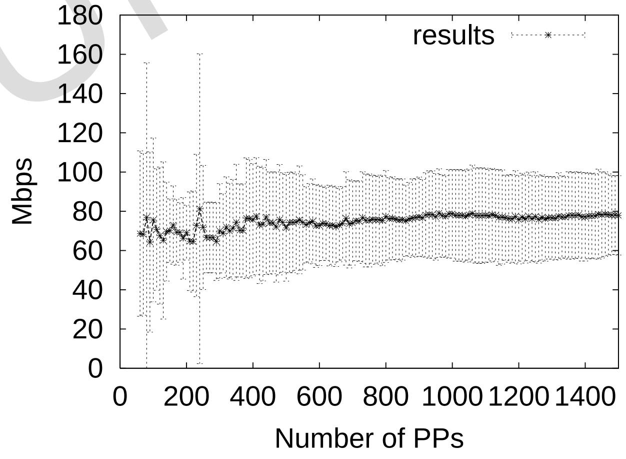

Fig. 9. Measurement results for the proposed algorithm. λ = 0.8. Correct Capacity value =80 Mbps)

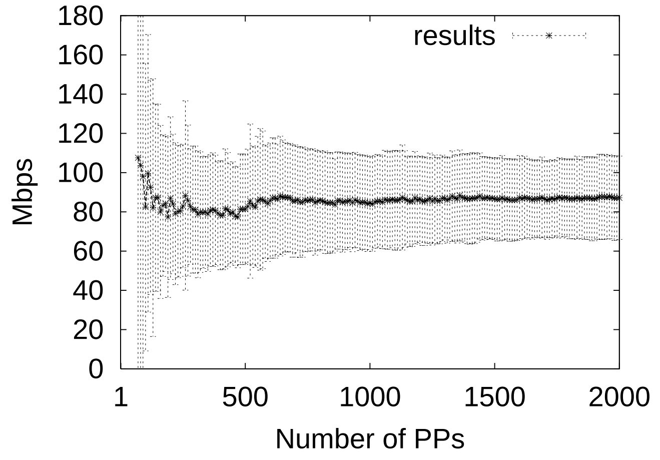

Fig. 12. Measurement results for the proposed algorithm. N = 5

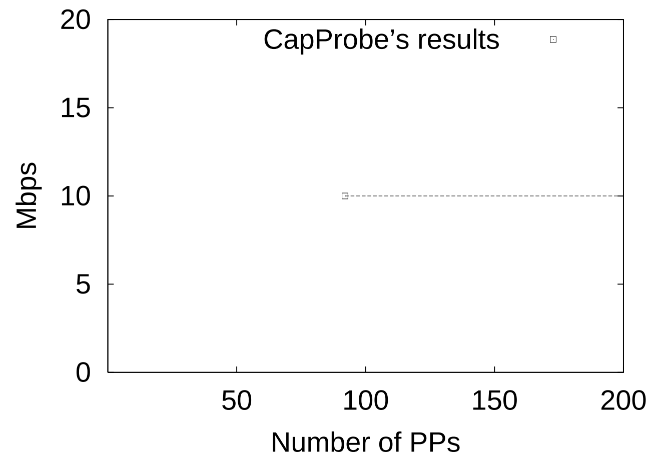

Fig. 13. CapProbe measurement results in small capacity, low network load scenario

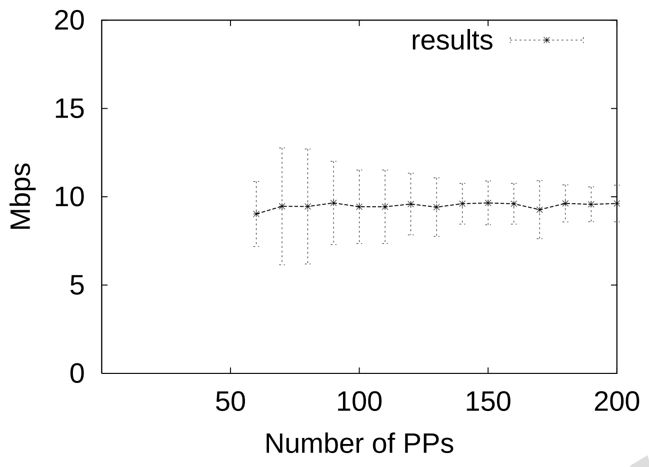

Fig. 14. Proposed algorithm measurement results in small capacity, low network load scenario

CapProbe algorithm proposed in [8] is that the packet size remains unchanged over the ”runs” in the algorithm, because in TCP connections, changing the data packet size may have a bad effect on the TCP performance. In the following simulation results, we show that the restriction on the packet size may be the reason for the poor performance of CapProbe in the following simulations. This means that CapProbe is not suitable for inline measurement of capacity.

1) Small capacity, low network load scenario: The capacity is set to 10 Mbps, and the rate of Cross traffic 2 is set to 4 Mbps. Figure 13 shows that the CapProbe only delivers a measurement result after sending 92 PPs. In contrast, the proposed algorithm delivers good measurement results from 60 PPs, as shown in Figure 14. Moreover, the results obtained by CapProbe have a very small confidence interval (approaching 0), because when CapProbe successfully finds the PP in Case A, the capacity can be calculated exactly. Another advantage of CapProbe is that, compared with the proposed algorithm, CapProbe is simple because it requires no complicated calculations.

2) Large capacity, high network load scenario: The capacity is set to 80 Mbps, and the rate of Cross traffic 2 is set to 60 Mbps. In a network with such a heavy load, the proposed algorithm can perform well (Figure 15), whereas CapProbe can not deliver accurate results (Figure 16), because, in this case, most of the PPs are cut into by other traffic and there are no Case-A PPs.

C. Comparision with Pathrate

In order to accommodate Pathrate algorithm into TCP, we used the interval of PPs delivered in ImTCP to form the histogram to be used in Pathrate. Pathrate also requires the measurement results of a packet train, referred to as the Average Dispersion Rate (ADR) in the Pathrate

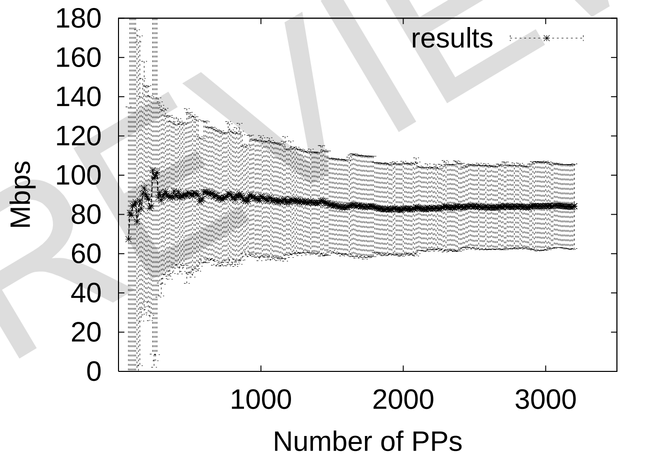

Fig. 15. Proposed algorithm measurement results in large capacity, high network load scenario

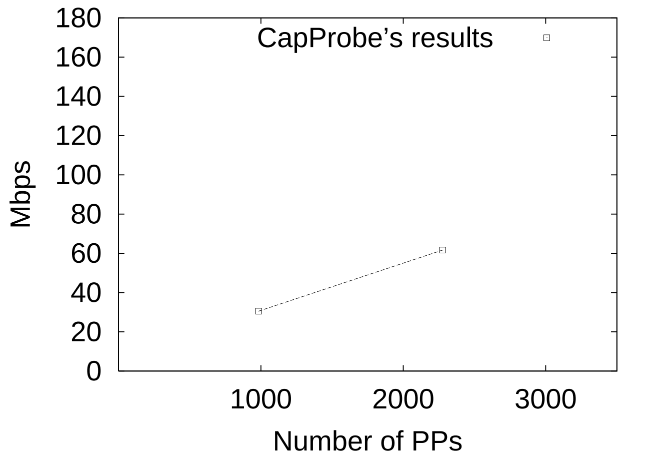

Fig. 16. CapProbe measurement results in large capacity, high network load scenario

TABLE I

MEASUREMENT RESULTS FOR PROPOSED ALGORITHM AND PATHRATE

Cross traffic 2	results	90% conf. inter.	Pathrate
75	81.91	33.89	77.00
60	84.20	43.1	77.00
40	86.80	46.83	80.00
10	80.43	50.01	80.00
0	81.35	12.11	100.00

algorithm [6]. However, integrating the packet train into TCP is difficult because this has an adverse effect on the performance of TCP. Therefore, we perform the packet train measurement separate from TCP connection, in the same environment as that in the TCP connection. The result of ADR is then used to find the measurement result for Pathrate.

1) Normal Internet traffic senario: We used the same environment as that for the above-described simulations, except that the transmission rate of Cross traffic 2 was variable. Table I shows the measurement results for Pathrate together with those for the proposed algorithm. Both of these algorithms can deliver rather good measurement results, independent of the rate of cross traffic. We explain in detail the respective behaviors of these two algorithms in Figure 17. In the figure, the ”Raw data” line indicates the measurement results calculated using Equation (1) that are used in Pathrate, and the ”Proposed method” graph shows the ”observation” results obtained using proposed algorithm. In this case, the red line has a high peak near the correct value of 80 Mbps because there are several packets in Case A. Therefore, Pathrate can deliver good results. The observation results obtained by the proposed algorithm show a concentration at 80 Mbps and so also delivers good results. When the cross traffic is 0 Mbps, the value of ADR is 82Mbps. Among the peaks of the ”Raw data” line that place in the area of larger than 82 Mbps, the highest one is near 100 Mbps, which corresponds to the capacity of the link on the downstream side of the bottleneck link, as shown in Figure 18. Therefore, Pathrate delivers an incorrect result, as shown in Table I.

2) Cross traffic with small packet size scenario: We next show the case when the cross traffic contains mainly packets of small size. We randomly varied the packet sizes of the cross traffic within the range of 400 to 600 KB. In this case, since most PPs are cut into by cross traffic packets, Pathrate should not work very well. The performance of the proposed program in this environment is also examined, and the measurement results are listed in Table II. When the network load is heavy, Pathrate fails to deliver good measurement results because in this case there are almost no Case-A PPs. In contrast, the proposed algorithm can deliver good results,

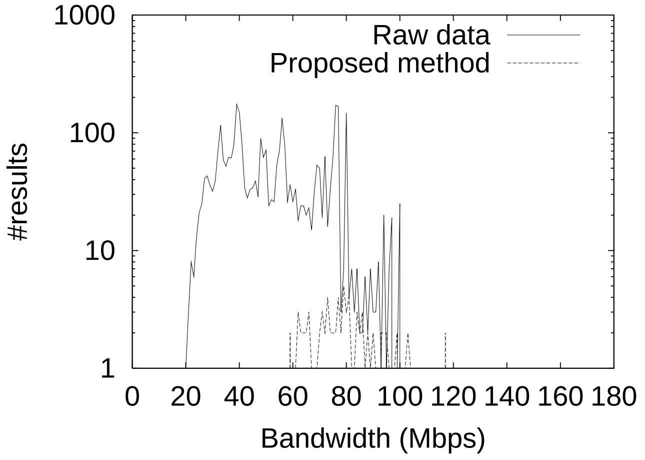

Fig. 17. Measurement results for PPs and observations value by the proposed algorithm. Cross traffic 2's transmission rate = 60 Mbps

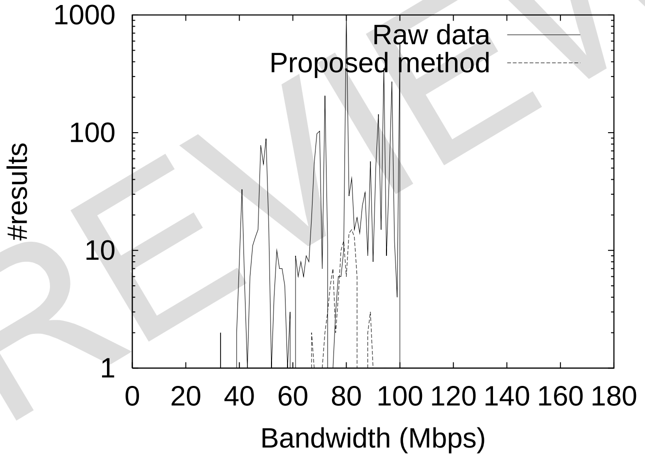

Fig. 18. Measurement results for PPs and observations value by the proposed algorithm. Cross traffic 2's transmission rate = 0 Mbps

regardless of the network load.

V. CONCLUSION AND FUTURE WORKS

In this paper, we have proposed a new capacity measurement technique that is suitable for use in TCP connections. In contrast to existing techniques, the proposed mechanism uses available bandwidth information that is available in ImTCP, which enables the utilization of packet pairs that can not be used in existing techniques to calculate the capacity. The simulation results show that, the proposed technique can deliver measurement results quickly, even for a heavily loaded network, in which other techniques do not work well. We are currently implementing ImTCP using the proposed technique on a FreeBSD system. For the final-version of this paper, we will include the implementation experiment results.

REFERENCES

[1] Cao Man, Go Hasegawa and Masayuki Murata, "Available bandwidth measurement via TCP connection," in *Proceeding of*

TABLE II

MEASUREMENT RESULTS OF PATHRATE AND THE PROPOSED ALGORITHM (SMALL PACKETSIZE CROSS TRAFFIC)

Cross traffic 2	results	90% conf. int.	Pathrate
75	78.98	15.27	49.00
60	79.63	18.81	50.00
40	77.38	20.97	80.00
10	79.56	41.35	80.00
0	86.27	28.91	100.00

the 2nd Workshop on End-to-End Monitoring Techniques and Services E2EMON, Oct. 2004.

[2] M. Jain and C. Dovrolis, "End-to-end available bandwidth: Measurement methodology, dynamics, and relation with TCP throughput," in *Proceedings of ACM SIGCOMM 2002*, Aug. 2002.

[3] J.Strauss, D.Katabi and F.Kaashoek, "A measurement study of available bandwidth estimation tools," in *Proceedings of Internet Measurement Conference 2003*.

[4] M.Gerla, B.Ng, M.Sanadidi, M.Valla, R.Wang, "TCP Westwood with adaptive bandwidth estimation to improve efficiency/friendliness tradeoffs," *To appear in Computer Communication Journal*.

[5] Cao Man, Go Hasegawa and Masayuki Murata, "A new available bandwidth measurement technique for service overlay networks," in *Proceeding of 6th IFIP/IEEE International Conference on Management of Multimedia Networks and Services Conference, MMNS2003*, pp. 436–448, Sept. 2003.

[6] C. Dovrolis and D. Moore, "What do packet dispersion techniques measure?," in *Proceedings of IEEE INFOCOM 2001*, pp. 22–26, Apr. 2001.

[7] K. Lai and M. Baker, "Nettimer: A tool for measuring bottleneck link bandwidth," in *Proceedings of the USENIX Symposium on Internet Technologies and Systems*, Mar. 2001.

[8] R. Kapoor, L. Chen, L. Lao, M. Gerla and M. Sanadidi, "CapProbe: a simple and accurate capacity estimation technique," in *Proceedings of the 2004 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, 2004.

[9] K. Lai and M. Baker, "Measurering link bandwidths using a deterministic model of packet delay," in *Proceedings of ACM Sigcomm*, Aug. 2000.

[10] Bruce A. Mah. Pchar, "`http://www.ca.sandia.gv/~bmah/Software/pchar`,"

[11] V. Jacobson, "Pathchar-A tool to infer characteristics of Internet paths," *`http://www.caida.org/tools/utilities/others/pathchar/`*, 1997.

[12] A. B. Downey, "Using pathchar to estimate internet link characteristics," in *Proceedings of ACM SIGCOMM*, 999.

[13] K. Lai and M. Baker, "Nettimer: A tool for measuring bottleneck link bandwidth," in *Proceedings of the USENIX Symposium on Internet Technologies and Systems*, Mar. 2001.

[14] NS Home Page, "`http://www.isi.edu/nsnam/ns/`,"

[15] "NLANR web site," available at `http://moat.nlanr.net/Datacube/`.