

On Fast Estimation of Network Bandwidth

Dulal C. Kar and Randy R. DeLeon
Department of Computing and Mathematical Sciences
Texas A&M University-Corpus Christi
Corpus Christi, TX 78412, U.S.A.

Abstract – Several tools are available to measure the bandwidth of an Internet link. However, these tools are found to be a bit slow since they send a large amount of probe packets to estimate not only the bandwidth of an Internet link, but also aim to find other characteristics of the link. In addition, these tools are not adaptable towards traffic conditions such as congestion. They send only a fixed number of probe packets in all situations, which often yield an inaccurate result on the actual bandwidth itself. In this work, we present a regression-based, iterative technique to filter out queue delays from all measured roundtrip times; this is done in order to estimate the bandwidth of a link along an Internet path. The technique can be used in existing ICMP (Internet Control Message Protocol) protocol based tools or on any tool that sends probe packets to measure roundtrip times to estimate the bandwidth. We test the proposed technique on the links along an Internet path. Our test results show that the proposed scheme is faster than the existing schemes and requires significantly fewer probe packets to achieve similar accuracy.

Keywords: Network bandwidth, bandwidth estimation, roundtrip time, Internet path.

1. Introduction

Network bandwidth is an important resource in regards to the Internet. The knowledge of link bandwidths along an Internet path can help plan an application that delivers data to a client in some systematic, timely manner, which avoids situations that can cause excessive packet loss or delay, particularly for some time-sensitive application. Also, such knowledge can help avoid unnecessary congestion along an Internet path. In recent years, several tools have been released to estimate the bandwidth of Internet links or to detect a bottleneck link along an Internet path [1]-[3][6]. With all measured link bandwidths having been recorded along an Internet path, it is the bottleneck link (the one with the lowest bandwidth) that can be detected. Accordingly, the delivery of data can be planned along the path for time-sensitive data.

Many schemes have been proposed to measure an Internet link bandwidth (also known as link capacity) as well as available bandwidth of an Internet path [1]-[3], [6], [8]-[10]. Recently, several tools that directly allow characterizing of an Internet path for link bandwidths, queue delays, packet losses, and latencies have been reported in literature [1]-[3], [6]. These tools are available for both UNIX or UNIX-like systems. One tool to note is *pathchar* (path characterization) which was developed by Van Jacobson of Lawrence Berkeley National Laboratory in 1997. Van Jacobson is also known to have developed *traceroute*, which is widely used [2]. Following the development of *pathchar*, *pchar* was developed by Bruce Mah in 1999 [3]. Around the same time, *clink* (characterization of a link) was developed by Allen Downey [1]. All these tools make use of the *traceroute-like* algorithm based on the ICMP (Internet Control Message Protocol) protocol. Particularly, these tools can run on a host machine on the Internet and measure the upstream bandwidths of all links along an Internet path. This is achieved by measuring the roundtrip times to all the nodes along the path towards the destination host, and filters the queue delays from the measured roundtrip times. However, the tools are not adaptive to network conditions such as congestion and send a fixed number of probe packets to measure roundtrip times for all situations and all links along an Internet path. In fact, we observe that a fewer number of probe packets are actually needed in order to achieve a similar accuracy as far as bandwidth measurement is concerned.

In this work, we propose an efficient regression-based, iterative algorithm that can be used to estimate the bandwidth of a link along an Internet path. Accuracy of the scheme is found to be in the range of the accuracy of the existing tools. In order to filter out the effect of queue delays from roundtrip times, we propose a new regression-based filtering algorithm to estimate the bandwidth. Compared to the existing techniques listed, the new algorithm sends fewer probe packets in order to find the bandwidth of an Internet link. As a result, it is faster and less intrusive to the

traffic along the Internet path under measurement. Hence the scheme can be used by an application which requires bandwidth information of a path or link to schedule its packet delivery to a destination in an efficient manner.

Section 2 of the paper describes a network model and its parameters for bandwidth estimation. Section 3 outlines, in general, a theoretical framework for bandwidth estimation of a link along an Internet path. Sections 4 and 5 delineate our proposed algorithm and provide some experimental results on a comparative study of the algorithm.

2. Network Model

For our development, we consider a network model similar to the one given in [1] and [3]. Accordingly, the network model for a link connecting two nodes in a path is shown in Fig. 1. Each node has a forwarding engine (FE) that forwards packets in the upstream as well as in the downstream directions along the path. Each node also has two queues, one for packets traveling upstream and the other one for packets traveling downstream. Let us consider a path $P(0, n)$ consisting of nodes N_0, N_1, \dots, N_n and links L_1, L_2, \dots, L_n where each link L_i connects only nodes N_{i-1} and N_i . For convenience as shown in Fig. 1, we use the following notations:

- $p(i)$: Signal propagation delay over link L_i
- $q_u(i)$: Upstream random queue delay at node i
- $q_d(i)$: Downstream random queue delay at node i
- $b_u(i)$: Upstream bandwidth of link L_i
- $b_d(i)$: Downstream bandwidth of link L_i
- $t(i)$: Processing or forwarding delay at node i

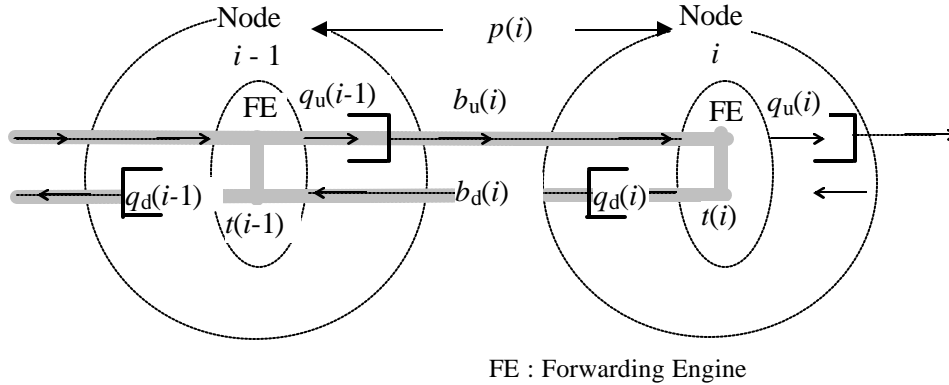


Fig. 1. Network model.

Queue delays at nodes are random and depend on network traffic conditions. Other network parameters can be considered deterministic. We assume that the processing delay is the same for all packets and the propagation delay is the same in both directions. Let us consider a path from node N_0 to node N_i and let us suppose node N_0 sends a packet of size s to node N_i and node N_i returns a packet of size r in response. The roundtrip time, $T_{s,r}(i)$ of this communication as seen by node N_0 is given by [7]

$$T_{s,r}(i) = \sum_{k=1}^i \left(\frac{s}{b_u(k)} + p(k) + q_u(k) + t(k) \right) - q_u(i) + \sum_{k=1}^i \left(\frac{r}{b_d(k)} + p(k) + q_d(k) + t(k) \right) - t(i) \quad (1)$$

After some manipulation, (1) can be written as:

$$T_{s,r}(i) = s \sum_{k=1}^i \left(\frac{1}{b_u(k)} \right) + r \sum_{k=1}^i \left(\frac{1}{b_d(k)} \right) + 2 \sum_{k=1}^i p(k) + 2 \sum_{k=1}^i t(k) - t(i) + \sum_{k=1}^i (q_u(k) + q_d(k)) - q_u(i) \quad (2)$$

Let $C(i)$ represent the total of all propagation and processing delays in (1) for the roundtrip time. That is,

$$C(i) = 2 \sum_{k=1}^i p(k) + 2 \sum_{k=1}^i t(k) - t(i)$$

The propagation delay of a link is fixed regardless of the packet size. Similarly, the packet processing time at a node can be assumed to be fixed regardless of the packet size. Hence $C(i)$ can be considered a constant for the path.

Let $Q(i)$ represent the total of all queue delays for the roundtrip time in (2). That is,

$$Q(i) = \sum_{k=1}^i (q_u(k) + q_d(k)) - q_u(i)$$

However, the queue delay at a node is random and depends on network traffic condition. Finally (2) can be expressed as:

$$T_{s,r}(i) = s \sum_{k=1}^i \left(\frac{1}{b_u(k)} \right) + r \sum_{k=1}^i \left(\frac{1}{b_d(k)} \right) + C(i) + Q(i) \quad (3)$$

In the following, we describe how to measure the bandwidth of a link L_i using (3).

3. Bandwidth Estimation

Let $M_{s,r}(i)$ represent the minimum roundtrip time measured between N_0 and N_i . Evidently $T_{s,r}(i) = M_{s,r}(i)$ when the total queue delay $Q(i)$ is zero. As described later, the total queue delay $Q(i)$ can be filtered out from $T_{s,r}(i)$ to obtain $M_{s,r}(i)$ and hence from (3), $M_{s,r}(i)$ can be expressed as:

$$M_{s,r}(i) = s \sum_{k=1}^i \left(\frac{1}{b_u(k)} \right) + r \sum_{k=1}^i \left(\frac{1}{b_d(k)} \right) + C(i) \quad (4)$$

Since a link bandwidth is constant, from equation (3) it is evident that $M_{s,r}(i)$ is a function of s and r only. However, in a situation in which the destination node responds to a source node with a packet of fixed size, regardless of the size of a probe packet sent by the source node, then $M_{s,r}(i)$ is a linear function of s only. Thus, $M_{s,r}(i)$ can be written as an equation of a straight line which can be given by:

$$M_{s,r}(i) = m(i)s + G(i) \quad (5)$$

where $m(i)$ is the slope of the line as given by:

$$m(i) = \sum_{k=1}^i \frac{1}{b_u(k)} \quad (6)$$

and $G(i)$ is the constant or intercept as given by:

$$G(i) = r \sum_{k=1}^i \left(\frac{1}{b_d(k)} \right) + C(i) \quad (7)$$

An example situation of a fixed sized response packet occurs on an Internet link when the TTL (Time To Live) field of an IP (Internet Protocol) datagram expires before reaching the destination. Regardless of the size of the datagram, a fixed size ICMP (Internet Message Control Protocol) TIME EXCEEDED message is sent back to the source by the node that detects the problem. Similar situation also occurs when an IP datagram reaches a destination but there is no application running on the port addressed to process the datagram, which causes an ICMP message of DESTINATION UNREACHABLE to be sent to the sender of the datagram [5][7]. The roundtrip times measured by the tools presented in [1]-[3] are based on these ICMP response messages received by the sender on various sized datagrams sent. Evidently from (6), one can obtain a recursive relation for $m(i)$ as:

$$m(i) = m(i-1) + \frac{1}{b_u(i)} \quad (8)$$

Thus the bandwidth $b_u(i)$ of the link i can be expressed as

$$b_u(i) = \frac{1}{m(i) - m(i-1)} \quad (9)$$

In order to find the bandwidth of the first link, $m(0)$ is to be considered zero and hence, $b_u(1) = 1/m(1)$ only. From (3), (5), and (9) it follows that in order estimate the bandwidth $b_u(i)$, one first obtain the slope $m(i)$ of the straight line representing the roundtrip time $M_{s,r}(i)$ for the path from N_0 to N_i and the slope $m(i-1)$ of the straight line representing roundtrip time $M_{s,r}(i-1)$ for the path from N_0 to N_{i-1} . However, filtering of queue delays from measured roundtrip times is to be performed to obtain the straight lines representing $M_{s,r}(i)$ and $M_{s,r}(i-1)$. The following is the summary of the approach used in [1]-[3] to filter out the queue delays from measured roundtrip times:

1. For $i = 1, 2, 3, \dots, n$
 - a. Send a packet of size s_i to the remote host and measure the corresponding RTT (roundtrip time) for the packet from the response of the remote host.
 - b. Repeat step 1(a) for p times and then find r_i as the minimum of all p RTT's measured. The process is called minimum filtering.
2. Interpolate a regression line with points $(r_1, s_1), (r_2, s_2), \dots, (r_n, s_n)$ and correspondingly find the slope of the line.

The scatter plot of roundtrip times as shown in Fig. 2 illustrates how the minimum filtering of RTTs for each probe packet size and interpolating a regression line with the minimum RTTs can yield a slope for bandwidth estimation. The tools proposed in [1]-[3] use the same value of p for minimum filtering of measured roundtrip times for estimating of bandwidths of all links along an Internet path.

4. Fast Adaptive Bandwidth Estimation

As stated in [1] and [3], the technique of minimum filtering of queue delays from measured roundtrip times requires sending many probe packets of the same size and then only keeping the minimum of all measured RTTs for the packet size. The process is time-consuming, wasteful of bandwidth, and not adaptive at all to the distance of the remote host or the traffic condition. Finding the bandwidth of a link closer to the probing host (without sacrificing accuracy) should not require the same number of probe packets compared to the one further away along the path. Also, using the same number of probe packets of certain packet size indiscriminately for all links may not provide accurate bandwidths of the distant links in the path either. Specifically, in some situations, we observe that we do not need as many of the fixed number of probe packets as needed for tools in [1] and [3] to find the bandwidth of the link with similar accuracy.

Here, we propose an adaptive and iterative technique that filters out the effect of queue delays on the regression line. This is achieved with extra computation rather than transmission of extra packets. The proposed technique is outlined below:

1. *Measuring roundtrip times.* Send probe packets of size $s_1, s_2, s_3, \dots, s_n$ to the remote host and measure the corresponding RTT's as r_1, r_2, \dots, r_n from the responses of the remote node. Let $R = \langle r_1, r_2, \dots, r_n \rangle$ be the corresponding measured RTT vector. The probe packets of different sizes can be sent in some random order with sufficient time interval in between them. It is important that the packet size must not exceed the maximum transmission unit (MTU) of the path to the remote node.
2. *Calculating regression line.* Interpolate a regression line for points $(r_1, s_1), (r_2, s_2), \dots, (r_n, s_n)$ with their corresponding weights w_1, w_2, \dots, w_n where $w_i = 1$ for $1 \leq i \leq n$ and then obtain the corresponding interpolated RTTs on the regression line as y_1, y_2, \dots, y_n . Let $Y = \langle y_1, y_2, \dots, y_n \rangle$ represent the corresponding vector. Find the slope $m^{(j)}$ of the line for the iteration with $j = 0$.

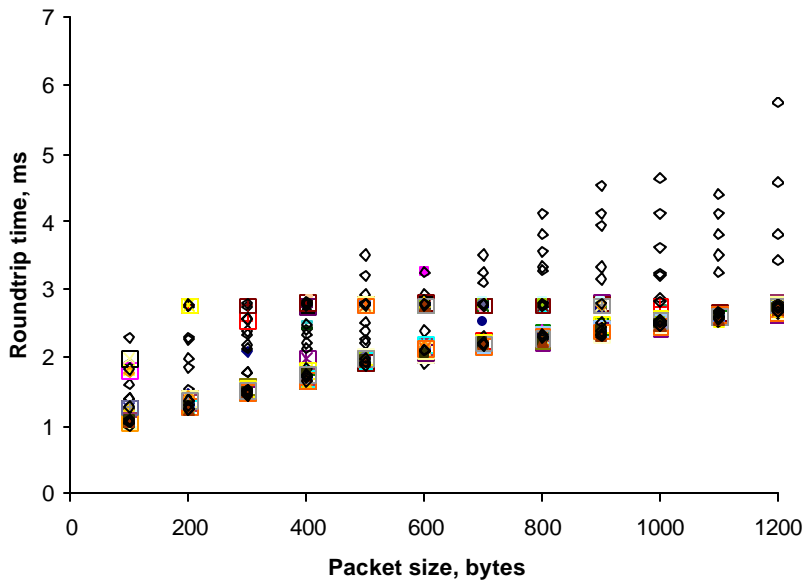


Fig. 2. Scatter plot of roundtrip times.

3. *Filtering and selective probing.* For $1 \leq i \leq n$, if $r_i > y_i$ (i.e. the point is above the regression line) then discard r_i and send a probe packet of s_i to the remote node to obtain a new RTT value. Set r_i as the minimum of the previous RTT and the new RTT.
4. *Iterating for convergence.* For $1 \leq i \leq n$, if $r_i \geq y_i$ then set $w_i = 0$; otherwise set $w_i = y_i - r_i$. Next, interpolate a regression line with the filtered points $(r_1, s_1), (r_2, s_2), \dots, (r_n, s_n)$ with weights w_1, w_2, \dots, w_n and obtain the corresponding new interpolated RTTs on the regression line as y_1, y_2, \dots, y_n . Increment j and obtain the slope $m^{(j)}$ of the line as in step 2. If $|m^{(j)} - m^{(j-1)}| / m^{(j)} \leq \mathbf{d}$ then stop; otherwise go back to step 3. It is to be noted that the constant \mathbf{d} has to be chosen carefully to ensure termination of the process without sacrificing accuracy too much.

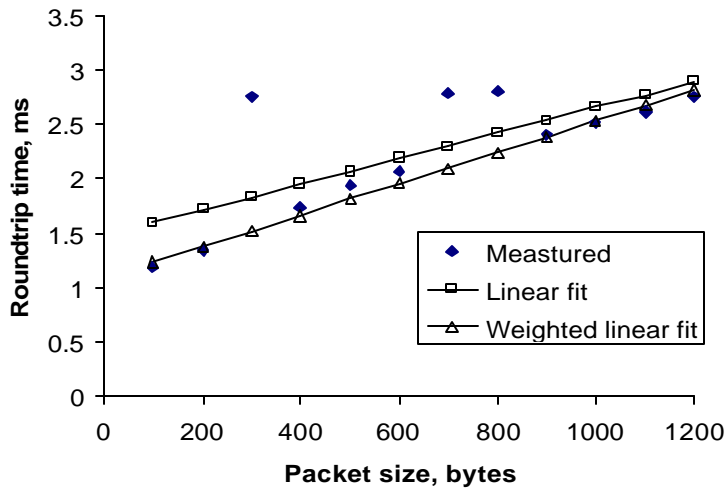


Fig. 3. Filtering of queue delays from RTT data.

Fig. 3 illustrates the algorithm for the first iteration of the filtering process, in which actual RTT points are shown with diamonds. A linear regression line with equal weights to all RTT data points is shown on the figure marked with squares. For the purpose of filtering and fast convergence, the RTT data points on or above the linear regression line are assigned weights of zero (essentially discarding the points) and the points below the line are assigned weights of their differences with their corresponding points on the regression line. As a result, the further

down a point is from the corresponding point on the regression line, the greater is its weight. This is what accelerates convergence for the process and also helps in selective probing of the certain packet sizes. A line marked with triangles is shown in Fig. 3 after a weighted linear regression fit, which is to be considered for selective probing and filtering of points for the next iteration. It is evident from Fig. 3 that the weighted linear regression line closely passes through the likely minimum RTT data points.

As stated earlier and shown in equation (9), to compute the bandwidth for link i , we need to find two regression lines, one for the distant node i and the other for the node $(i-1)$, and then obtain the slopes $m(i)$ and $m(i-1)$ from the lines respectively.

5. Experiments and Results

Based on the network model discussed above, we conduct an experiment on the publicly accessible Internet. In order to be consistent in comparing the performance of the proposed technique with the existing tools, we use the same ICMP protocol for probing and messaging as is used in the existing tools such *pathchar*, *clink*, and *pchar* [1]-[3]. As a representative of the existing tools, we decide to use *pchar* to probe a path containing 11 links on the Internet. By default, *pchar* starts with a probe packet of size 32 bytes with an increment of 32 bytes for succeeding packets until the packet size exceeds the MTU (maximum transmission unit) of the path with 32 repetitions for each link. For a path with MTU of 1500 bytes, *pchar* sends 1472 probe packets for each node along the path. Our results show that out of 11 links, *pchar* was only able to report the bandwidths of links 1, 5, and 7 which are shown in table 1 for comparison with the proposed technique. Due to various network conditions, route changes, limitations of the network path and the protocols, it may not be possible to correctly estimate the bandwidth of a link along an Internet path [4][7]. More information of the factors affecting accurate estimation of a link bandwidth may be found in [1]-[3].

Table 1. Comparison of Proposed Technique with pchar

Link	No. of Probe Packets Sent Using Proposed Technique	Estimated Bandwidth Using Proposed Technique	Estimated Bandwidth Using pchar
1	49	5.43 Mbps	5.87 Mbps
5	61	83.99 Mbps	40.43 Mbps
7	124	85.72 Mbps	76.40 Mbps

During our experiment with the proposed technique, we also experience a problem of obtaining an ICMP response message from a router (node) along the path of probing. Many routers on the Internet are these days configured for security reasons not respond to a probe packet or to delay the response based on some security policy or packet forwarding priority scheme. As a result, we limit our comparison of the proposed scheme with *pchar* for links 1, 5, and 7 only. Our results show that only 49, 61, and 124 probe packets were needed for links 1, 5, and 7 respectively for bandwidth estimation with $d = 1.0 \times 10^{-9}$. Although the bandwidths obtained for links 1 and 7 both with the proposed technique and *pchar* are close to each other, we observe some disparity of the results for link 5. As no information is available on the actual bandwidth of the link to us, it is not possible whether the proposed technique or *pchar* is inaccurate in the estimation of the bandwidth of the link. As can be seen from table 1, the proposed technique only needs 124 probe packets to estimate the bandwidth of link 7 compared to 1472 probe packets sent by *pchar*, which is less than 10% of the probe packets needed by *pchar*. Similar performance gain can be seen for other links as well. As can be observed from the table 1, for further links the scheme requires larger number of probe packets for convergence in the presence of more unstable traffic condition in a longer path.

6. Conclusion

In this work, we present an efficient scheme to measure the upstream bandwidth of a link along an Internet path. The scheme is based on fast and adaptive filtering of queue delays from roundtrip times measured on variable sized probe packets sent from a host to two successive nodes along the path for the link. Fast filtering of queue delays is achieved through a weighted linear regression fit of the roundtrip time data in an iterative manner. Depending on the

results of the previous iteration, the scheme selectively probes and includes certain data points on roundtrip times with judicious selection of weights for them for the next iteration of linear regression fit. Compared to existing schemes, it requires significantly less probe packets to achieve the similar accuracy for bandwidth estimation of a link. As a result, the proposed technique is less intrusive to the ongoing traffic along an Internet path. It is fast and hence can be used in an application that needs to know the path or link bandwidth immediately and accordingly schedule delivery of packets to the host at the other end of the path or link. In our future work, an extensive study of the scheme for performance will be conducted on various Internet paths and conditions and the corresponding results will be included in future publications.

7. References

- [1] A. B. Downey, "Using pathchar to estimate Internet link characteristics," *Proc. ACM SIGCOMM'99*, Cambridge, MA, September 1999.
- [2] V. Jacobson, "pathchar - a tool to infer characteristics of Internet paths", presented at the Mathematical Sciences Research Institute (MSRC), April 1997, <ftp://ftp.ee.lbl.gov/pathchar/>.
- [3] B. A. Mah, "pchar: a tool for measuring Internet path characteristics", <http://www.kitchenlab.org/www/bmah/Software/pchar/>, last accessed on March 17, 2006.
- [4] V. Paxson, "End-to-end routing behavior in the Internet", *IEEE/ACM Transactions on Networking*, vol. 5, pp. 601-615, 1997.
- [5] W. R. Stevens, *UNIX Network Programming*, Prentice Hall, 1997.
- [6] K. Lai and M. Baker. "Nettimer: a tool for measuring bottleneck link bandwidth," *Proceedings of the USENIX Symposium on Internet Technologies and Systems*, March 2001.
- [7] L. P. Peterson and B. S. Davie, *Computer Networks – A Systems Approach*, 3rd Edition, Morgan Kaufmann, 2003.
- [8] M. Pucci,, "Accuracy and expressiveness in adaptive bandwidth measurements," *Proceedings of the ISMA Bandwidth Estimation Workshop*, San Diego, CA, December 2003.
- [9] V. J. Ribeiro, R. H. Riedi, R. G. Baraniuk, J. Navratil, and L. Cottrell, "pathChirp: efficient available bandwidth estimation for network paths," *Proceedings of Passive and Active Measurement Workshop*, La Jolla, CA, April 2003.
- [10] A. Capone, L. Fratta, F. Martignon, "Bandwidth estimation schemes for TCP over wireless networks," *IEEE Transactions on Mobile Computing*, Vol. 3, No. 2, April-June 2004.