Available Bandwidth Measurement, Implementation and Experiment

Jingsha He, Yingping Lu[†], C. Edward Chow[‡] and Takafumi Chujo
Fujitsu Laboratories of America, Inc.
595 Lawrence Expressway
Sunnyvale, CA 94085

***Abstract* - We describe the algorithm and the implementation and experiment for available bandwidth measurement based on variable speed probing and zoom-in/zoom-out [7]. We show that, compared with other techniques, our algorithm yields better performance with low overhead and fast convergence because it detects the change of traffic trends (through variable speed probing) rather than specific patterns of probing samples. It can also self-adapt to any bandwidth (through zoom-out) and respond to resolution requirement (through zoom-in). Therefore, no prior knowledge about bottleneck bandwidth is required and measurement resolution can be specified to meet application requirements. We focus on the implementation and experiment and present some results and statistics. We are currently conducting more experiment over live networks to gain more experience and to further improve the algorithm.**

I. INTRODUCTION

The available bandwidth of a path between two hosts is an important network parameter for optimizing resource utilization in traffic engineering and for admission control in quality of service. Since available bandwidth is very dynamic that is determined by link capacity and the current traffic volume but necessarily by bottleneck bandwidth although it is capped by it, any practical measurement must keep overhead low while achieving reasonably good results. This is because available bandwidth has to be measured more often, especially during periods of dramatic change, to make the results useful. Consequently, the intervals between successive measurements must be dynamically adjustable to maintain the timeliness and usefulness of the results. This is in contrast to bottleneck bandwidth measurement in which the results can serve a relatively longer time and the specific times for performing the measurement can be chosen, e.g., during periods of low traffic volume.

In this paper, we present a practical available bandwidth measurement algorithm, discuss the implementation and show some experiment results. Compared with previous work, our algorithm yields better results and fast convergence with lower overhead because it relies on the detection of traffic trends (through variable speed probing) rather than specific patterns of probing samples. It also self-adapts to any bandwidth (through zoom-out) and responds to resolution requirement (through zoom-in), which is unique among all measurement techniques. Therefore, our algorithm doesn't require any prior knowledge about bottleneck bandwidth, nor does it need any knowledge about the current traffic. We substantiate our claims through a large number of experiments using a testbed and present the results and statistics obtained from the experiments in the paper. In particular, we compare our results with those of Cprobe [4], which shows that our algorithm performs better with lower overhead.

The rest of the paper is organized as follows. In the next section, we review some previous work related to available bandwidth measurement. In Section III, we briefly describe our algorithm [7] and discuss some implementation issues. In Section IV, we discuss the experiment, present some results and statistics and compare our results with Cprobe [4]. Finally, we conclude this paper in Section V.

II. RELATED WORK

Network measurement has been the subject of numerous studies during the past few years. Bolot [3] analyzed end-to-end packet delay and loss in the Internet and used a phase plot to characterize the phenomenon in a congested network. He showed that, when multiple packets are transmitted at a low speed, the plot for the round trip delay shows random distribution and, at a high speed, forms a distinctive pattern of distribution. The difference can be attributed to congestion when the packets are transmitted at high speed. NEPRI [1] is a bandwidth measurement tool based on this theory in which a phase plot is drawn after each round of probing to detect the pattern that corresponds to network congestion. Based on the current phase plot, a new probing speed is determined for the next round. The process continues until the pattern that corresponds to congestion has formed and the measurement result can be derived. The disadvantage is the high overhead because of the excessive number of packets in each round at a fixed speed to determine the congestion. In addition, chasing the movement of available bandwidth from one probing round to another could result in a large number of rounds before the measurement result can be derived.

Pathchar is a measurement tool that can be used to measure the bottleneck bandwidth of the links along a path [8]. During the measurement, a number of probing rounds with different packet sizes are used towards each intermediate node in the path until the desired destination is reached. Downey [6] did extensive experiment with it and showed that Pathchar could yield reasonably good results. However, because it is developed for measuring link capacity, the overhead is

[†] This author's address is Department of Computer Science, University of Minnesota, Minneapolis, MN 55455.
[‡] This author's address is Department of Computer Science, University of Colorado, Colorado Springs, CO 80918.

extremely high, which is prohibitive for available bandwidth, and cannot be directly applied to available bandwidth in its present form. Similarly, the method by Dovrolis, et al. [5] is primarily for bottleneck bandwidth measurement and cannot be directly applied to available bandwidth due to high overhead and long measurement time.

In Bprobe and Cprobe [4], multiple rounds of probing are used to calculate the bottleneck and the available bandwidth, respectively. A result is computed based on all the probing rounds at different speeds and with different packet sizes. Consequently, measurement overhead is very high. Furthermore, always probing at the bottleneck speed in Cprobe incurs unnecessarily high overhead, which would cause severe packet delay and loss to all traffic. Also, Cprobe requires the knowledge about the bottleneck bandwidth, which requires accurate bottleneck bandwidth results before the measurement and, consequently, may cause unnecessary overhead for bottleneck bandwidth measurement to ensure the validity of its results. The method by Banerjee [2] for the estimation of available capacity by imposing a delay bound also suffers the problem high overhead and long latency in the derivation of a measurement result.

Various studies and experiments [6] showed that, even with high overhead, it is very hard to produce satisfactory results, especially for available bandwidth due to its dynamic nature. This could also be caused by excessive probing traffic that would cause sever packet delay and loss that affect all types of traffic in the network, not to mention the impracticality due to their disturbing consequences to the network. Experience shows that, for available bandwidth measurement, high overhead resulting from a large number of probing rounds and excessive probing traffic don't necessarily improve measurement results. Rather, algorithms that use a fewer probing rounds while being able to quickly detect the network phenomenon caused by congestion is both desirable and practical. In this paper, we present such an algorithm that can achieve the objectives of both low overhead and quick divergence, which makes it advantageous to and more practical over all the previous work.

III. Measurement Algorithm and Implementation

We briefly describe the measurement algorithm and some of the implementation issues in this section.

A. The Algorithm

The algorithm for available bandwidth measurement was presented in details in [7] with some preliminary simulation results. In the algorithm, we use the active probing approach with the techniques of variable speed probing and zoom-in/zoom-out.

For variable speed probing, we send a series of probing packets at variable speeds, from low to high. Since they are sent at different speeds, the first packet and the second to the last define the range of the bandwidths required to carry the probing traffic through the path without causing congestion. Here, the bandwidth requirement of each packet is determined by the size of the packet and the time interval between it and the next one. Therefore, the probing packets will impose increasingly higher bandwidth requirements on the path due to the increases in speed (or decreases in packet intervals). When the bandwidth requirement of a packet exceeds the available bandwidth, congestion starts to occur so that we can observe different RTTs (round trip delays) before and after the congestion. Based on this phenomenon, we can derive the available bandwidth by detecting the point where congestion starts to occur based on the RTTs. This congestion point then provides us with the needed information for available bandwidth estimation because, before the congestion, all probing packets should have about the same RTTs and, after the congestion, although the RTTs may be different, the general trend will be consistent and the RTTs should be larger than those before the congestion due to additional queuing delays. Note that, in our illustration, we use the variable speed scheme although the variable packet size is equally effective in getting the different bandwidth requirements by the probing packets on the measured path. We could also use a combination of variable speeds and packet sizes to get the desired bandwidth requirements more flexibly and efficiently.

In general, the bandwidth requirement of a probing packet is determined by the size of the packet S and the time interval t between it and the packet that immediately follows it:

$$S/t.$$

Assuming a fixed packet size S, if the time interval between packets P_i and P_{i+1} is t_i, $1 \le i < n$ and $t_i > t_{i+1}$, the bandwidth requirement of packet P_i is:

$$S/t_i.$$

Assuming n probing packets $\{P_1, P_2, \ldots, P_n\}$ of size S but decreasing time intervals, the bandwidth requirement of the packets on the measured path increases as the packets travel to the server and return to the client or the probing agent. After collecting the RTTs for the probing packets by the probing agent, we use curve matching between the one for sending the probing packets (the sending curve) and the one for receiving the acknowledgement packets (the receiving curve) to detect congestion and derive available bandwidth. Please refer to [7] for details about the curve matching.

One way to calculate the time intervals is to assume that the intervals decrease linearly in equal amount in time. Then, with S, n and (B_L, B_H), $B_L > 0$ and $B_H > B_L$, we can compute the time intervals using the formula:

$$t_i = \frac{S}{n-2}\left(\frac{n-i-1}{B_L} + \frac{i-1}{B_H}\right) \qquad 1 \le i \le n-1.$$

We can assume other relationships between the time intervals and use the same technique to compute them. One

such a method is to assume that the intervals decrease linearly in equal amount in bandwidth. We are also studying other probing packet patterns and distributions that can yield better, or optimized, performance in terms of measurement results and overheads as an optimization problem.

The zoom-in technique is used when the measurement detects the congestion but the result doesn't meet the required resolution. Therefore, if (B_L, B_H) is set too large, the resolution will be poorer with a fixed number of probing packets. To improve the resolution, we use zoom-in to initiate an additional round of probing with a smaller (B_L, B_H) around where the congestion occurred. With the new (B_L, B_H), a new set of time intervals t_i, $1 \le i \le n-1$, will be computed. This process will continue until a measurement result is obtained that meets the specified resolution. That is, our algorithm will automatically determine whether zoom-in is needed after each round of probing. The number of probing rounds is then determined by the number of probing packets n, the packet size S, the resolution requirement, the initial (B_L, B_H) and the fluctuation of the traffic. The zoom-in process is a necessary step for the improvement of the measurement result.

Zoom-out is the opposite of zoom-in and is used for the detection of congestion when the current round cannot identify one. This is possible when the probing speed is too slow (curves with total matching) or too fast (curves without any matching). Since we don't have any knowledge about link capacity, we don't know what the highest bandwidth should be set. Even if we know, we don't want to unconditionally probe at the highest bandwidth for its disruptive consequence. Therefore, this technique will try to dynamically expand the bandwidth range to automatically adapt to any bandwidth in a gentle manner. Zoom-out enlarges the bandwidth range (B_L, B_H). It could also move the bandwidth range downward if the previous probing is too fast or upward if too slow. The enlargement and movement of the bandwidth range could also be used together. Similar to zoom-in, a single zoom-out may still fall short. Therefore, the algorithm would automatically determine if zoom-out is needed and, if yes, an additional probing round is initiated with a new bandwidth range and corresponding time intervals. The zoom-out process is a necessary step for the detection of congestion.

Note that zoom-in and zoom-out will be interchangeably invoked and there is not a general rule regarding which one would be used first and in what order; their invocation is only determined by the current measurement result. It may also be affected by the volatility of the traffic. Depending on the starting bandwidth range and the selection for the next round, the number of rounds of zoom-in and zoom-out could differ significantly.

The algorithm that combines the techniques of variable speed probing and zoom-in/zoom-out is thus summarized as follows:

(1) Using n probing packets of size S and picking a bandwidth range based on past measurement results or other knowledge or heuristics, invoke the basic variable speed probing technique and curve matching to detect congestion. The selection of the initial bandwidth range is not essential but a better selection could result in fewer rounds of probing and, consequently, lower overhead.

(2) If congestion is detected, calculate the result and determine its resolution. If the resolution requirement is met, the algorithm concludes and the result is reported as the measured available bandwidth. In our experiment, the available bandwidth is the average of the bandwidth requirements of the probing packets immediately before and after the congestion. The available bandwidth could also be computed based on more than one packet around the area of congestion to neutralize the effect of traffic volatility and noise during the measurement.

(3) If congestion is detected but the resolution requirement is not met, the zoom-in procedure is invoked in which a smaller bandwidth range (B_L, B_H) is used for the next round. The algorithm continues with (1).

(4) If congestion is not detected, the zoom-out procedure is invoked in which a larger or a different bandwidth range (B_L, B_H) is picked through the examination of the current measurement. The algorithm continues with (1).

It is therefore clear that higher overhead could result from higher resolution requirement and from the flexibility of automatic adaptation of the algorithm to any bandwidth, both of which are desirable features in any measurement algorithm. Without the resolution requirement, zoom-in could be avoided. So could zoom-out without self-adaptation to any bandwidth. No previous work can achieve the same functionality and flexibility as the zoom-in and zoom-out. In terms of completeness and performance, we believe that our algorithm is superior to all the previous work, which we have validated through simulation [7] and experiment that is presented later in the next section.

B. The Implementation

First of all, it may seem to be straightforward to implement the algorithm. However, since the performance of the algorithm relies on accurate timing of the probing packets for detecting congestion and for computing the result, it would be difficult without controlling the behavior of the underlying kernel. One difficulty is that most general-purpose operating systems provide little real-time support. Consequently, it is difficult to ensure that a packet departs at the specified time and is stamped with accurate receiving time. Another difficulty is that, due to the inherent variance nature of network in handling packet forwarding, the real sending curve and receiving curve may not exactly match with each other. Therefore, we have to tolerate certain time variances in the implementation accordingly. In this section, we briefly describe how we deal with the difficulties.

B. 1. Timing in Sending and Receiving Probing Packets

General-purpose operating systems provide little support for real-time operations. Therefore, during the measurement, other processes could share the same CPU and cause uncontrollable context switching. Consequently, accuracy of timing cannot easily be achieved. For example, we found that the difference between ideal and real packet departure times can vary up to 80ms. If packets of 1000 bytes are transmitted at the rate of 10Mbps, the departure gap should be 800us. Therefore, the 80ms time gap caps the departure speed at only 100Kbps.

Fortunately, although the standard Linux is not designed for real-time applications, it provides some real-time support since Kernel 1.2 that we have taken advantage of to serve our purpose. The support includes the dynamic change of the scheduling policies and process priorities. Specifically, we made use of the SCHED_RR among the three scheduling policies offered: SCHED_OTHER for the default time-sharing scheduling while SCHED_RR and SCHED_FIFO for time-critical applications that need more precise control over the execution of processes. In the implementation, we switch the scheduler to SCHED_RR just before the probing and switch it back right after the receipt of the final acknowledgment packet with timeout control. After extensive testing, we found that the timing gap is greatly improved within a few microseconds. Since most of Unix platforms support POSIX API, this implementation can be easily ported to other platforms.

Another factor for inaccurate timing is the conflict between sending and receiving probing packets because both can happen at the same time. Although the most obvious way of the implementation is to create two processes handling sending and receiving separately, if there is only one CPU, context switching between these two processes could affect timing accuracy. To avoid such context switching, we combined the two activities into one process. By using TCP *select* function to get the receiving status, the probing agent will constantly check the packet departure time and the receiving status in turns. If one condition becomes true, the corresponding operation is performed. Extensive testing showed that the timing accuracy has been greatly improved with this solution.

B. 2. Detection of Congestion

Since there are many factors in packet transmission, we cannot always get accurate RTTs, among which are:

(1) Internal transmission. This is generally related to the IP protocol stack. As a packet traverses through the stack, interrupts may occur that causes the current process to be suspended and the CPU preempted to handle the interrupts. This could introduce timing variance.

(2) Conflict with cross-traffic. Even when traffic is light, it is still possible that a probing packet collide with the cross-traffic packets. Consequently, the probing packet will have to wait in a queue until all the packets in front of it are transmitted. This could introduce timing variance.

(3) Server response. ICMP processing in the server may also cause timing variance due to the internal transmission discussed above and due to the time it takes the server to process the packet and return a reply. Some machines may impose a limit on the rate for ICMP processing. Consequently, the probing packets may not be handled consistently. This could introduce timing variance.

Therefore, it can be seen that, due to timing variance, the determination of congestion becomes very challenging. Our solution is to rely on more points in the detection of congestion and in the computation of the result. We call these consecutive points a window and take the median of the RTTs in the window as the window's RTT. More specifically, when detecting the congestion, we put next three points in a window and use the window's RTT in the computation. Then, the next packet to look at in the analysis is the packet after the window. In addition, were more than one congestion point be identified, the final result would be the weighted average of such window's available bandwidth. The weight of each available bandwidth is determined by the duration of that window to the next.

IV. EXPERIMENT AND RESULTS

We set up a testbed to measure the available bandwidth of a dedicated link, i.e., the measured link. With the testbed environment, we are able to control the traffic volume and type over the measured link, apply the algorithm and compare the measurement results with the real available bandwidth numbers to evaluate the performance and conduct various other experiments. The testbed is depicted in Fig. 1 in which, in addition to the probing agent and the server, there are three other components: the traffic generator, the traffic captor and a display. The traffic generator is used to send a specified type and volume of traffic to the server and only the generated traffic and the probing traffic go over the measured link. Thus, the measurement algorithm can be evaluated with

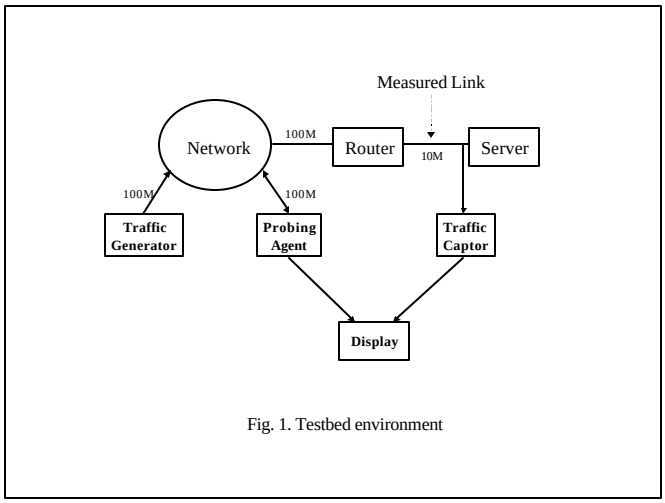
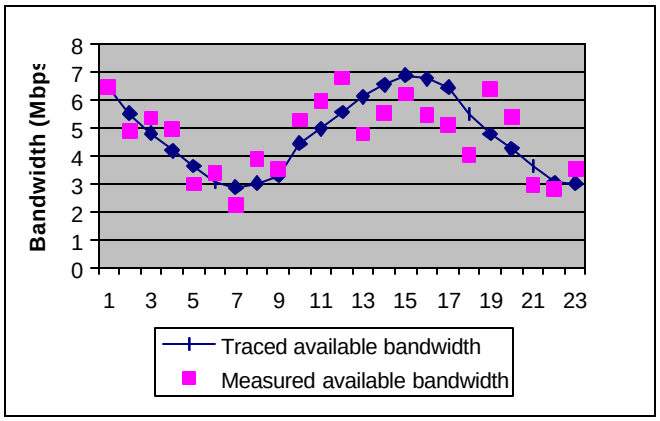


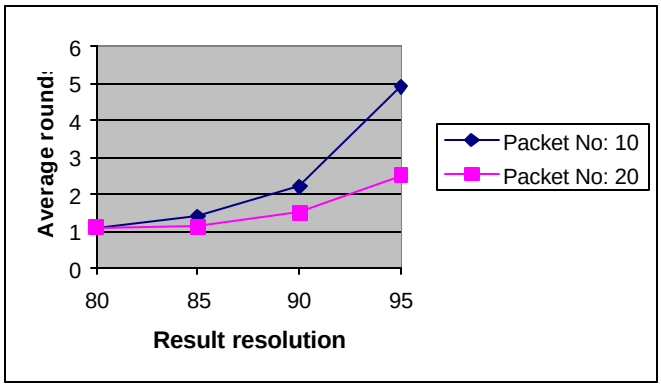
Fig. 1. Testbed environment

different types of traffic. The traffic captor is used to capture all the traffic to the server over the measured link and to calculate the real available bandwidth by subtracting the total traffic captured from the link capacity of the measured link. The results are then sent to the display and plotted along with the measurement results sent from the probing agent for comparison and evaluation.

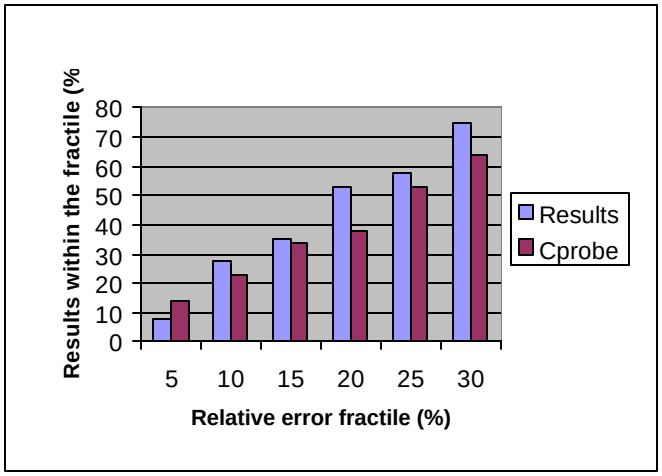
We conducted many experiments using different types of traffic. The following figure shows the traffic type that resembles a sine curve and the measurement results along the curve for the real available bandwidth results in which we used 30 packets with the size of 600 bytes for each packet.



We did the experiment to find the relationship between the overhead in terms of number of probing rounds and measurement resolutions. The following figure shows the relationship for 10 packets and 20 packets, respectively. It also shows that more packets used in each probing round would reduce the number of rounds needed.



Finally, we evaluated our algorithm and compared it with Cprobe [4]. The following figure demonstrates that our algorithm performs better than Cprobe with lower overhead. In terms of measurement results, three quarters, i.e., 75%, of our measurement results are within 30% of the actual available bandwidth values vs. 64% for Cprobe. In terms of overhead, we injected 10K bytes data into the network for probing vs. 24K for Cprobe during the measurement.



V. CONCLUSION

We presented an algorithm for available bandwidth measurement using the techniques of variable speed probing and zoom-in/zoom-out. By systemically collecting and comparing the arrival times of the probing packets, we can detect whether network congestion occurs and compute the measurement results accordingly. The algorithm can self-adapt to any bandwidth and respond to measurement resolution requirements through zoom-out and zoom-in, respectively. We also discussed some implementation issues, presented some experiment results and statistics and compared our results with some previous work. We are currently moving the experiment effort to a live network to gain more experience and to further validate and improve the algorithm and the implementation.

REFERENCES

[1] M. Adachi, et al., "NEPRI: available bandwidth measurement in IP networks," *Proc. ICC'00*, June 2000.

[2] S. Banerjee and A. Agrawala, "Estimating available capacity of a network connection", *Proc. ICON*, Sept. 2000

[3] J. Bolot, "Characterizing end-to-end packet delay and loss in the Internet," *Journal of High Speed Networks*, vol. 2, No. 3, Dec. 1993.

[4] R. Carter and M. Crovella, "Measuring bottleneck link speed in packet-switched networks," *TR BU-CS-96-006*, Boston University, 1996.

[5] C. Dovrolis, et al., "What do packet dispersion techniques measure?" *Proc. Infocom'01*, Jan. 2001.

[6] A. Downey, "Using Pathchar to estimate Internet link characteristics," *Proc. ACM SIGCOMM'99*, 1999.

[7] J. He, et al., "An algorithm for the measurement of available bandwidth," *Proc. IEEE ICN'01*, July 2001.

[8] V. Jacobson, "Pathchar -- a tool to infer characteristics of Internet paths," *Presented at the Mathematical Science Research Institute (MSRI)*, April 1997. Slides available from ftp://ftp.ee.lbl.gov/pathchar/.