A New Available Bandwidth Measurement Technique for Service Overlay Networks

Cao Le Thanh Man, Go Hasegawa, and Masayuki Murata

Graduate School of Information Science and Technology, Osaka University
1-3, Machikaneyama-cho, Toyonaka, Osaka 560-8531, Japan
Phone: +81-6-6850-6863, Fax: +81-6-6850-6868
{mlt-cao, hasegawa, murata}@ist.osaka-u.ac.jp

Abstract. We introduce a new measurement algorithm for the available bandwidth in a network path between two endhosts. This algorithm is an adaptation from existing active measurement methods for the purpose of being applied in *inline network measurement*. Inline network measurement method measures the available bandwidth by using packets which an active TCP connection transmits, instead of using probe packets, as in the existing methods. The measurement is performed by inferring network characteristics information from transmitting and receiving intervals of TCP data and ACKnowledgement packets. Inline network measurement is useful in service overlay networks, in which up-to-date information on the available bandwidth in the underlying IP network should be known as early as possible. In this paper, we first introduce the measurement algorithm and then discuss the problems with applying the proposed measurement algorithm to an active TCP connection. We evaluate the proposed measurement algorithm using simulation experiments and show that it can provide an estimation of available bandwidth every 2-4 RTTs. Moreover, the accuracy of measurement results can be maintained considerably well while the number of packets in a measurement is reduced by approximately 90% in comparison with existing methods.

1 Introduction

Network measurement techniques have received a great deal attention, and numerous measurement tools have been developed, as reported in [1,2,3,4,5,6,7,8,9,10]. These tools observe and/or monitor network characteristics such as physical link bandwidth [5,6,7,8], available bandwidth [1,2,3], delay [7], loss [9] and topology of the network [10]. The observed results are often used for network trouble-shooting, isolation of fault location, and network provisioning [11]. Measurement techniques can be categorized into *active* and *passive* approaches. The active approaches [1,2,3,4,7,8,9,10] inject test packets into the network, and utilize the feedback information to derive the measurement results. The passive approaches [5,6] do not use extra packets, but rather monitor packets traversing a router interface or a link.

The population of Internet users and network diversity grow rapidly, and various types of service oriented networks, called *service overlay networks*, are emerging. Such networks include peer-to-peer (P2P) networks, Grid computing networks, Content Delivery/Distribution Networks (CDNs), and IP-VPNs. These networks are upper-layer networks that provide special-purpose services that are built upon a lower-layer network, i.e., the IP network. Therefore, in order to improve the performance of these service overlay networks, accurate and rapid information concerning the total and remaining resources of the lower IP network is important when a data transfer request for a particular service is initiated.

For example, in a CDN service such as Akamai [12] or Exodus [13], measurement techniques can be used to estimate the available bandwidth and regulate the sending rate of transmissions for Web prefetching [14] in order to avoid degrading the performance of other traffic. The network measurement technology can also be used to realize adaptive control mechanisms in various service overlay networks, including the following examples:

- In P2P networks, when the resource discovery mechanism finds multiple peers having the requested contents, the measurement results help to determine from which peer the contents are transmitted
- In data grid networks, when multiple sites have the same data, the measurement results help to determine from which site data will be copied or read.

However, we cannot directly employ existing measurement tools when we measure network characteristics on a service overlay network. One reason for this is that we want to avoid utilizing the test packets used by active measurement approaches for this measurement. This is because using the test packets would degrade the performance of other traffic when the measurements repeat continuously. Another reason is that we want to obtain the latest, and hence the most accurate, information about network characteristics as quickly as possible, because the volume of IP network traffic fluctuates greatly. However, the existing measurement techniques require a long time to obtain one measurement result.

Based on the considerations mentioned above, we are now developing a new network measurement technique to resolve these problems and to improve the quality of service overlay networks. In particular, we focus on a method to measure the *available bandwidth* for a path between two endhosts. Although our method is based on existing measurement approaches, we do not use extra packets for the measurement. Instead, the measurement is performed by collecting the information about the network characteristics obtained from the packets which a regular TCP connection transmits in providing a particular service (we call this approach *Inline Network Measurement*). The method is a sender-based measurement method; it does not require any change in the TCP receiver.

The idea of using packets in TCP for network measurements has attracted a lots of studies [9,15,16,17]. The most advantage of using TCP is that the measurement tool is able to work on a single host. Sting [9], a sender-side only modification of the TCP stack, takes advantage of TCP behavior to measure packet loss rate. In TCP Vegas [18] and TCP Nice [19], the sender uses RTT values to estimate the bandwidth available for the connection. TCP Westwood [17] and a study by Hoe [16] are the most similar to our approach. In these studies, the sender exploits ACK arrival intervals for bandwidth estimation, and decides the value of *ssthresh* according to the estimated values.

In the existing inline network measurement methods, there is a lack of validity in bandwidth measurement algorithms. The measurement algorithms in [16,17] are good in terms of simplicity but, as a result, fail to deliver good measurement results. A study in [20] mentions that the *packet pair* technique used in [16] is not suitable for estimating available bandwidth. Moreover, the packet pair techniques are evaluated in [15] and appear to yield bad effect to the connections in some cases, "due to an inability to form an estimate or overestimating". The same problem can be seen in TCP Westwood [17]. The measurement algorithm in TCP Westwood is similar to the *packet train* technique first proposed in [1]. The technique overestimates and leads to starvation and fairness disruption [21]. To mitigate the high values of the measurement results of TCP Westwood, some low-pass filters have been developed [22] . The filters successfully adjust the measurement results but considerably slow down the measurement speed. According to the experiment results given in [22], TCP Westwood, with filters, cannot give a good measurement result within longer than 40RTTs since the connection starts.

In this paper, therefore, we propose a valid measurement algorithm which becomes the fundamental mechanism of our inline network measurement method. The algorithm gives measurement results periodically; the algorithm reports an estimated value of available bandwidth every 2-4 RTTs (RoundTrip Times) so as to rapidly reflect changes in the IP network. The basic concept behind realizing such a rapid measurement is limiting the measurement range of the bandwidth by using statistical information of previous estimated results, rather than searching from 0 Mbps to the upper limit of the physical bandwidth, as in existing algorithms. By this mechanism, the proposed algorithm requires a rather small number of packets and a short time for measurement as compared to existing algorithms. Consequently, even when the available bandwidth of the path changes dramatically, the proposed algorithm can show the change after only a few measurements. The simulation results show that the proposed measurement algorithm can provide accurate results quickly and periodically, while using a rather small number of probe packets. In addition, we discuss the problems of applying the proposed measurement algorithm to an active TCP connection.

2 Inline Network Measurement in IP-Based Service Overlay Networks

2.1 Requirements

In accordance with the above discussion, we consider the following factors to be the requirements of the measurement algorithm of inline network measurement:

– Small number of packets used
  Because our method uses TCP packets for the measurement, there is a restriction on the number of packets available for transmission at any one time. This is because of the TCP window size. Since the TCP window size is relatively small and changes dynamically, the measurement algorithm should use as small a number of packets as possible.
– No effect on other traffic on the network
  Since the goal of measurement is to improve the quality of services of the service overlay network, the measurement should not affect either the traffic of the supported services itself or the external traffic. The measurement may adversely affect the

network in two ways: by sending numerous probe packets and by sending probe packets at a high rate.
- Providing results continuously
  Since the characteristics of the IP network changes constantly and dynamically, measurement should provide periodic estimation results. Furthermore, the interval should be as small as possible in order to provide an accurate depiction of the rapidly network change.
- Providing results quickly
  The measurement should be performed quickly in order to obtain up-to-date information of the IP network. In the proposed method, we therefore assign a higher priority to measurement speed than to measurement accuracy.

2.2 Existing Network Measurement Methods

As mentioned in Section 1, the existing measurement methods can be divided into two groups: passive measurement methods and active measurement methods. The passive methods, represented by SPAND [5] and Nettimer [6], observe passing traffic at some certain points in the network and use the monitored information to obtain the measurement results. These approaches require quite a long time to gather information for accurate measurement results and many measurements are necessary in order to estimate the characteristics of the end-to-end path. Furthermore, passive approaches cannot provide high-accuracy measurement results because the available information is very limited.

On the other hand, the active measurement methods inject probe packets into the network and collect the feedback information from monitored results including transmission delay, packet arrival-interval time, packet loss ratio and so on. Therefore, we can expect a higher accuracy of measurement results in an end-to-end fashion than is possible by passive methods. Cprobe [1], Topp [2], and Pathload [3], are representative tools to measure the available bandwidth of the network path between two endhosts. These algorithms work on endhosts and require no change inside the network, so that they seem suitable for application to measurement in service overlay networks. However, these algorithms also have fundamental disadvantages. One is that many probe packets are sent at a high transmission rate. For instance, Topp sends 5000 packets to obtain only one measurement, and Cprobe injects 100-200 probe packets at the physical bandwidth speed of the link connected to the sender host. The probe traffic can affect other traffic along the path, for example by degrading traffic throughput and increasing the packet loss ratio and packet transmission delay. Existing active measurement algorithms also require a long time to obtain one measurement result (for example, 50-100 RTTs are necessary to obtain one estimation value for the available bandwidth in Topp and Pathload). Long-term measurement can provide an accurate result but cannot follow the dynamic changes on the IP network.

3 Proposed Measurement Algorithm

Figure 1 shows an outline of the proposed measurement algorithm. A sender host transmits measurement packets to a receiver host, which immediately sends received packets

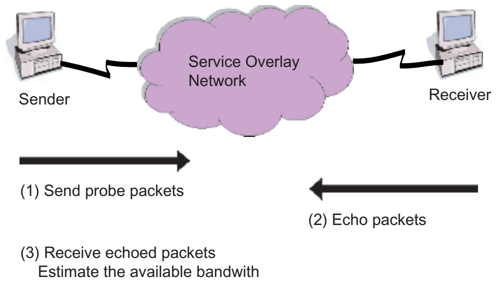


Fig. 1. Outline of proposed measurement algorithm

back to the sender host. The sender then estimates the available bandwidth of the path using the arrival intervals of the echoed packets.

In every measurement, we use a *search range* to find the value of the available bandwidth. Search range I= (B_l, B_u) is a range of bandwidth which is expected to include the current value of the available bandwidth. The proposed measurement algorithm searches for the available bandwidth only within the given search range. The minimum value of B_l, the lower bound of the search range, is 0, and the maximum value of B_u, the upper bound, is equal to the physical bandwidth of the link directly connected to the sender host. By introducing the search range, we can avoid sending probe packets at an extremely high rate, which seriously affects other traffic. We can also keep the number of probe packets for the measurement quite small. As discussed later herein, even when the value of the available bandwidth does not exist within the search range, we can find the correct value in a few measurements. The following are the steps of the proposed algorithm for one measurement of the available bandwidth A:

1. Set the initial search range.
2. Divide the search range into multiple sub-ranges.
3. Inject a packet stream into the network for each sub-range and check the increasing trend of the packet inter-arrival times of the received stream.
4. Find a sub-range which is expected to include the correct value of the available bandwidth using the increasing trends of sent streams.
5. Calculate the available bandwidth by means of linear regression analysis for the chosen sub-range.
6. Create a new search range and return to Step 2.

A packet stream is a group of packets sent at one time for the measurement. In what follows, we explain in detail the algorithm by which to implement the above steps.

1. Set initial search range.
   We first send a packet stream according to the Cprobe algorithm [1] to find a very rough estimation of the available bandwidth. We set the search range to $(A_{cprobe}/2, A_{cprobe})$, where A_{cprobe} is the result of the Cprobe test.
2. Divide the search range.

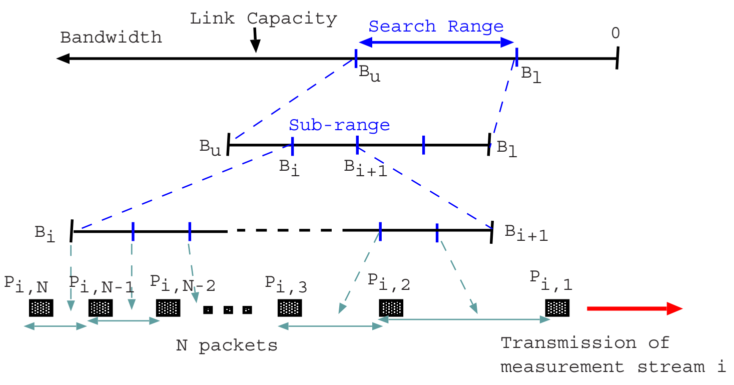


Fig. 2. Relationship of search range, sub-ranges, streams, and probe packets

We divide the search range into k sub-ranges $\mathrm{I}_i = (B_{i+1}, B_i)$ $(i = 1, 2..k)$. All sub-ranges have the identical width of the bandwidth. That is,

$$B_i = B_u - \frac{B_u - B_l}{k}(i-1) \ \ (i = 1, ..., k+1)$$

As k increases, the results of Steps 4 and 6 become more accurate, because the width of each sub-range becomes smaller. However, a larger number of packet streams is required, which results in an increase in the number of used packets and the measurement time.

3. Send packet streams and check increasing trend.
For each of k sub-ranges, a packet stream i $(i = 1...k)$ is sent. The transmission rates of the stream's packets vary to cover the bandwidth range of the sub-range. We denote the j-th packet of the packet stream i as $\mathrm{P}_{i,j}$ $(1 \leq j \leq N$, where N is the number of packets in a stream) and the time at which $\mathrm{P}_{i,j}$ is sent from the sender host as $S_{i,j}$, where $S_{i,1} = 0$. Then $S_{i,j}$ $(j = 2..N)$ is set so that the following equation is satisfied:

$$\frac{M}{S_{i,j} - S_{i,j-1}} = B_{i+1} + \frac{B_i - B_{i+1}}{N-1}(j-1)$$

where M is the size of the probe packet. Figure 2 shows the relationship between the search range, the sub-ranges and the packet streams. In the proposed algorithm, packets in a stream are transmitted with different intervals, for this reason the measurement result may not be as accurate as the Pathload algorithm [3], in which all packets in a stream are sent with identical intervals. However, the proposed algorithm can check a wide range of bandwidth with one stream, whereas the Pathload checks only one value of the bandwidth with one stream. This reduces the number of probe packets and the time required for measurement. By this mechanism, the measurement speed is improved at the expense of measurement accuracy, as described in Subsection 2.1.

We then observe $R_{i,j}$, the time the packet $\mathrm{P}_{i,j}$ arrives at the sender host, where ($R_{i,1} = 0$). We then check if an increasing trend exists in the packet arrival intervals $(R_{i,j} - R_{i,j-1})$ $(2 \leq j \leq N)$ according to the algorithm used in [3]. As explained in [3], the increasing trend of a stream indicates that the transmission rate of the stream is larger than the current available bandwidth of the network path.
Let T_i be the increasing trend of stream i as follows:

$$T_i = \begin{cases} 1 & \text{increasing trend in stream } i \\ -1 & \text{no increasing trend in stream } i \\ 0 & \text{unable to determine the existence of an increasing trend in stream } i \end{cases}$$

As i increases, the rate of stream i decreases. Therefore, T_i is expected to be 1 when i is sufficiently small. On the other hand, when i becomes large, T_i is expected to become -1. Therefore, when neither of the successive streams m or $m+1$ have an increasing trend ($T_m = T_{m+1} = -1$), the remaining streams are expected not to have increasing trends ($T_i = -1$ for $m+2 \leq i \leq k$). Therefore, we stop sending the remaining streams in order to speed up the measurement.

4. Choose a sub-range.
Based on the increasing trends of all streams, we choose a sub-range which is most likely to include the correct value of the available bandwidth. First, we find the value of a $(0 \leq a \leq k+1)$, which maximizes $(\sum_{j=0}^{a} T_j - \sum_{j=a+1}^{k} T_j)$. If $1 \leq a \leq k$, we determine the sub-range I_a is the most likely candidate of the sub-range which includes the available bandwidth value. That is, as a result of the above calculation, I_a indicates the middle of streams which have increasing trends and those which do not. If $a = 0$ or $a = k+1$, on the other hand, the algorithm decides that the available bandwidth does not exist in the search range (B_l, B_u). We determine that the available bandwidth is larger than the upper bound of the search range when $a = 0$, and that when $a = k+1$ the available bandwidth is smaller than the lower bound of the search range.
In this way, we find the sub-range which is expected to include the available bandwidth according to the increasing trends of the packet streams. We avoid using the values of packet receiving intervals like TOPP or Cprobe because their use may lead to serious estimation errors when the transmission rate of probe packets is much larger than the available bandwidth [3,20].
5. Calculate the available bandwidth.
We then derive the available bandwidth A from the sub-range I_a chosen by Step 4. We first determine the transmission rate and the arrival rate of the packet $\mathrm{P}_{a,j}$ ($j = 2...N$) as $\frac{M}{S_{a,j}-S_{a,j-1}}$, $\frac{M}{R_{a,j}-R_{a,j-1}}$, respectively. We then approximate the relationship between the transmission rate and the arrival rate as two straight lines using the linear regression method, as shown in Figure 3. Since we determine that the sub-range I_a includes the available bandwidth, the slope of line (i) which consists of small transmission rates is nearly 1 (the transmission rate and the arrival rate are almost equal), and the slope of line (ii) which consists of larger transmission rates is smaller than 1 (the arrival rate is smaller than the transmission rate). Therefore, we determine that the highest transmission rate in line (i) is the value of the available bandwidth.

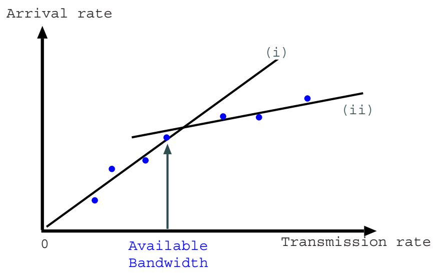


Fig. 3. Finding the available bandwidth within the sub-range

On the other hand, when we have determined that the available bandwidth value does not exist in the search range (B_l, B_u) in Step 4, we temporarily set the value of available bandwidth as follows:

$$A = \begin{cases} B_l & a = 0 \\ B_u & a = k + 1 \end{cases}$$

6. Create a new search range.
When we have found the value of the available bandwidth from a sub-range I_a in Step 5, we accumulate the value as the latest statistical data of the available bandwidth. The next search range (B_l', B_u') is calculated as follows:

$$(B_l', B_u') = \left(A - max\left(1.96\frac{S}{\sqrt{q}}, \frac{B_m}{2}\right), A + max\left(1.96\frac{S}{\sqrt{q}}, \frac{B_m}{2}\right)\right)$$

where S is the variance of stored values of the available bandwidth and q is the number of these values. Therefore, we use the 95% confidential interval of the stored data as the width of the next search range, and the current available bandwidth is used as the center of the search range. B_m is the lower bound of the width of the search range, which is used to prevent the range from being too small. When no accumulated data exists (when the measurement has just started or just after the accumulated data is discarded), we use the search range of the previous measurement.
On the other hand, when we can not find the available bandwidth within the search range, it is possible to consider that the network status has changed greatly. Therefore, we discard the accumulated data because this data becomes unreliable as statistical data. In this case, the next search range (B_l', B_u') is set as follows:

$$B_l' = \begin{cases} B_l & a = 0 \\ B_l - \frac{B_u - B_l}{2} & a = k + 1 \end{cases}$$

$$B_u' = \begin{cases} B_u + \frac{B_u - B_l}{2} & a = 0 \\ B_u & a = k + 1 \end{cases}$$

This modification of the search range is performed in an attempt to widen the search range in the possible direction of the change of the available bandwidth.

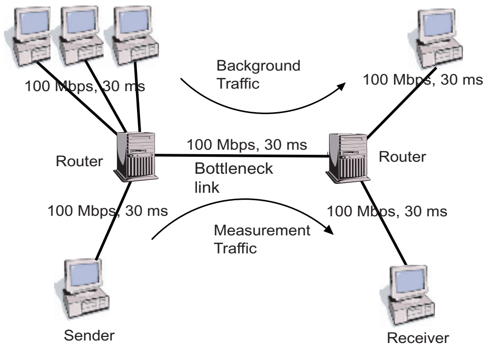


Fig. 4. Network model for simulation experiments

By this statistical mechanism, we expect the measurement algorithm to behave as follows: when the available bandwidth does not change greatly over a period of time, the search range becomes smaller and more accurate measurement can be obtained. On the other hand, when the available bandwidth varies greatly, the search range becomes large and the measurement can be restarted from the rough estimation. That is, the proposed algorithm can give a very accurate estimation of the available bandwidth when the network is stable, and a rough but rapid estimate can be obtained when the network status changes.

4 Simulation Results

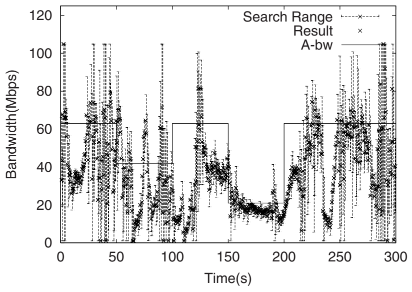
This section shows some simulation results in ns [23] and validates the measurement algorithm proposed in Section 3. Figure 4 shows the network model used in the simulation. A sender host connects to a receiver host through a bottleneck link. The capacity of the bottleneck link is 100 Mbps and the propagation delay is 30 msec. All of the links from the endhosts to the routers have a 100-Mbps bandwidth and a 30-msec propagation delay.

There is background traffic generated by endhosts connecting to the routers. The background traffic is made up of UDP packet flows, in which various packet sizes are used according to the monitored results reported in [24]. The correct value of the available bandwidth of the bottleneck link is calculated as:

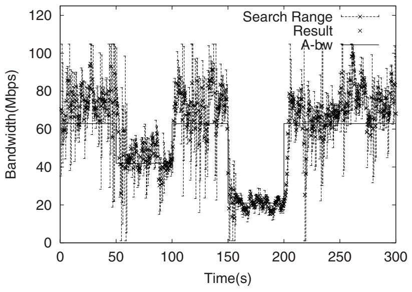
$$Bottleneck\ link\ capacity - Total\ rate\ of\ background\ traffic$$

We make the available bandwidth on the bottleneck link fluctuate by changing background traffic rates.

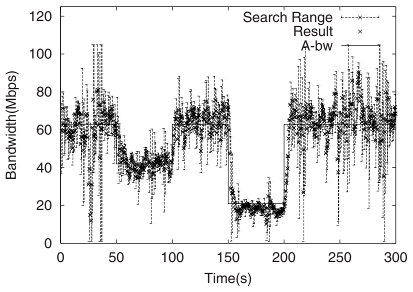
The sender host sends probe packets to the receiver host and the receiver host echoes the packets to the sender. The sender, using the algorithm proposed in Section 3, measures the available bandwidth of the path between the two hosts. In this situation, the result corresponds to the available bandwidth of the bottleneck link between the routers.



(a) N=3



(b) N=5



(c) N=8

Fig. 5. Simulation results

The number of sub-range k, which a search range is divided into, is decided according to the width of the search range and the latest result of the measured available bandwidth, A_{prev};

$$k = \begin{cases} 2 & (0 \leq \frac{B_u - B_l}{A_{prev}} < 0.15) \\ 3 & (0.15 \leq \frac{B_u - B_l}{A_{prev}} < 0.2) \\ 4 & (0.2 \leq \frac{B_u - B_l}{A_{prev}}) \end{cases}$$

B_m, the lower bound of the width of search ranges, is set to 10% of A_{prev}. The probe packet size is 1500 Bytes.

Figure 5 shows the measurement results of the available bandwidth and the search ranges for a simulation time of 300 sec. During the simulation, the background traffic is changed so that the available bandwidth of the bottleneck link is 60 Mbps from 0 sec to 50 sec, 40 Mbps from 50 sec to 100 sec, 60 Mbps from 100 sec to 150 sec, 20 Mbps from 150 sec to 200 sec and 60 Mbps from 200 sec to 300 sec. We also plot the correct values of the available bandwidth in all figures. Figures 5(a)-5(c) show the results when the number of the probe packets in a stream (N) is 3, 5 and 8, respectively. These figures indicate that when N is 3, the measurement results are far from the correct values. When N becomes larger than 5, on the other hand, the estimation result accuracy increases. The proposed measurement algorithm can determine the available bandwidth rapidly, even when the available bandwidth changes suddenly. When N is very small, we can not determine the increasing trend of the streams correctly in Step 3 in the proposed algorithm, which leads to the incorrect choice of subrange in Step 4. Although the accuracy of measurement results increases as N is increased from 5 to 8, since we place a higher priority on measurement speed than on measurement accuracy, as described in Subsection 2.1, $N = 5$ is judged to be the better setting. Actually, finding a suitable value of N is a difficult problem because selection depends on many factors, such as the available bandwidth, the bandwidth size of changes of the available bandwidth and the background traffic. A solution to this problem will require further study. From these simulation results, we can conclude that the proposed algorithm can quickly estimate the available bandwidth, independent of the degree of change in available bandwidth.

5 Problems with Applying the Proposed Algorithm to an Active TCP Connection

In TCP, the TCP sender sends data packets to the receiver and the receiver sends corresponding ACK packets back to the sender. Therefore, we can apply the measurement algorithm in Section 3 to TCP by considering data and ACK packets as probe packets. Due to various characteristics and mechanisms of TCP, however, there are some problems when applying the proposed measurement algorithm.

- Window size
  In a TCP connection, the number of packets which can be transmitted at one time is limited by the window size. The maximum number of packets sent for one measurement stream, therefore, is also limited by the window size. This limitation seriously affects the measurement algorithm, especially when the TCP has a small window size. Moreover, the TCP window size always fluctuates greatly due to the congestion control mechanism of TCP, which also causes difficulty in choosing the optimal number of packets to send at one time for the measurement.
  In order to overcome the problem, the measurement algorithm should be able to dynamically adapt to the changes of the window size of TCP as follows. The measurement algorithm should not attempt to send a measurement stream when the current window size is smaller than the number of packets required for the stream. On the other hand, when the window size is sufficiently large, the algorithm should create as many measurement streams as is allowed by the window size in order to enhance the measurement speed.
- Degrading TCP transmission speed
  The TCP sender generally transmits data packets as soon as possible when it receives an ACK packet from the receiver. However, the measurement algorithm requires multiple packets to build up a measurement stream. Therefore, it is necessary for these packets to be stored briefly before transmission, which causes a delay in packet transmission.

We can avoid degrading TCP transmission speed, caused by storing data packets, by setting an appropriate timeout value for stopping the creation of a stream. There is a trade-off relationship between the speed of the measurement and the TCP transmission speed in choosing the timer length. That is, if the timer length is long, the measurement algorithm can create more measurement streams, so the measurement can be performed quickly. However, since numerous TCP data packets are stored at the intermediate buffer for a long time, TCP transmission speed may be deteriorated. Moreover, the long packet delay may lead to TCP timeout events.

– Behavior of TCP receiver
The proposed measurement algorithm expects the TCP receiver to send an ACK packet back to the sender immediately when a data packet arrives. However, some TCP receivers utilize the delayed ACK option [25], in which the TCP receiver does not deliver an ACK packet for each data packet. In this case, the measurement algorithm may not work properly. In this case, Step 3 of the proposed algorithm should be changed so that intervals of three packets are used rather than intervals of two packets. That is, we calculate the arrival intervals $(S_{i,2j'+2} - S_{i,2j'})$ $(1 \leq j' \leq \lfloor N/2 \rfloor)$ for the probing packets in stream i in order to check its increasing trend. This has the same effect as halving the number of packets in one stream, which results in the degradation of measurement accuracy. Therefore, the number of packets in a stream should be increased appropriately.

6 Conclusion

In this paper, we introduced a measurement method for the available bandwidth of a path between two endhosts using an active TCP connection. We first constructed a new measurement algorithm which uses quite a small number of probe packets and yet provides periodic measurement results quickly. We presented a number of simulation results in order to validate the effectiveness of the proposed algorithm. We then discussed problems when applying the proposed algorithm to an active TCP connection and presented possible solutions. The proposed methods were evaluated by simulation experiments and we have observed that the proposed measurement algorithm can successfully measure the available bandwidth. As future research, we will integrate the measurement algorithm into TCP and validate its effectiveness.

References

1. R. L. Carter and M. E. Crovella, "Measuring bottleneck link speed in packet-switched networks," Tech. Rep. TR-96-006, Boston University Computer Science Department, Mar. 1999.
2. B. Melander, M. Bjorkman, and P. Gunningberg, "A new end-to-end probing and analysis method for estimating bandwidth bottlenecks," in *Proceedings of IEEE GLOBECOM 2000*, Nov. 2000.
3. 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.
4. M. Allman, "Measuring end-to-end bulk transfer capacity," in *Proceedings of ACM SIGCOMM Internet Measurement Workshop 2001*, ACM SIGCOMM, November 2001.

5. S. Seshan, M. Stemm, and R. H. Katz, "SPAND: Shared passive network performance discovery," in *Proceedings of 1st Usenix Symposium on Internet Technologies and Systems (USITS '97)*, pp. 135–146, Dec. 1997.
6. 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.
7. V. Jacobson, "pathchar-A Tool to infer characteristics of Internet paths," *http://www.caida.org/tools/utilities/others/pathchar/*, 1997.
8. A. B. Downey, "Using pathchar to estimate internet link characteristics," *Measurement and Modeling of Computer Systems*, pp. 222–223, 1999.
9. S. Savage, "Sting: A TCP-based network measurement tool," in *Proceedings of USITS '99*, Oct. 1999.
10. B. Hu, A Daniel and P. David, "Topology discovery by active probing," in *Proceedings of the 2002 Symposium on Applications and the Internet (SAINT)*, Jan 2002.
11. K. Matoba, S. Ata, and M. Murata, "Capacity dimensioning based on traffic measurement in the Internet," in *Proceedings of IEEE GLOBECOM 2001*, pp. 2532–2536, Nov. 2001.
12. Akamai Home Page, *http://www.akamai.com/*.
13. Exodus Home Page, *http://www.exodus.com/*.
14. L. Fan, P. Cao, and Q. Jacobson, "Web prefetching between low-bandwidth clients and proxies: Potential and performance," in *Proceedings of ACM SIGMETRICS '99*, May 2000.
15. Mark Allman and Vern Paxson, "On estimating end-to-end network path properties," in *Proceedings of SIGCOMM '99*, pp. 263–274, 1999.
16. J. C. Hoe, "Improving the start-up behavior of a congestion control sheme for TCP," in *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, vol. 26,4, pp. 270–280, ACM Press, 1996.
17. M. Gerla, M. Y. Sanadidi, R. Wang, A. Zanella, C. Casetti and S. Mascolo, "TCP Westwood: Congestion window control using bandwidth estimation," in *Proceedings of IEEE Globecom 2001*, pp. 1698–1702.
18. M. Gerla, Y. Sanadidi, R. Wang, A. Zanella, C. Casetti and S. Mascolo, "TCP Vegas: New techniques for congestion detection and avoidance," in *Proceedings of the SIGCOMM '94 Symposium*, pp. 24–35, Aug. 1994.
19. A. Venkataramani, R. Kokku and M. Dahlin, "TCP-Nice: A mechanism for background transfers," in *Proceedings of the Fifth Symposium on Operating Systems Design and Implementation*, Dec. 2002.
20. C. Dovrolis and D. Moore, "What do packet dispersion techniques measure?," in *Proceedings of IEEE INFOCOM 2001*, pp. 22–26, Apr. 2001.
21. Luigi Alfredo Grieco and Saverio Mascolo, "TCP westwood and easy RED to improve fairness in high-speed networks," in *Proceeding of seventh International workshop on Protocols for High-Speed Networks (PfHSN 2002)*, pp. 130–146, 2002.
22. L. A. Grieco and S. Mascolo, "End-to-end bandwidth estimation for congestion control in packet networks," in *Proceedings of Quality of Service in Multiservice IP Networks, II Int. Workshop, QoS-IP 2003*, Feb. 2003.
23. NS Home Page, *http://www.isi.edu/nsnam/ns/*.
24. NLANR web site, *http://moat.nlanr.net/Datacube/*.
25. R. Wright and R. Stevens,, "TCP/IP Illustrated, Volume 1: The Protocols," *Addison-Wesley Publishing, USA 1994*.