Bottleneck bandwidth measurement using PATH algorithm.

Jun. Wei

University of Windsor

Follow this and additional works at: https://scholar.uwindsor.ca/etd

Recommended Citation
https://scholar.uwindsor.ca/etd/1325

This online database contains the full-text of PhD dissertations and Masters' theses of University of Windsor students from 1954 forward. These documents are made available for personal study and research purposes only, in accordance with the Canadian Copyright Act and the Creative Commons license—CC BY-NC-ND (Attribution, Non-Commercial, No Derivative Works). Under this license, works must always be attributed to the copyright holder (original author), cannot be used for any commercial purposes, and may not be altered. Any other use would require the permission of the copyright holder. Students may inquire about withdrawing their dissertation and/or thesis from this database. For additional inquiries, please contact the repository administrator via email (scholarship@uwindsor.ca) or by telephone at 519-253-3000ext. 3208.
Bottleneck Bandwidth Measurement

Using PATH Algorithm

By

Jun Wei

A Thesis
Submitted to the Faculty of Graduate Studies and Research
Through the School of Computer Science
In Partial Fulfillment of the Requirements for
The Degree of Master of Science at the
University of Windsor

Windsor, Ontario
Canada
2003

© Jun Wei
The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.
Abstract

Network bandwidth is an important resource in the Internet. As greater numbers of people and companies implement their businesses using the Internet, the measurement of network bandwidth, especially the bottleneck-link bandwidth, becomes a critical and challenging issue.

To improve the practicability, accuracy and performance of existing network-bandwidth measurement algorithms, such as PathChar, Bprobe and Cartouche algorithm, we have developed a new method: PATH (PAcket-Train with Header) algorithm. PATH is the combination of single-packet and packet-pair algorithms. It is a sender-based algorithm that is straightforward to implement in the Internet.

To verify and improve upon the PATH algorithm, we have developed a user-friendly tool: BAMA (BAndwidth Measurement Algorithms) simulator. It can simulate most network bottleneck bandwidth algorithms. By using this BAMA simulator and Network Simulator to implement the PATH algorithm and compare it with the other algorithms, we prove that, under heavy cross traffic, PATH is more accurate than existing sender-based algorithms.

Keywords:
Bottleneck link bandwidth, single packet, packet pair, packet train, PATH, SMSP, SDSP
Acknowledgements

Completing this thesis is a great opportunity for me to study and implement the network knowledge that I learned in my master program. I would like to gratefully acknowledge the enthusiastic supervision of Dr. Akshai Aggarwal. He is the best advisor and teacher I could have wished for. Thanks for his encouragement, excellent teaching, and many good suggestions.

I would also like to express my appreciation to Dr. Diana Kao, Dr. Robert Kent, Judy Decou, and Dr. Jessica Chen for their valuable advice and comments concerning this thesis.

Lastly, and most importantly, I wish to thank my wife, Zhe, for her understanding, endless patience and encouragement.
# Table of Contents

Abstract ........................................................................................................................................ III

Acknowledgements ...................................................................................................................... IV

List of Figures ............................................................................................................................... VI

1. Introduction ............................................................................................................................... 1
   1.1 Link bandwidth and available bandwidth ................................................................. 1
   1.2 Why measure bandwidth? ......................................................................................... 2
   1.3 Thesis statement ......................................................................................................... 3
   1.4 Structure of thesis document ..................................................................................... 4

2. Literature Review ..................................................................................................................... 5
   2.1 Single-packet algorithm ............................................................................................. 5
   2.2 Packet-pair algorithm ................................................................................................. 8
   2.3 Other techniques ......................................................................................................... 14

3. PATH Algorithm ..................................................................................................................... 21
   3.1 The problems of existing algorithms ......................................................................... 21
   3.2 Packet Train with Header .......................................................................................... 22
   3.3 Features of PATH algorithm ..................................................................................... 26

4. Experiments and Results ......................................................................................................... 30
   4.1 BAMA simulator ......................................................................................................... 30
   4.2 Network simulator ....................................................................................................... 43
   4.3 SDSP and SMSP .......................................................................................................... 46

5. Conclusions and Future Work ............................................................................................. 51
   5.1 Conclusions .................................................................................................................. 51
   5.2 Future work .................................................................................................................. 51

References .................................................................................................................................... 53

Appendix A Network Quantitative Metrics .............................................................................. 56
Appendix B Some Experiments on Network Simulator ............................................................ 60

VITA AUCTORIS .......................................................................................................................... 61
List of Figures

Figure 2-1 Pathchar network model [Dow99a] ................................................................. 6
Figure 2-2 Illustration of packet flow through a bottleneck link [Jac88] .......................... 8
Figure 2-3 Packet pair from source to target [CC96] ...................................................... 9
Figure 2-4 Probe packets alone queued at router [CC96] ............................................... 10
Figure 2-5 SBPP, RBPP and ROPP [LB99] ..................................................................... 12
Figure 2-6 Several packets across the link l-1 and l [LB00] ........................................... 15
Figure 2-7 Paced probe example [HBB01] ................................................................. 18
Figure 2-8 Cartouche algorithm [HBB01] .................................................................... 18
Figure 3-1 Network structure example ....................................................................... 23
Figure 3-2 Packet train with header ........................................................................... 23
Figure 3-3 PATH illustration –1 .................................................................................. 27
Figure 3-4 PATH illustration –2 .................................................................................. 27
Figure 3-5 PATH illustration –3 .................................................................................. 28
Figure 3-6 PATH illustration –4 .................................................................................. 28
Figure 3-7 PATH illustration –5 .................................................................................. 29
Figure 3-8 PATH illustration –6 .................................................................................. 29
Figure 4-1 BAMA simulator ....................................................................................... 31
Figure 4-2 BAMA flow chart ...................................................................................... 33
Figure 4-3 Network Simulator estimation distribution .................................................. 35
Figure 4-4 BAMA simulator estimation distribution ...................................................... 35
Figure 4-5 The estimation result of Network Simulator ................................................ 37
Figure 4-6 The estimation result of BAMA simulator .................................................... 37
Figure 4-7 PATH without ACK .................................................................................... 38
Figure 4-8 PATH without ACK (with other cross traffic) ............................................ 39
Figure 4-9 Parameters for the PATH algorithm ............................................................ 39
Figure 4-10 Parameters for the BProbe algorithm ......................................................... 40
Figure 4-11 Parameters for the PATH algorithm (with other cross traffic) .................. 41
Figure 4-12 Parameters for the Cartouche algorithm ................................................... 42
Figure 4-13 Parameters for the Cartouche algorithm (with other cross traffic) .......... 43
1. Introduction

Similar to telephone techniques in the early nineteenth century, more and more people, enterprises, and organizations are now using and depending on the Internet. One of the most important reasons why the Internet has become so popular is the improvement of Internet performance. Nielsen's Law of Internet bandwidth states that: “a high-end user's connection speed grows by 50% per year” [Nie98]. Although over the last decade, the Internet bandwidth has increased enormously, the applications, which need greater bandwidth, have also proliferated. Hence monitoring the performance of the network, especially the network bandwidth, continues to be a challenging research field.

1.1 Link bandwidth and available bandwidth

Bottleneck link bandwidth and available bandwidth are useful for most network applications and users. In this thesis, we mainly focus on the link bandwidth. The bottleneck link bandwidth of a path is also called path capacity, path bandwidth or bottleneck capacity. In an ideal network environment, if there is no other traffic in the path, bottleneck link bandwidth is the bandwidth of the lowest-bandwidth link from source to destination. It is limited by the bottleneck link’s underlying capacity.

If there are $H$ hops in a path and $C_i$ is the capacity of link $i$, then the bottleneck link bandwidth is

$$C = \min_{i=0, \ldots, H} C_i$$  \hspace{1cm} (1-1) [DRM01a]
Available bandwidth is defined as "the maximum rate that the path can provide to a flow without reducing the rate of the cross traffic". [JD02a] Simply speaking, available bandwidth is the amount of bandwidth, available for a flow from the sender to the receiver, if we consider the cross traffic. Millions of users from all over the world use the Internet simultaneously. The available bandwidth depends on the link bandwidth and the presence of competing traffic.

1.2 Why measure bandwidth?

Network users often state that network speed is slow. There are two main reasons that would cause the experience on the machine to be "slow". One is the limitation of the host machine, such as slow file servers or poor performance workstations, which do not have enough memory or hard disks. Another reason is the properties of the network, such as the limited bandwidth and network latency. The first problem may be solved by the user through acquisition of better hardware. But, it is difficult for a user to attempt a solution of the second problem, without the availability of an easy method to measure the network bandwidth. To improve network performance, we need to find ways to monitor it. Some of the reasons that bandwidth measurement is necessary are [LB99]:

- As more and more applications are implemented in the Internet, a large investment is being made to implement high-speed and large-bandwidth networks. But, how do we know the performance of the network has improved and we have achieved the expected result?
- Network users benefit from knowing the bandwidth. For example, the network protocol or application programmers need to know the bandwidth and latency
with accuracy to evaluate the efficiency of their protocols or programs. The more
the availability of information about network path properties, the more efficiently
network users can use the network to transfer their data.

- Measuring bandwidth is the key to congestion control. In order to keep the
  amount of data that an application sends out below the network capacity, we must
  know the available bandwidth.

- Network applications could choose the best “route” dynamically to access the
  server. There are many potential paths through the Internet between two hosts.
  The bandwidth, round trip time and packet loss rate of various paths are different.
  By measuring these parameters, network applications can select the best path to
  get quick response and better performance.

- For mobile computing, knowing the bandwidth is important. Mobile computers
  generally have more than one network interface and the available server changes
  frequently. Measuring the bandwidth and latency is the first step to find the best
  network interface to transfer the data.

- In the network control mechanism, measuring the bandwidth is a critical technique
  to provide the guarantees of QoS (Quality of Service).

1.3 Thesis statement

This thesis presents a new sender-based method, PATH, to improve the practicability and
accuracy of the network bottleneck-link bandwidth algorithms. PATH is a new algorithm
based on the single-packet, packet-pair and packet-train algorithms. To verify and
improve the PATH algorithm, the thesis introduces a user-friendly tool, BAMA simulator.
By using the BAMA simulator and Network Simulator [NS], PATH is compared with
other algorithms, such as the \textit{BProbe}, packet train, and Cartouche. Under heavy cross traffic, PATH is found to be more accurate than existing sender-based algorithms.

1.4 \textit{Structure of thesis document}

The remainder of the thesis is organized as follows. Chapter 2 describes the existing network bottleneck link bandwidth algorithms and their problems, Chapter 3 explains our PATH algorithm and the features of the PATH algorithm, Chapter 4 presents the experiments and results by using our BAMA simulator and Network Simulator, and Chapter 5 concludes the thesis and outlines possible future directions for research in this area.
2. Literature Review

Because the Internet continues to expand at a fast rate with an ever-increasing traffic volume and diversity of technologies, it is a challenging task to measure the bandwidth of the network. Generally, link-measurement techniques can be divided into three categories: single-packet algorithms, packet-pair algorithms and other techniques.

2.1 Single-packet algorithm

Single-packet techniques are based on the common sense idea that a packet can be transmitted faster using a high bandwidth link. So, if the time that a packet takes to cross a link is known, the bandwidth of that link can be estimated.

Van Jacobson at Lawrence Berkeley National Laboratory (LBL) presented pathchar at the Mathematical Sciences Research Institute (MSRI) in 1997 [Jac97]. Downey improved the accuracy of pathchar and reduced the number of measurements pathchar needs [Dow99a]. Matoba improved this algorithm by statistical methods [MAM00]. Clink [Dow99b] and Pchar [Mah01] both are link-bandwidth measurement tools based on the single-packet algorithm.

Pathchar uses the time-to-live (TTL) field in the IP packet. TTL is used to monitor and remove the packets that cannot reach the destination. The sender initializes the TTL value of an IP packet and transfers the packet to a router, where the TTL value will be decremented by one. TTL will be decremented by one at each intermediate router the packet passes through, until the IP packet arrives at the destination computer. However,
if the TTL value is decremented to zero, the packet will be discarded and the router will send back an ICMP time exceeded packet to the original sender. In other words, if packets are sent to the destination with TTL equal to \( n \) (\( n \) is less then the number of intermediate routers), the \( n^{th} \) router is forced to send back ICMP packets with its own IP address.

*Pathchar* sends out a series of probe packets with different values of \( n \) and different packet sizes. By using statistics to analyze the time that error packets are received, *pathchar* can calculate the bandwidth and latency of each link along the path [Dow99a].

![Diagram](image)

**Figure 2-1 Pathchar network model [Dow99a]**

Figure 2-1 [Dow99a] shows the *pathchar* network model. The round trip time (RTT) from the \((n-1)^{th}\) node to \( n^{th} \) node is:

\[
RTT = q_1 + (lat1 + \text{size}/bwn1) + q_2 + \text{forward} + q_3 + (lat2+\text{errorSize}/bwn2) + q_4
\]

(2-1) [Dow99a]

where \( q_1, q_2, q_3 \) and \( q_4 \) represent the queue times in the node \((n-1)\) and node \( n \), and \text{forward} is the time that the router takes to process the packet and to forward the error packet to the return queue. \text{lat1} and \text{lat2} are the latency times between node \((n-1)\) and
node n. bwn1 and bwn2 are the bandwidth between these two nodes. err_size represents the size of the error packet.

After simplification, we can get the RTT from node (n-1) to node n is:

$$RTT_n = \frac{size}{b_n} + l_n$$

(2-2) [Dow99a]

Where $b_n$ is the bandwidth of the link from node (n-1) to node n, and $l_n = lat1 + lat2$.

Summing the links we can get the RTT from the sender to node n:

$$RTT_x = \frac{size}{b_x} + l_x + \sum_{i=0}^{x-1} (RTT_i)$$

(2-3) [CM01]

Pathchar uses equations (2-2) and (2-3) to estimate link characteristics.

The features and problems of the single-packet algorithm:

- The single-packet algorithm is easy to implement in the Internet. It is based on TCP/IP protocol, which is widely used today. It is not necessary to install any special software to measure the parameters of the network.

- The single-packet algorithm cannot provide a very accurate estimation. The main modeling errors include: neglect of the queuing and forwarding time; the return path can be different from the outgoing path; probes may exceed the Maximum Transfer Unit (MTU) of a link, and router alteration (the path between two hosts changes sometimes).

- The methods based on the single-packet algorithm send out a large number of probe packets, which consume most of the available bandwidth of the network. This affects the network performance.
2.2 Packet-pair algorithm

The basic packet-pair algorithm was presented in “A Control – Theoretic Approach to Flow Control” by Srinivasan Keshav, in 1991 [Kes91]. Figure 2-2 [Jac88] shows the effect of packet flow through a bottleneck link.

![Diagram of packet flow through a bottleneck link](image)

*Figure 2-2 Illustration of packet flow through a bottleneck link [Jac88]*

If the sender sends out a sequence of packets, under ideal conditions, these packets will be queued at the bottleneck link and will be transferred to the next router with an interval, which may depend upon the bandwidth of the bottleneck link. By measuring the interval time of returning packets, the link bandwidth can be estimated.

We assume that the path is stable and routing tables do not update very frequently. The source sends out two \( b \)-bit packets (probes) with an interval of \( T_s \) sec. The bottleneck bandwidth of the path is \( \beta \) and we define \( Q_b \) as the service time of one \( b \)-bit packet at the bottleneck link:
If $T_s < Q_b$, before the first packet is able to go through the bottleneck link, the second packet is guaranteed to arrive at the queue of the start point of the bottleneck link and begin to wait. So, these two packets will exit the link $Q_b$ seconds apart. Figure 2-3 [CC96] shows the procedure of a packet pair through a bottleneck link.

![Diagram](image_url)

**Figure 2-3 Packet pair from source to target [CC96]**

By monitoring the packet size $b$ and interval $Q_b$, we can estimate the bottleneck bandwidth $\beta$ by the formula:

$$\beta = b/Q_b$$

There are three kinds of packet-pair algorithms: sender-based packet pair, receiver-based packet pair and receiver-Only Packet pair.

- **Sender-based Packet Pair**

  *BProbe* is a sender-based packet pair (SBPP) tool to measure the link bandwidth. It was introduced in 1996 [CC96]. To make reliable measurements, *BProbe* sends out a sequence of ICMP ECHO packets. By measuring the interval time of returning packets,
the link bandwidth can be estimated. Figure 2-4 [CC96] illustrates how BProbe obtains the interval $Q_b$.

![Diagram](image)

Figure 2-4 Probe packets alone queued at router [CC96]

SBPP algorithms are easy to implement in the Internet. No special measurement software needs to be installed on the receiver side. But the accuracy of estimation is not satisfactory. The methods based on SBPP use ICMP packet, which will cause some additional noise: possible asymmetry of the return path and noise due to delay in generating the ACK. Another problem is that the BProbe algorithm does not work very well under heavy cross traffic situations.

- Receiver-based Packet Pair

Vern Paxson’s algorithm [Pax97a, Pax97b] is based on a receiver-based packet pair (RBPP). RBPP checks the data at the receiver side. It gets rid of the additional noise:
possible asymmetry of the return path and noise due to delay in generating the ACK.
So, RBPP is more accurate than SBPP.

Vern Paxson extended the packet-pair concept by developing Packet Bunch Modes (PBM). The basic packet-pair algorithm cannot measure the multi-channel bottleneck link bandwidth. Multi-channel links, such as ISDN links, use multi-channels that operate in parallel. When the first packet comes and the link is idle, the packet goes to the first channel. If the second packet comes immediately while the first channel is not idle, then it goes out over the second channel. The delivery of packets in a multi-channel does not queue behind each other. It invalidates the estimation by the packet-pair algorithm. PBM can solve this problem. It forms a range of estimates based on different packet bunch sizes and then analyzes the results to find more than one bottleneck value. By estimating at the receiver side for a range of packet bunch sizes, multiple bottleneck values can be detected.

Using different large bunch sizes helps to solve the clock resolution problem. Clock resolution is the minimum time that the clock can detect. It may introduce estimation errors. For example, if the clock resolution is 10 milliseconds and the packet size is 512 bytes, then it is implies we cannot distinguish a bottleneck bandwidth greater than 512 bytes/10 ms = 51,200 byte/sec.

Using small bunch sizes minimizes the risk of underestimation due to noise diluting the spacing of bunches. Allowing for multiple bottleneck values gets rid of the effect of multi-channel paths.
• Receiver-Only Packet pair

To make packet-pair algorithms practical, robust and widely used, Kevin Lai and Mary Baker developed an improved version, Receiver-Only Packet pair (ROPP), in 1999 [LB99].

ROPP, as shown in Figure 2-5 [LB99], is a variation of RBPP. It installs software on the receiver side. Only the receiver records the packet arrival time. So, there is no clock synchronization problem. It provides almost the same accurate result as RBPP. Figure 2-4 explains the differences among the SBPP, RBPP and ROPP algorithms.

![Diagram showing Sender-Based, ROPP, and Receiver-Only Packet pair]

Figure 2-5 SBPP, RBPP and ROPP [LB99]

Kevin Lai and Mary Baker proposed three solutions to improve link-bandwidth estimation algorithms: using packet windows to accurately estimate fast bandwidth changes, using ROPP to make the algorithm practical and accurate, and using Potential Bandwidth Filtering to deliver accurate bandwidth estimation for various packet sizes and transmission rates.
In 2001, based on the kernel-density estimation and potential bandwidth-filtering (PBF) algorithm, Kevin Lai and Mary Baker presented Nettimer [LB01] to measure the link bandwidth. The kernel-density estimator is a well-known statistical algorithm. Consider a kernel function $K(t)$ [Sco92, ASW81] with the property:

$$\int_{-\infty}^{\infty} K(t) dt = 1$$

Then the density of sample $x$ is

$$d(x) = \frac{1}{n} \sum_{i=1}^{n} K \left( \frac{x - x_i}{c \cdot x} \right)$$  \hspace{1cm} (2-6) [LB01]

where $c$ is the kernel width ratio and $n$ is the number of points within $c \cdot x$ of $x$. The larger the value of $c$, the more accurate the result. At the same time, it means more computation. $C$ is 0.10 in Nettimer. Nettimer uses the kernel-density algorithm to estimate the density of the sample. The kernel function it uses is

$$K(t) = \begin{cases} 
1 + t, & t \leq 0 \\
1 - t, & t > 0 
\end{cases}$$  \hspace{1cm} (2-7) [LB01]

Nettimer can passively measure the bottleneck link bandwidth. Active measurement methods, such as tcpanaly [Pax97a, Pax97b] and pathrate [DRM01a, DRM01b], are more accurate than passive ones. But an active method requires more resources to estimate the bandwidth. The experiments show that “in most cases, nettimer has an error of less than 10%, but at worst it has an error of 40%. It converges within 10KB of the first large packet arrival while consuming less than 7% of the network traffic being measured” [LB01].

13
*Pathrate* is a tool developed by Constantinos Dovrolis, Parameswaran Ramanathan and David Moore in 2001 [DRM01a, DRM01b]. As already discussed, the main problem of the packet-pair algorithms is that the estimates of bandwidth are still not accurate, especially in heavily loaded paths. Because the distribution of bandwidth measurements is multi-modal, some local modes are often stronger than the capacity mode. So using standard statistical procedures cannot provide accurate results.

*Pathrate* uses the two-end-points methodology. By implementing two phases, packet-pair probing phase and packet-train probing phase, *Pathrate* can get a more accurate estimation.

Although RBPP and ROPP algorithms are more accurate than SBPP, measurement software needs to be installed at both the sender and receiver sides. This is really difficult to implement for most network applications, such as congestion control, dynamic server selection, QoS and so on.

### 2.3 Other techniques

- Packet-tailgating algorithm

Lai and Baker presented the packet-tailgating algorithm in 2000 [LB00]. It combines both the single-packet algorithm and packet-pair algorithm. “For each link, it sends a large packet with a time-to-live (TTL) set to expire at that link followed by a very small packet that will queue continuously behind the large packet until where the large packet expires.” [LB00] Figure 2-6 [LB00] displays an example of several packets across the link $l-1$ and $l$. 

14
Figure 2-6 Several packets across the link I-I and I [LB00]

From Figure 2-6 [LB00], $t_i^k$ (the time that packet $k$ reaches the link $l$) can be calculated by:

$$t_i^k = t_0^k + \sum_{i=0}^{I-1} \left( \frac{s_i^k}{b_i} + l_i + \max(0, t_{i+1}^{k-1} - l_i - t_i^k) \right)$$

(2-8) [LB00]

To simplify the calculation, we make an assumption that the first packet will not queue. And we define variable $l'$ and $b'$

$$l' = \sum_{i=0}^{I} l_i \quad \text{and} \quad b' = \sum_{i=0}^{I} \left( \frac{1}{b_i} \right)$$

[LB00]
Then we can simplify the equation (2-8) to (2-9) to:

\[
b_i = \frac{S^{k-1}}{\left( t_n^k + \frac{S^k - S^{k-1}}{b^{l-1}} - \frac{S^k}{b^{n-1}} - t_{0}^{k-1} - t^{n-1} \right)}
\]  \hspace{1cm} \text{(2-9) [LB00]}

The sizes of two packets are \(s^{k-1}\), and \(s^k\). \((t_n^k)\) is the arrival time of the second packet. \((t_0^{k-1})\) is the transmission time of the first packet. The bandwidth of all earlier links are \((b^{l-1})\) and \((b^{n-1})\), and the delay of all earlier links is \((l^{n-1})\). Using these parameters, we can calculate the bandwidth of a link at which queuing occurs \((b_i)\).

To measure these parameters, packet-tailgating technique uses two phases, sigma phase and tailgating phase.

Sigma phase is used to measure \((b^{n-1})\) and \((l^{n-1})\), which are non-link-specific quantities. Just like the single packet we have discussed above, it sends single packets of various sizes. The difference between the single-packet algorithm and sigma phase of packet-tailgating algorithm is that the single packet needs to measure these parameters for each link; the sigma phase only calculates the \((b^{n-1})\) and \((l^{n-1})\) for the source to the destination once.

The tailgating phase is used to calculate the remainder of the variables in equation (2-9). First, we send out the largest possible packet without queuing and fragment, and then we send out a small packet. The TTL field of the first packet is \(l\). It will be
dropped at link $l$. Then the second packet can continue without queuing to the destination.

Packet tailgating consumes less network bandwidth because it sends out an order of magnitude of fewer packets but gets similar accuracy. "The estimation does not rely on the consistent behavior of the router handling ICMP packets, and does not rely on timely delivery of acknowledgements." [LB00]

- **Cartouche 's algorithm**

  The main feature of Cartouche 's algorithm [HBB01] is that it can measure bottleneck bandwidth of any sub path along a path in the network. This algorithm actively measures packet bunch probes with non-uniform packet sizes.

  To activate the probes at a particular location in the network, a large pacer packet is used. A pacer packet is a large probe packet, which is sent before the packet-pair probes and which is designed to be dropped where the packet pair is to be active. Figure 2-7 [HBB01] depicts an example of paced probes: packet pair [pp] follows a large pacer packet $a$. So, if pacer packet $a$ is large enough, both of $p$ 's will queue after $a$ at the router $R$. When $a$ is dropped at router $R$, packet pair [pp] is activated to measure the bottleneck bandwidth of the sub path (from $R$ to $A$).
The Cartouche algorithm sends out \([apm/pq]r^{-1}pm\) probe list to the receiver. In this probe list, \(a, p, q, m\) all are the probes. The size of packet \(a\) is much larger than packet \(p, q\) and \(m\). The Cartouche algorithm can measure the bottleneck bandwidth of any sub path of the network. Figure 2-8 illustrates the situation when the Cartouche probe list passes the bottleneck link.
- **NetProbe**

*NetProbe* [Ciu02] is an active tool, which can measure one-way bandwidth and jitter asymmetry. Jitter is the variation of the communication delay. If \( t(i) \) is the time when the \( n^{th} \) octet of a stream is received by the destination, the one-way bandwidth from octet \( i \) to octet \( j \) can be calculated by:

\[
Bw(i, j) = \frac{j - i}{t(j) - t(i)} \quad \text{(Octets/sec)} \tag{2-10}[Ciu02]
\]

*NetProbe* set up a new protocol to implement equation (2-8). Compared with other one-way measuring tools, *NetProbe* does not require accurate clock synchronization.

Packet tailgating, Cartouche, and *NetProbe* all use a receiver-based algorithm. Both the sender and receiver side need to install measurement software.

- **Packet-train algorithm**

Ahlgren, Bjorkman and Melander extended the *Bprobe* algorithm by generating even longer trains and analyzing the data and behaviour of these longer trains. The packet-train algorithm [ABB99] was used to measure available bandwidth. It sends a fairly long train of back-to-back packets (100 UDP packets where the size of each packet is 1000 bytes [ABB99]) from the sender to a receiver. Each packet will be time-stamped by the receiver. After the receiver gets the whole packet train, it will send all timestamps to the sender. The bandwidth can be estimated by using the following formula:

\[
bw_{i-j} = \frac{(i-j)s}{t_i - t_j}, i = j + 1,..., n \tag{ABB99}
\]
where $s$ is the packet size, $t_i$ is the packet $i$’s timestamp and $n$ is the number of the packets in the train.

“The analysis shows that longer trains give more reliable bandwidth estimations.” [ABB99]. But Constantinos Dovrolis, Parameswaran Ramanathan and David Moore demonstrated that, as the number of probes increases, the bandwidth distribution becomes unimodal, gradually gathering at the value of Asymptotic Dispersion Rate (ADR) [DRM01a, DRM01b]. ADR is the average bandwidth estimate at the receiver. It is lower than path capacity and is independent of the number of packet train. It is not available bandwidth.
3. PATH Algorithm

3.1 The problems of existing algorithms

- Practicability

Most of the link-bandwidth estimation packet-pair algorithms, such as the Paxson [Pax97a, Pax97b], Lai [LB99, LB00, LB01], and the Cartouche algorithm [HBB01], need to install software on the receiver side. This is hard to implement for most bandwidth-measurement applications. For example, when users want to download a file from a FTP server, they try to find the "best" path to minimize the file-transfer time. For the receiver-based packet pair (RBPP) and receiver-only packet pair (ROPP) algorithms, to estimate link bandwidth, some special bandwidth-estimation software has to be installed on the receiver side, which is the FTP server in this case. This limitation restricts in that the RBPP and ROPP algorithms can only be implemented on a server that already has the required measurement software installed.

Single-packet and Sender-based Packet Pair algorithms are easy to implement. Based on the TCP/IP protocol, these do not need installation of any special software to measure the parameters of network.

- Accuracy

The accuracy of the methods based on the single-packet algorithm is not adequate. There are some assumptions in the single-packet model, such as:

(1) The transmission delay is linear with the packet size.

(2) The intermediate routers use store-and forward technology.
(3) No other traffic affects the measurement.

The first and second assumptions are almost true for most networks, especially for the Internet environment. The last assumption makes the single-packet algorithm usable only for some special-purpose networks.

- Performance

To get better bandwidth estimation, methods based on single-packet algorithms send out a large number of probe packets, which may consume a large part of the available bandwidth of the network. It affects the network performance.

### 3.2 Packet Train with Header

Based on the existing network bottleneck-link bandwidth measurement algorithms, such as the BProbe, packet-tailgating, packet-train and Cartouche algorithm, we present a new method: Packet Train with Header (PATH). PATH is a new algorithm based on the single-packet, packet-pair and packet-train algorithms.

There are two steps to implement the PATH algorithm:

**First:** we use the simple single-packet algorithm (SMSP) to check the network structure and determine the bottleneck link $L_k$. Compared with the standard single-packet algorithm (SDSP), the SMSP algorithm does not have to measure the bandwidth of each link of the whole network. It only checks the position of the minimal bandwidth, which is the bandwidth of the bottleneck link. Although SMSP can give an estimate of the bottleneck link bandwidth, this value may not be accurate. It is only used as a reference value for the second step of the PATH algorithm.
In the case of Figure 3-1, the bottleneck link (between router \( R_{k,i} \) and \( R_k \)) is \( L_k \), the main purpose of the SMSP algorithm is estimating the value of \( k \). This should greatly reduce the traffic that the SDSP algorithm generates.

![Network structure example](image)

**Figure 3-1 Network structure example**

We use the SDSC dataset [Jac97, Dow99a] to verify the difference of traffic volume between the SDSP and SMSP. The SDSC is one of the two datasets used by Downey [Dow99a]. In his paper he named it as the SDSC dataset, which referred on the network between Colby College, Waterville, Maine and San Diego Supercomputing Center. We shall discuss the SDSC dataset and the experiment in the section 4.3.

**Second:** Use Packet Train with header probe to measure link \( L_k \).

![Packet train with header](image)

**Figure 3-2 Packet train with header**
In Figure 3-1, suppose that the link between router $R_{k-1}$ and $R_k$ is the bottleneck link $L_k$. The source sends out a header packet $H$ and a packet train $T_1, T_2, ..., T_n$, just like the illustration of Figure 3-2. Both the header and packet train are UDP packets. All the packets $T_i$ of the packet-train are of the same size. The size of header packet $H$, $S_h$, is much larger than $S_n$, the size of $T_i$. In each $T_i$, there is data of only 8 bytes. This data is used for identifying the packet.

The TTL of packet $H$ is $k-1$, and it will be removed after it arrives at router $R_{k-1}$ and release the packet train to test the bottleneck link. The TTL of the packet-train packet $T_i$ is $k$, so they will stop at router $R_k$ and $R_k$ should reply through ICMP time exceeded packets to the source.

ICMP packets can include the 8 bytes of original data from the UDP packets, which is the packet-train packet identification. We can check these identifications to discard the result with the packet missing and packet disorder.

The sender will record the time when these ICMP packets return to sender: $t_1, t_2, ..., t_n$, we can get the intervals: $\Delta t_1, \Delta t_2, ..., \Delta t_{n-1}$, by using equation (3-1)

$$\Delta t_i = t_{i+1} - t_i$$  \hspace{1cm} (3-1)

If the sender sends out $m$ times packet train with header probes:

$$[t_{11}, t_{12}, ..., t_{1n}], [t_{21}, t_{22}, ..., t_{2n}] \ldots [t_{m1}, t_{m2}, ..., t_{mn}]$$

we can calculate the interval result array $\Delta t_{ij} \ (i \leq m \ \text{and} \ j < n)$
\[ \Delta t_{ij} = t_{i(j+1)} - t_{ij} \] (3-2)

Based on the array \( \Delta t_{ij} \), we use histogram to analyze it and try to get the estimated interval \( \Delta t_i \). The most difficult part in using histograms to analyze data is to decide the bin value of the histogram. To decide the bin of a histogram, Scott, D. presented the optimal histogram bin size, which "provides the most efficient, unbiased estimation of the probability density function". [Sco79]

\[ W = 3.49 \alpha N^{-1/3} \] (3-3) [Sco79]

where \( W \) is the width of the histogram bin, \( \alpha \) is the standard deviation of the distribution and \( N \) is the number of available samples.

After we get the estimated interval \( \Delta t_i \), the bottleneck bandwidth \( \beta \) can be calculated from equation (3-4):

\[ \beta = \frac{S_i}{\Delta t_i} \] (3-4)

where \( S_i \) is the size of packet in the packet train.

In the first step of the PATH algorithm, we use the SMSP (Simple Single Packet) algorithm to check the position of the bottleneck link. At the same time, SMSP can give an estimate of the bottleneck link bandwidth \( \beta_s \). Although \( \beta_s \) may be not accurate, we still can use it as a reference to check the accuracy of the bottleneck bandwidth \( \beta \).
3.3 Features of PATH algorithm

3.3.1 Easy to implement

PATH is a sender-based algorithm. We do not install any special measurement software on the receiver side. It is easy to implement the PATH algorithm in the Internet.

3.3.2 Accuracy

It is more accurate than the existing sender-based algorithms in the situation with cross traffic. The PATH algorithm works better than the standard sender-based algorithm (such as the single-packet and BProbe algorithm) under heavy cross traffic.

In the structure of Figure 3-1, cross traffic can exist in the link: between source computer and router $R_{k-1}$, or between router $R_k$ and destination computer. To explain why the PATH algorithm works better than the standard sender-based algorithm under cross traffic, we will discuss these cases one by one.

- Cross traffic exists in the links between router $R_k$ and destination computer

Because all the PATH packets will stop at the router $R_k$, the cross traffic in the link between the router $R_k$ and destination computer will not affect the measurement of the PATH algorithm. But, the standard sender-based algorithms need to send packets to the destination computer. The cross traffic in the link between the router $R_k$ and destination computer will dilute the spacing among these probes. So, the estimates of standard sender-based algorithms under cross traffic are not as accurate as those obtained by PATH.
• Cross traffic exists in the links between the source computer and router $R_{k,l}$

In this case, the large header of the PATH probe train can make packets $T_1, T_2, ... T_n$ stay together before they reach router $R_{k,l}$. In Figure 3-3, suppose the link $L_j$ (between router $R_{j-1}$ and $R_j$) is part of the network in Figure 3-1, and $j<k-1$. If $S_h$ is large enough, $H$ will temporarily block the link $L_j$. If the packet train packets, $T_1, T_2, ... T_n$, reach the router $R_j$, during the transmission of the header packet $H$ in the link $L_j$, they will queue in the router $R_{j-1}$ and wait until the header packet is received by $R_j$.

![Figure 3-3 PATH illustration -1](image)

After $H$ is received by $R_j$, the packet train $T_1, T_2, ... T_n$ will be transferred, just as in Figure 3-4. There will be some intervals between the packet-train packets.

![Figure 3-4 PATH illustration -2](image)
Figure 3-5 PATH illustration -3

Figure 3-5 shows the packet train as it reaches router $R_j$. Again, they will wait in the queue of $R_j$ and the intervals among the packet-train packets will become zero again. So, if the header is large enough, it can keep the packet train together until they reach the bottleneck link.

Consider that the link $L_j$ has cross traffic. Figure 3-6 shows that the cross traffic packet waits in the queue of router $R_{j-1}$. CT is the cross traffic packet.

Figure 3-6 PATH illustration -4
From Figure 3-7 and Figure 3-8, if the cross traffic will not go to link $L_{j+1}$, it will not wait in the same queue of router $R_j$; the intervals between the packet-train packets will become zero again. So, this kind of cross traffic will not affect the result of the PATH algorithm. But it will change the spacing among the probes in standard sender-based algorithms. So, the PATH can get a more accurate estimation than the other sender-based methods under cross traffic.
4. Experiments and Results

We developed our own BAMA (BAndwidth Measurement Algorithms) simulator to help develop and adjust the PATH algorithm. It is also used to compare the performance of the PATH algorithm with other algorithms. Then we used Network Simulator [NS], a widely used network protocol simulation software, to verify and confirm the result.

4.1 BAMA simulator

4.1.1 What is BAMA simulator?

The BAMA simulator is a user-friendly network simulation tool, which is specially designed for testing and tuning of PATH algorithm. The BAMA simulator can implement most of the bottleneck bandwidth estimation algorithms. We have used it to test the PATH, BProbe and Cartouche algorithms. We used Visual Basic to develop this tool. Users can design their probing structure to measure network bandwidth. The BAMA simulator can record the intervals among these probes when they reach the destination and generate an output file.

Compared with Network Simulator, the BAMA simulator is a user-friendly tool. It is much easier to implement the construction of different probes. But, BAMA is specially designed to simulate the network-bandwidth-estimation algorithms. Network Simulator has the ability to simulate various network environments and algorithms.
4.1.2 What does the BAMA simulator look like?

By using the BAMA simulator, in the Figure 4-1, the user can easily set the parameters to implement the bandwidth-estimation algorithms.

![Bandwidth Estimation](image)

Cross traffic information:

- Minimum Packet Size (bytes): 10
- Maximum Packet Size (bytes): 500

Run | Quit

Figure 4-1 BAMA simulator

There are three kinds of parameters in the BAMA simulator: router information, packet information, and cross traffic information.

- **Router/link information**

  In this section, users can enter the following parameters to simulate their algorithms:

  - **The number of routers**
  - **Bandwidth**: The capacity bandwidth (Kbps)
  - **Latency**: The latency of each link (millisecond)
- **Cross traffic FR:** the flow rate of cross traffic (Kbps) from the sender to the receiver. Zero means there is no cross traffic.

- **ACK CTFR:** the flow rate of cross traffic (Kbps) from the receiver to the sender. Zero means there is no cross traffic.

- **Packet information**

  Users can design their own probe schema. For example, user can set:

  - **Packet number:** The number of the packets in each run
  - **Packet Size:** The size of each packet (bytes)
  - **ACK:** If there is an ACK packet coming back to the sender
  - **TTL:** The TTL value in the header of the packets
  - **Times:** How many times the whole packet train (we call it “run”) is sent out from sender.

- **Cross traffic information**

  The minimal and maximum packet size of cross traffic packets (bytes) can be set in this section.

Figure 4-2 explains the flow charts of the BAMA simulator. The BAMA simulator will save all the configuration parameters into a database and generate result files to display the simulation result.
Begin

Initialize the bandwidth, latency, and packet array.

Read router, packet, cross traffic information

Save configuration parameters into database

Calculate arrival time, departure time and available time for each packet and each router

Analyze and calculate the interval between packets

Generate the result file

END

Figure 4-2 BAMA flow chart
4.1.3 BAMA verification

We used Network Simulator [NS] to verify the BAMA simulator. Based on the same parameters, we used the BAMA simulator and Network Simulator respectively to implement the network-bandwidth estimation algorithms, such as BProbe and PATH algorithms. By comparing their estimation results, we can validate the BAMA simulator.

By using the BAMA and Network Simulator, we tested two network bandwidth estimation algorithms: BProbe and PATH.

- **BProbe**

We used BAMA and NS to implement BProbe algorithm respectively. The network parameters can be found in Table 4-1.

<table>
<thead>
<tr>
<th>Link 1</th>
<th>2 M</th>
<th>10ms</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Link 2</td>
<td>622 M</td>
<td>10ms</td>
<td>0</td>
</tr>
<tr>
<td>Link 3</td>
<td>10 M</td>
<td>10ms</td>
<td>0</td>
</tr>
<tr>
<td>Link 4</td>
<td>2 M</td>
<td>10ms</td>
<td>0</td>
</tr>
<tr>
<td>Link 5</td>
<td>5 M</td>
<td>10ms</td>
<td>4.5 M</td>
</tr>
<tr>
<td>Link 6</td>
<td>10 M</td>
<td>10ms</td>
<td>0</td>
</tr>
<tr>
<td>Link 7</td>
<td>16 M</td>
<td>10ms</td>
<td>0</td>
</tr>
<tr>
<td>Link 8</td>
<td>51 M</td>
<td>10ms</td>
<td>0</td>
</tr>
<tr>
<td>Link 9</td>
<td>2 M</td>
<td>10ms</td>
<td>0</td>
</tr>
<tr>
<td>Link 10</td>
<td>1.544 M</td>
<td>10ms</td>
<td>0</td>
</tr>
<tr>
<td>Link 11</td>
<td>10 M</td>
<td>10ms</td>
<td>0</td>
</tr>
<tr>
<td>Link 12</td>
<td>16 M</td>
<td>10ms</td>
<td>0</td>
</tr>
<tr>
<td>Link 13</td>
<td>622 M</td>
<td>10ms</td>
<td>0</td>
</tr>
<tr>
<td>Link 14</td>
<td>2 M</td>
<td>10ms</td>
<td>0</td>
</tr>
</tbody>
</table>

| **Table 4-1 BProbe parameters**

By using Network Simulator, 77.78% of bottleneck-bandwidth estimates are 1.54M, which is the real value of the bottleneck bandwidth. The BAMA simulator gets a
77.78% accurate result too. The histograms of the results are shown in the Figure 4-3 and Figure 4-4. The bin width is 0.068 Mbps. We can see that the distributions of the estimation results are similar and the accuracy of estimation is the same.
Another experiment to verify the BAMA simulator is implementing the PATH algorithm on both simulators. Table 4-2 shows the parameters we used:

<table>
<thead>
<tr>
<th>Link</th>
<th>Bandwidth (bps)</th>
<th>Latency</th>
<th>Cross traffic flow rate (bps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Link 1</td>
<td>2 Mb</td>
<td>10ms</td>
<td>200 k</td>
</tr>
<tr>
<td>Link 2</td>
<td>622 Mb</td>
<td>10ms</td>
<td>200 k</td>
</tr>
<tr>
<td>Link 3</td>
<td>10 Mb</td>
<td>10ms</td>
<td>200 k</td>
</tr>
<tr>
<td>Link 4</td>
<td>2 Mb</td>
<td>10ms</td>
<td>2500 k</td>
</tr>
<tr>
<td>Link 5</td>
<td>5 Mb</td>
<td>10ms</td>
<td>6000 k</td>
</tr>
<tr>
<td>Link 6</td>
<td>10 Mb</td>
<td>10ms</td>
<td>200 k</td>
</tr>
<tr>
<td>Link 7</td>
<td>16 Mb</td>
<td>10ms</td>
<td>200 k</td>
</tr>
<tr>
<td>Link 8</td>
<td>51 Mb</td>
<td>10ms</td>
<td>200 k</td>
</tr>
<tr>
<td>Link 9</td>
<td>2 Mb</td>
<td>10ms</td>
<td>200 k</td>
</tr>
<tr>
<td>Link 10</td>
<td>1.544 Mb</td>
<td>10ms</td>
<td>200 k</td>
</tr>
<tr>
<td>Link 11</td>
<td>10 Mb</td>
<td>10ms</td>
<td>200 k</td>
</tr>
<tr>
<td>Link 12</td>
<td>16 Mb</td>
<td>10ms</td>
<td>200 k</td>
</tr>
<tr>
<td>Link 13</td>
<td>622 Mb</td>
<td>10ms</td>
<td>200 k</td>
</tr>
<tr>
<td>Link 14</td>
<td>2 Mb</td>
<td>10ms</td>
<td>200 k</td>
</tr>
</tbody>
</table>

Table 4-2 PATH Parameters

To estimate the bottleneck bandwidth, Network simulator can get a 97.92% accurate result and the BAMA simulator gets 98.45%. The distributions of the estimation of these two simulators are quite similar. Figure 4-5 and Figure 4-6 show the histogram of the estimation result.

From the experiments we have discussed, the BAMA simulator can get almost the same result as the Network Simulator.
<table>
<thead>
<tr>
<th>Bandwidth (kbps)</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Less than 300</td>
<td>86</td>
</tr>
<tr>
<td>300 - 400</td>
<td>8</td>
</tr>
<tr>
<td>400 - 500</td>
<td>10</td>
</tr>
<tr>
<td>500 - 600</td>
<td>3</td>
</tr>
<tr>
<td>600 - 700</td>
<td>4</td>
</tr>
<tr>
<td>700 - 800</td>
<td>4</td>
</tr>
<tr>
<td>800 - 900</td>
<td>4</td>
</tr>
<tr>
<td>900 - 1000</td>
<td>9</td>
</tr>
<tr>
<td>1000 - 1100</td>
<td>2</td>
</tr>
<tr>
<td>1100 - 1200</td>
<td>0</td>
</tr>
<tr>
<td>1200 - 1300</td>
<td>0</td>
</tr>
<tr>
<td>1300 - 1400</td>
<td>1</td>
</tr>
<tr>
<td>1400 - 1500</td>
<td>3</td>
</tr>
<tr>
<td>1500 - 1600</td>
<td>7866</td>
</tr>
<tr>
<td>Great than 1600</td>
<td>0</td>
</tr>
<tr>
<td>Total</td>
<td>8000</td>
</tr>
</tbody>
</table>

**Figure 4-5** The estimation result of Network Simulator

<table>
<thead>
<tr>
<th>Bandwidth (kbps)</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Less than 300</td>
<td>4</td>
</tr>
<tr>
<td>300 - 400</td>
<td>99</td>
</tr>
<tr>
<td>400 - 500</td>
<td>9</td>
</tr>
<tr>
<td>500 - 600</td>
<td>5</td>
</tr>
<tr>
<td>600 - 700</td>
<td>11</td>
</tr>
<tr>
<td>700 - 800</td>
<td>14</td>
</tr>
<tr>
<td>800 - 900</td>
<td>21</td>
</tr>
<tr>
<td>900 - 1000</td>
<td>10</td>
</tr>
<tr>
<td>1000 - 1100</td>
<td>5</td>
</tr>
<tr>
<td>1100 - 1200</td>
<td>1</td>
</tr>
<tr>
<td>1200 - 1300</td>
<td>0</td>
</tr>
<tr>
<td>1300 - 1400</td>
<td>0</td>
</tr>
<tr>
<td>1400 - 1500</td>
<td>0</td>
</tr>
<tr>
<td>1500 - 1600</td>
<td>8385</td>
</tr>
<tr>
<td>Great than 1600</td>
<td>0</td>
</tr>
<tr>
<td>Total</td>
<td>8564</td>
</tr>
</tbody>
</table>

**Figure 4-6** The estimation result of BAMA simulator
4.1.4 The comparison result

By using the BAMA simulator, we compared PATH with the other algorithms, such as the packet train, BProbe and Cartouche algorithm.

4.1.4.1 PATH without ACK vs Packet train

Let’s first assume that the packet is always in order and will never be lost. We can implement the PATH without ACK by setting up the parameters as shown in Figure 4-7. The PATH without ACK algorithm can get a 99.99% accurate result. When we remove the big header (it becomes the packet-train algorithm), the accuracy percentage drops to 91.66%. If we consider the condition of cross traffic on the other path, as shown in Figure 4-8, the PATH without ACK algorithm can still get a 98.17% result. The packet train gets a 90.3% result.

![Bandwidth Estimation](image)

**Figure 4-7 PATH without ACK**
### Bandwidth Estimation

#### Route Information

<table>
<thead>
<tr>
<th>Link ID</th>
<th>Bandwidth (Mbps)</th>
<th>Latency (ms)</th>
<th>Cross Traffic (Mbps)</th>
<th>ACK (Mbps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>450</td>
<td>10</td>
<td>450</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>1200</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>1000</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>4</td>
<td>1000</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
</tbody>
</table>

#### Packet Information

<table>
<thead>
<tr>
<th>Packet ID</th>
<th>Packet Size (bytes)</th>
<th>ACK Size (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>40</td>
<td>40</td>
</tr>
<tr>
<td>2</td>
<td>40</td>
<td>40</td>
</tr>
<tr>
<td>3</td>
<td>40</td>
<td>40</td>
</tr>
<tr>
<td>4</td>
<td>40</td>
<td>40</td>
</tr>
<tr>
<td>5</td>
<td>40</td>
<td>40</td>
</tr>
</tbody>
</table>

---

**Figure 4-8** PATH without ACK (with other cross traffic)

#### 4.1.4.2 PATH vs Bprobe

We can implement the PATH algorithm by setting up the following parameters:

---

**Figure 4-9** Parameters for the PATH algorithm

39
**BProbe** algorithm parameters can be setup as follows:

<table>
<thead>
<tr>
<th>Router Information</th>
<th>Packet Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Please enter the number of Routers:</td>
<td>Packet number: 11</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Link ID</th>
<th>Bandwidth(Kb/s)</th>
<th>Latency(μs)</th>
<th>Cross Traffic PR</th>
<th>ACK(CHAR)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>450</td>
<td>100</td>
<td>4000</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>1200</td>
<td>15</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>1000</td>
<td>15</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>e</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Packet ID</th>
<th>Packet ID</th>
<th>Packet Size</th>
<th>ACK (N)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 2</td>
<td>4</td>
<td>40 Y</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>40 Y</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>40 Y</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>40 Y</td>
<td></td>
</tr>
</tbody>
</table>

Cross traffic Information:
- Minimum Packet Size(bytes): 40
- Maximum Packet Size(bytes): 1500

![Diagram]

Figure 4-10 Parameters for the BProbe algorithm

In this case, as shown in Figure 4-9 and Figure 4-10, PATH can get 100% accuracy estimation while **BProbe** can get 93.16% accuracy estimations.

Taken into consideration with the cross traffic on the other path, as shown in Figure 4-11, the PATH algorithm can still get 94.54% and **BProbe** gets 87.70% accuracy results.
Figure 4-11 Parameters for the PATH algorithm (with other cross traffic)

4.1.4.3 PATH vs Cartouche

The Cartouche algorithm is similar to PATH without ACK algorithm. The difference between the Cartouche algorithm and PATH is that the Cartouche algorithm discards some probes to enlarge the space between the packet pair. Figure 4-12 shows how the parameters are set up:
Cartouche is a receiver-based algorithm while PATH is a sender-based algorithm. If we assume no packet loss and packet disorder, both PATH and Cartouche can obtain almost 100% accuracy estimations without cross traffic. If we consider cross traffic on the other path, as shown in Figure 4-13, the accuracy of PATH is at 94.85% and that of Cartouche is 94.41%.
Figure 4-13 Parameters for the Cartouche algorithm (with other cross traffic)

4.2 Network simulator

Network Simulator [NS] is based on TCL/TK and C++ languages. It is widely used by many people conducting research in the area of network. We do some experiments to verify packet-pair and PATH without ACK algorithm on Network Simulator. The results can be found in Appendix B.

By using Network Simulator, we compare the PATH and the BProbe algorithms. Table 4-3 shows the network structure and parameters that we simulated:
<table>
<thead>
<tr>
<th>Link</th>
<th>Bandwidth (bps)</th>
<th>Latency</th>
<th>Cross traffic flow rate (bps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Link 1</td>
<td>2 Mb</td>
<td>10ms</td>
<td>200 k</td>
</tr>
<tr>
<td>Link 2</td>
<td>622 Mb</td>
<td>10ms</td>
<td>200 k</td>
</tr>
<tr>
<td>Link 3</td>
<td>10 Mb</td>
<td>10ms</td>
<td>200 k</td>
</tr>
<tr>
<td>Link 4</td>
<td>2 Mb</td>
<td>10ms</td>
<td>2500 k</td>
</tr>
<tr>
<td>Link 5</td>
<td>5 Mb</td>
<td>10ms</td>
<td>6000 k</td>
</tr>
<tr>
<td>Link 6</td>
<td>10 Mb</td>
<td>10ms</td>
<td>200 k</td>
</tr>
<tr>
<td>Link 7</td>
<td>16 Mb</td>
<td>10ms</td>
<td>200 k</td>
</tr>
<tr>
<td>Link 8</td>
<td>51 Mb</td>
<td>10ms</td>
<td>200 k</td>
</tr>
<tr>
<td>Link 9</td>
<td>2 Mb</td>
<td>10ms</td>
<td>200 k</td>
</tr>
<tr>
<td>Link 10</td>
<td>1.544 Mb</td>
<td>10ms</td>
<td>200 k</td>
</tr>
<tr>
<td>Link 11</td>
<td>10 Mb</td>
<td>10ms</td>
<td>200 k</td>
</tr>
<tr>
<td>Link 12</td>
<td>16 Mb</td>
<td>10ms</td>
<td>200 k</td>
</tr>
<tr>
<td>Link 13</td>
<td>622 Mb</td>
<td>10ms</td>
<td>200 k</td>
</tr>
<tr>
<td>Link 14</td>
<td>2 Mb</td>
<td>10ms</td>
<td>200 k</td>
</tr>
</tbody>
</table>

Table 4-3 Network parameters

To implement the BProbe algorithm, the sender sent out 10 probes in each run. These probes will reach the destination server and the ACK packets will be returned. The sender will record these ACK packets when they arrive at the sender. We sent out 1000 runs to test the BProbe algorithm. Figure 4-14 displays the estimation result.

Figure 4-14 shows that, using the BProbe algorithm, $3787 / 8594 = 44.07\%$ of the estimates are in the bin 1535 – 1600 kbps (the bottleneck link bandwidth is 1544 kbps).
<table>
<thead>
<tr>
<th>Bandwidth (kbps)</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Less than 300</td>
<td>29</td>
</tr>
<tr>
<td>300 – 365</td>
<td>169</td>
</tr>
<tr>
<td>365 – 430</td>
<td>38</td>
</tr>
<tr>
<td>430 – 600</td>
<td>89</td>
</tr>
<tr>
<td>495 – 560</td>
<td>21</td>
</tr>
<tr>
<td>560 – 625</td>
<td>18</td>
</tr>
<tr>
<td>625 – 690</td>
<td>1165</td>
</tr>
<tr>
<td>690 – 755</td>
<td>50</td>
</tr>
<tr>
<td>755 – 820</td>
<td>45</td>
</tr>
<tr>
<td>820 – 855</td>
<td>100</td>
</tr>
<tr>
<td>885 – 950</td>
<td>1503</td>
</tr>
<tr>
<td>950 – 1015</td>
<td>34</td>
</tr>
<tr>
<td>1015 – 1080</td>
<td>42</td>
</tr>
<tr>
<td>1080 – 1145</td>
<td>44</td>
</tr>
<tr>
<td>1145 – 1210</td>
<td>45</td>
</tr>
<tr>
<td>1210 – 1275</td>
<td>1067</td>
</tr>
<tr>
<td>1275 – 1340</td>
<td>26</td>
</tr>
<tr>
<td>1340 – 1405</td>
<td>15</td>
</tr>
<tr>
<td>1405 – 1470</td>
<td>294</td>
</tr>
<tr>
<td>1470 – 1535</td>
<td>13</td>
</tr>
<tr>
<td>1535 – 1600</td>
<td>3787</td>
</tr>
<tr>
<td>Great than 1600</td>
<td>0</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>8594</strong></td>
</tr>
</tbody>
</table>

Figure 4-14 BProbe algorithm estimation

The PATH algorithm sends out a large packet header and 10 probes in each run. The header packet will stop after it passes the link 9 and the other 10 probes will be released at the link 10 and stop. ICMP packets will be sent out and the sender will record the time that they arrive at the sender.

We sent out 1000 runs. In the estimates of Figure 4-15, there are \( \frac{8385}{8564} = 97.90\% \) of the estimates in the bin 1535 – 1600 kbps (the bottleneck link bandwidth is 1544 kbps).
<table>
<thead>
<tr>
<th>Bandwidth (kbps)</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Less than 300</td>
<td>4</td>
</tr>
<tr>
<td>300 – 365</td>
<td>92</td>
</tr>
<tr>
<td>365 – 430</td>
<td>7</td>
</tr>
<tr>
<td>430 – 600</td>
<td>6</td>
</tr>
<tr>
<td>495 – 560</td>
<td>6</td>
</tr>
<tr>
<td>560 – 625</td>
<td>3</td>
</tr>
<tr>
<td>625 – 690</td>
<td>5</td>
</tr>
<tr>
<td>690 – 755</td>
<td>16</td>
</tr>
<tr>
<td>755 – 820</td>
<td>3</td>
</tr>
<tr>
<td>820 – 855</td>
<td>21</td>
</tr>
<tr>
<td>885 – 950</td>
<td>10</td>
</tr>
<tr>
<td>950 – 1015</td>
<td>0</td>
</tr>
<tr>
<td>1015 – 1080</td>
<td>5</td>
</tr>
<tr>
<td>1080 – 1145</td>
<td>0</td>
</tr>
<tr>
<td>1145 – 1210</td>
<td>1</td>
</tr>
<tr>
<td>1210 – 1275</td>
<td>0</td>
</tr>
<tr>
<td>1275 – 1340</td>
<td>0</td>
</tr>
<tr>
<td>1340 – 1405</td>
<td>0</td>
</tr>
<tr>
<td>1405 – 1470</td>
<td>0</td>
</tr>
<tr>
<td>1470 – 1535</td>
<td>0</td>
</tr>
<tr>
<td>1535 – 1600</td>
<td>8385</td>
</tr>
<tr>
<td>Great than 1600</td>
<td>0</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>8564</strong></td>
</tr>
</tbody>
</table>

Figure 4-15 PATH algorithm estimates

From the above tests, the PATH algorithm is proven to be more accurate than the BProbe algorithm.

### 4.3 SDSP and SMSP

The simple single-packet algorithm (SMSP) can check the network structure and find the position of the bottleneck link. The standard single-packet algorithm (SDSP) measures the bandwidth and latency of each link on the network. SMSP sends much less traffic than SDSP.
4.3.1 Introduction

To compare the difference of traffic volume that SDSP and SMSP generate, based on the SDSC dataset [Jac97, Dow99a], we develop a tool for Estimation of Traffic Volume for Convergence (ETVOC) to analyze the traffic.

The SDSC dataset is the measurements that Allen B. Downey used for pathchar by using the path from the server rocky.colby.edu (on the campus of the Colby College in Waterville, Maine) to mach5.sdsc.edu (at the San Diego Supercomputer Center). There are 14 links in this path. In the SDSC dataset, pathchar sent out more than 2880 probes of 45 different sizes (from 120 bytes to 1528 bytes) for each link. There are totally more than 43,000 measurements in the SDSC dataset.

To compare the SDSP and SMSP algorithms, using ETVOC, we first send a small number of packets and check the measurements. Based on these measurements, we estimated the position of the bottleneck link (for SMSP) and the bandwidth of each link (for SDSP). If the estimates are not convergent [Dow99a], we will go on sending more packets to get additional measurements. The tool will stop only when either the estimation results are convergent or we reach the maximum probes limit.

To detect convergence, we divided the sample into two parts, odd and even. By using these two sub-samples, we can get three estimates: one from the odd sub-sample, one from the even part, and one from the whole sample. If the difference between the largest
and smallest of these three estimates is less than 10% of the smallest, we will consider the result is convergent.

4.3.2 Development of ETVOC

We used Java language to develop ETVOC to analyze the SDSC dataset and compare SDSP and SMSP. The flow chart of ETVOC is shown in Figure 4-16:

![ETVOC Program flow](image)

Figure 4-16 ETVOC Program flow
4.3.3 Tests using ETVOC

After sending a small number of packets (we call it the first “run”) and checking the measurements, we will send out more packets to get more measurements. There are two ways to send the second run packets:

- Send packets with same size

There are 64 packets for each link in the first run. Their sizes are the same; all are 120 bytes. The size of packets in the following runs will be increased by 32 bytes each time. For example, in the second run, we will send another 64 packets and their size is 152 bytes and so on.

By using this method to implement the tool to compare SDSP and SMSP, we found that the SMSP can get an accurate position from the fourth run. For SDSP, most links can get convergent results after 29 runs. The bandwidth measurements for some links are not convergent even after 45 runs.

- Send packets with different sizes

In this case, in the first run, there are 90 packets at 45 different sizes (from 120 bytes to 1528 bytes) for each link. There are two packets for each size. In the second run, we will send another 90 packets at 45 different sizes for each link, and so on.

The experiments show that SMSP can get an accurate result from the third run while SDSP cannot get convergent results for some links even after 32 runs.
So, from the experiments we discussed above, SMSP consumes many fewer packets than SDSP. The methods based on the standard single-packet algorithm consume a large part of the available network bandwidth and may affect the network performance. The SMSP algorithm reduces the data traffic that PATH generates and improves the efficiency of the PATH algorithm.
5. Conclusions and Future Work

As the Internet grows in complexity and scale, accurate measurement of network performance becomes increasingly important. The goal of bandwidth estimation is estimating network bandwidth practically and accurately.

5.1 Conclusions

We present a new bottleneck link-bandwidth estimation algorithm: PATH. It is a sender-based algorithm and straightforward to implement in the Internet. There are two steps to implement PATH. First, we use SMSP to find out the position of the bottleneck link. Then we send out the PATH probes to measure the bottleneck-link bandwidth. We have developed the BAMA simulator, which is a user-friendly tool to simulate most network bottleneck bandwidth algorithms and to verify and improve the PATH algorithm.

By using another tool called ETVOC, our experiments show that the SMSP algorithm sends much less data traffic volume than the SDSP algorithm. By using BAMA and Network Simulator, we have proven that PATH is more accurate than the existing sender-based algorithms under heavy cross traffic.

5.2 Future work

Most sender-based algorithms don’t work well when the number of nodes is greater than 10. The performance of the PATH algorithm should be better because only the path between the sender and the bottleneck link will affect the estimation. The experiments
that we used to verify the PATH algorithm are based on the simulators (BAMA and Network Simulator). It is very difficult to use simulators to simulate all the cases of the real network, where the forwarding time of the routers may be changed from time to time; the return path could be different from the outgoing path; the path between the sender and receiver could change.

A test-bed may be created and PATH and other algorithms may be tested on it. If required, PATH may be further tuned with these tests. Availability of a test-bed would have been helpful. By collaborating with some large ISPs, production tests could be the final method of tuning PATH.

The tool may be integrated with NAM and WebDedip, the two tools, which are being developed by our research group, the High Performance Grid Computing and Networking Group (HPGCNG) at the University of Windsor. PATH may also be used in a Web Service application or with any other GRID tool-set.
References


[Jac97] V. Jacobson, *Pathchar - a tool to infer characteristics of Internet paths*  
97 MSRI talk

Accepted for publication at Sigcomm 2002, Pittsburg, PA

In Proceedings of ACM SIGCOMM, August 2002

In Proceedings of ACM SIGCOMM'91, pages 3--15, Zurich, Switzerland, September 1991


In Proceedings of the First ACM SIGCOMM 2001 Workshop on Internet Measurement Workshop, San Francisco, California, USA

[Mah01] B. A. Mah, *pchar: A Tool for Measuring Internet Path Characteristics (Website)*  
http://www.employees.org/~bmah/Software/pchar/

[MAM00] K. Matoba, S. Ata, and M. Murata, *Improving Bandwidth Estimation for Internet Links by Statistical Methods*  
In Proceedings of 13th ITC Specialist Seminar, pp. 29.1-29.10, September 2000

[Nie98] Nielsen's Law of Internet Bandwidth (website)  
http://www.useit.com/alertbox/980405.html

[NS] Network Simulator (NS), version 2 (website)  
http://www.isi.edu/nsnam/ns/

[Pax97a] V. Paxson, *End-to-End Internet Packet Dynamics*  
In Proceedings ACM SIGCOMM, pp. 139--152, September 1997

[Pax97b] Vern Paxson, *Measurements and Analysis of End-to-End Internet Dynamics*  
PhD thesis, Computer Science Division, University of California, Berkeley, April 1997
[PD00] L. L. Peterson, and B. S. Davie, *Computer Networks -- A systems approach*  
Publisher: Morgan Kaufmann Co.

In Proceedings of the 1999 USENIX Symposium on Internet Technologies and Systems, pp. 71-79,  
Boulder, CO, October 1999.


[Sco92] D. Scott, *Multivariate Density Estimation: Theory, Practice and Visualization*  
Addison Wesley, 1992

IEEE Network, January/February 2002
Appendix A Network Quantitative Metrics

There are two fundamental quantitative metrics to evaluate the network performance: bandwidth and latency.

1. What is bandwidth?

"A host is a computer capable of communicating using the Internet protocols; includes routers." [HDZ98]. Hosts send or receive data to the other hosts. Every host computer has a unique IP address in the Internet. In an Ethernet network, every host computer has a unique Ethernet address. Store and forward is used for the communication between devices in most networks, such as switches and routers. When the messages are received at the intermediate routing points, they are first recorded (stored) in the devices’ memory and then transmitted (forwarded) to the next routing point.

The bandwidth of a network, also called throughput, “is given by the number of bits that can be transmitted over the network in a certain period of time” [PD00]. For example, the bandwidth of a STS-1 network is 51.840 Mbps, which means that the transfer ability of the network is 51.840 million bits per second.

Bottleneck link bandwidth and available bandwidth are useful for most network applications and users.

- Bottleneck link bandwidth

  The bottleneck link bandwidth of a path is also called path capacity, path bandwidth or bottleneck capacity. In an ideal environment, if there is no other traffic in the path,
bottleneck link bandwidth is the maximum throughput that the path can provide from source to destination. It is limited by the bottleneck link’s underlying capacity. If there are $H$ hops in a path and $C_i$ is the capacity of link $i$, then the bottleneck link bandwidth is

$$C = \min_{i=0\ldots H} C_i$$  \hspace{1cm} (A-1) \hspace{1cm} [DRM01a-Page 1]

For example: if we have four routers from our source computer to destination computer, just like the structure in Figure A-1

```
  Source  Router1  Router2  Router3  Router4  Destination
    |       |       |       |       |
    R1     R2     R3     R4     |
    Link1  Link2  Link3  Link4  Link5
    1.5M    10M    155M   400K   44M
```

Figure A-1 Network structure example

The minimum bandwidth of link 1 to link 5 is link 4. Then the bottleneck link bandwidth is bandwidth of link 4, which is 400 K bit/s

- **Available bandwidth**

Available bandwidth is defined as “the maximum rate that the path can provide to a flow without reducing the rate of the cross traffic”. [JD02a] Simply speaking, available bandwidth is the amount of bandwidth if we consider the cross traffic. In the Internet, it is almost impossible that only one user uses the network. Millions of users from all over the world use it together.
For example, if we use FTP to download some large files, link bandwidth is not the only factor which effects the transfer time. The number of other users who share the same link and their application are also very important. The higher the available bandwidth for a FTP application, the faster would be the file transfer speed.

Generally, available bandwidth depends on link bandwidth and the presence of competing traffic.

2. What is latency?

Latency is another important network performance parameter. It measures how long it takes a message to travel from the source of network to the destination.

For example, if the latency of a local network is 0.44 milliseconds, this means that it takes the message 0.44 milliseconds to travel from the source computer of this network to the destination computer.

Round trip time (RTT) is the time of sending a message from source to destination and back.

Latency consists of three components: Propagation delay, transmit delay and queue delay.

\[
\text{Latency} = \text{Propagation delay} + \text{Transmit delay} + \text{Queue delay}
\]  

(A-2)
• **Propagation delay** is the delay caused by propagation of electromagnetic (EM) waves. In vacuum, EM waves travel with the speed of light. However, the speed of EM waves depends upon medium: it is $3 \times 10^8$ m/s in vacuum and only $2.0 \times 10^8$ m/s in fiber.

\[
\text{Propagation delay} = \frac{\text{Distance}}{\text{Speed of EM waves}} \quad \text{(A-3)[PD00-Page 22]}
\]

• **Transmit delay** is the time to transmit a unit of data.

\[
\text{Transmit delay} = \frac{\text{Size}}{\text{Bandwidth}} \quad \text{(A-4)}
\]

• **Queue delay** is the waiting time in the queue before the packet is processed. Generally, switches in the network need to store and forward the packets in the network. After switches receive and store the packets, they need time to forward these packets.
Appendix B Some Experiments on Network Simulator

We set up network topology in Network Simulator by using TCL:

![Network topology diagram]

Source | L₁ | L₂ | L₃ | Destination
---|---|---|---|---
      | Bandwidth=100M | Bandwidth=10M | Bandwidth=44M |

**Figure B-1 Network topology**

**Experiment one:** validate the basic packet pair algorithm [Kes91].

The source sends out 10 packets with interval $0.1 \times 10^{-6}$ s. The size of these packets is 40 bytes. No cross traffic. From the output file, we determine that the intervals of these small packets $T$ are $3.2 \times 10^{-5}$ s. So the output bandwidth is $40 \times 8 \text{ bit} / 3.2 \times 10^{-5} \text{ s} = 10$M bps. This is the bottleneck bandwidth, the bandwidth of link two. It is agreed with [Kes91].

**Experiment two:** validate the result of PATH without ACK.

The source sends out one 1000 bytes packet and 10 packets (40) bytes with interval is $0.1 \times 10^{-6}$ s. No cross traffic. From the output file, we can get the intervals of these small packets $T$ are $7.244 \times 10^{-6}$ s. So the output bandwidth if $40 \times 8 \text{ bit} / 7.244 \times 10^{-6} \text{ s} = 44$M bps. This is the bandwidth of link three.

So if the size of the header $H$ is much larger than the size of train packet $T$, the packet train with a big packet header can test the bandwidth of any link among the path.
VITA AUCTORIS

Name: Jun Wei

Place of Birth: China

Year of Birth: 1972

Education:
TianJin University, TianJin, China
1990 - 1994 Double BSc.

University of Windsor, Windsor, Ontario, Canada

Honours and Awards:

Chinese High School National Mathematics Competition
The bronze medal, 1990

TianJin University Excellent Student Scholarship
1991 - 1994

University of Windsor Graduate Scholarship
2001

University of Windsor Tuition Scholarship
2002

Ontario Graduate Scholarship
2002