University of Windsor
Scholarship at UWindsor

Electronic Theses and Dissertations

2009

PathAB: A New Method to Estimate End-to-End Available Bandwidth of Network Path

Debashis Roy
University of Windsor

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

Recommended Citation
Roy, Debashis, "PathAB: A New Method to Estimate End-to-End Available Bandwidth of Network Path" (2009). *Electronic Theses and Dissertations*. 335.
https://scholar.uwindsor.ca/etd/335

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.

PATHAB: A NEW METHOD TO ESTIMATE END-TO-END AVAILABLE BANDWIDTH OF NETWORK PATH

by

Debashis Roy

A Thesis
Submitted to the Faculty of Graduate Studies
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

2009

© 2009 Debashis Roy

PATHAB: A NEW METHOD TO ESTIMATE END-TO-END AVAILABLE BANDWIDTH OF NETWORK PATH

By

Debashis Roy

APPROVED BY:

_____

Dr. K. Tepe
Department of Electrical Engineering

_____

Dr. R. Kent
School of Computer Science

_____

Dr. A. K. Aggarwal, Advisor
School of Computer Science

_____

Dr. S. Bandyopadhyay, Chair of Defense
School of Computer Science

May 22, 2009

AUTHOR'S DECLARATION OF ORIGINALITY

I hereby certify that I am the sole author of this thesis and that no part of this thesis has been published or submitted for publication.

I certify that, to the best of my knowledge, my thesis does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my thesis and have included copies of such copyright clearances to my appendix.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office, and that this thesis has not been submitted for a higher degree to any other University or Institution.

ABSTRACT

Estimating available bandwidth accurately is extremely important for many network related applications, especially the ones which need real-time traffic information. With the ever increasing use of Internet, several available bandwidth measurement techniques have been proposed. But most of them assume fluid traffic model, whereas studies show that current Internet traffic follows Poisson distribution. Moreover, very few can operate in stand-alone mode and have relatively high estimation errors. We propose a new method, PathAB, which combines the concepts of three existing algorithms, MoSeab, PoissonProb and PathChirp. It first obtains a rough estimation of available bandwidth using an exponential probing train, and later obtains the final estimate using several Poisson distributed probing trains. It can operate both in client-server and stand-alone modes. Unlike other stand-alone methods, PathAB sends very small echo packets back-to-back after the large probe packets to reduce the cross-traffic effect in returning path as well as the estimation error.

DEDICATION

I would like to dedicate this thesis to my parents and sisters.

ACKNOWLEDGEMENTS

First and foremost, I would like to express my sincere gratitude to my supervisor, Dr. Akshai Aggarwal, for always being a source of motivation and for giving me a definite direction to my research.

I would like to thank my committee members, Dr. Robert Kent and Dr. Kemal Tepe, for their valuable comments and suggestions towards my thesis. I would also like to thank Dr. S. Bandyopadhyay for being the chair in the examination committee.

Last but not the least; I would like to express my gratefulness to my sisters, Deepa and Beeta, for being an unending source of encouragement and confidence during the period of my graduate studies.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

Network measurement techniques continue to receive a great deal of attention since networks are becoming an increasingly important part of today's life. Numerous measurement tools and techniques have been developed to observe or monitor various network characteristics such as link capacity, available bandwidth, transmission delay, transmission loss and network topology etc. The results obtained from these tools have a number of applications in network management such as network troubleshooting, locating fault locations, network provisioning etc. Moreover with the ever increasing use of Internet in various applications, such as audio-video streaming, web applications, distributed database applications, mobile computing etc., estimating the available bandwidth of a network path has become more important. Knowledge of the available bandwidth of an end-to-end path can be used to enhance the performance and QoS of many network related applications, which require real-time traffic information to choose the best route for message transmission.

One important physical characteristic of a large network is the available bandwidth of a network path, which is defined as the maximum rate that the path can provide to a flow without affecting the rate of cross-traffic in the path. Knowledge of real time end-to-end available bandwidth has a variety of applications, such as, end-to-end flow control, in which hosts use end-to-end available bandwidth estimation to determine the rate at which they should transmit the data to avoid congestion in the network. Hosts can dynamically select the server with the highest potential available bandwidth for downloads and streaming media and determine whether the network has enough available bandwidth to meet the desired rate. In peer-to-peer networks, hosts use the available bandwidth information to select peers that can offer the best timely and efficient transfer of content. Network engineers and administrator use bandwidth estimations to troubleshoot networks, reroute network traffic and plan for future network expansions.

In recent years there has been a considerable interest in the research on available bandwidth measurement methods. But measuring the available bandwidth accurately and

efficiently is a challenging task as the value of available bandwidth is highly dynamic in nature. The accuracy of measurement depends on the location of the bottleneck-link and the tight-link in the path, the cross-traffic rate of the path and several other factors. Moreover measurement methods have to take into account the complexity of network topologies, the diversity of traffic models and the probability of dropping measurement packets by the Intrusion Detection Systems (IDS).

1.1. Related Concepts

Before discussing the available bandwidth estimate techniques, it is necessary to clarify some terms and concepts that are very frequently used in network bandwidth related research. The most commonly used terms are explained in this section.

1.1.1. Capacity

Capacity is the maximum transmission rate at which a link can transmit data. It is a physical property of a link and thus does not change with time. A Link's capacity or the maximum transmission rate of data through the link is mainly limited by two factors: the underlying physical transmission medium and the transmitter/receiver hardware. For a multi-hop network path, the link with minimum capacity determines the path capacity C.

$$C = \min_{i=1,2,\ldots,H} C_i \tag{1.1}$$

where, C_i is the capacity of the i-th hop and H is the number of hops in the path.

1.1.2. Bottleneck Link & Bottleneck Bandwidth

In an end-to-end network path, the link with minimum capacity is called the bottleneck link and the capacity of the bottleneck link is called the bottleneck capacity or bottleneck bandwidth or generally the capacity of the path. The bottleneck bandwidth of a path represents the maximum bandwidth that can be available between a sender and receiver through the path, in the absence of competing traffic.

1.1.3. Utilization

Utilization is the portion of capacity that is currently being used by cross-traffic on a hop or a path.

1.1.4. Available Bandwidth

Available bandwidth describes the portion of link capacity that is not being used by the network traffic. It can be obtained by subtracting utilization from capacity. It is the maximum rate at which data can be injected without affecting the cross-traffic. In a multi-hop path the link with minimum available bandwidth determines the available bandwidth of the path.

Let C_i be the link capacity of link i of an end-to-end path having H number of hops. If $\lambda_i(t)$ is the cross-traffic of link i at time t, then the available bandwidth $A_i(t,T)$ of link i is the average of unused bandwidth over some time interval T is given by:

$$A_i(t,T) = \frac{1}{T}\int_t^{T+t} (C_i - \lambda_i(t))dt \tag{1.2}$$

Hence, the average available bandwidth of the path over the time interval T will be $A(t,T)$, which is determined by the link with minimum available bandwidth, is:

$$A(t,T) = \min_{i=1,2,...,H} A_i(t,T) \tag{1.3}$$

Figure 1-1 shows a *pipe model with fluid traffic* representation of a four-hop network path, where each link is represented by a rectangle. The height of each rectangle represents the capacity of the link and the height of shaded portion represents the amount of capacity used by the cross-traffic or the utilization. The height of un-shaded portion represents the available bandwidth of the link. In this example the minimum capacity C_3 determines the end-to-end capacity and the minimum available bandwidth A_4 determines the end-to-end available bandwidth.

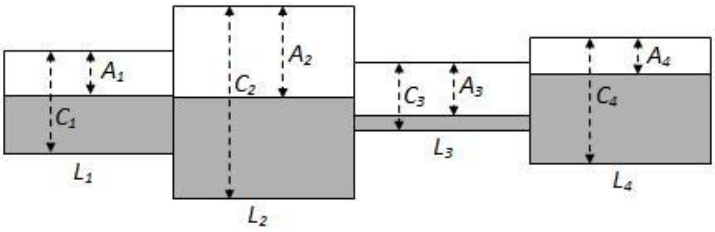

Figure 1-1. A pipe model with fluid traffic for four-hop network path

1.1.5. Tight Link

For a network path, the link with the least amount of available bandwidth is called the tight link. The available bandwidth of the tight link determines the path's available bandwidth. The tight link of a network path may be different from the bottleneck link. In Figure 1-1 link L_3 is the bottleneck link whereas link L_4 is the tight link of the path.

1.1.6. Achievable Bandwidth

Achievable bandwidth is the maximum data transmission rate that an application can actually obtain over a network path. Achievable bandwidth depends on several factors such as, the available bandwidth of the path, the protocol and its implementation, the operating system(s) used, performance capability and the load of end hosts etc.

The difference between achievable bandwidth and available bandwidth is that, achievable bandwidth is an application metric that measures how much throughput an application can achieve, whereas available bandwidth is a physical layer metric that measures how much additional traffic can be injected into the path without interrupting the other network traffic.

1.1.7. Active and Passive Measurement

Available bandwidth measurement techniques can be categorized primarily into active and passive approaches. Active measurement approaches [3−30] inject a series of test packets into the network, and use the feedback information to derive measurement results. Passive approaches [59−61] do not use test packets but rather monitor the packets passing through the routers without interfering with the cross-traffic packets. Active measurement techniques are usually intrusive in nature as some of them send large

number of packets into the network to collect as many samples as possible to filter out the random behaviour of the network. Although passive measurements do not affect the network traffic, they are often less reliable than the active ones. Claffy and McCreary [1] showed that from passive measurements, it might not be possible to extract any useful data at all in some cases. Due to real time and accuracy requirements by most of the applications, available bandwidth estimation methods usually operate in the active mode.

1.1.8. Receiver-based vs. Sender-based Measurement

The active available bandwidth measurement tools can be divided into two categories: client-server based tools (also referred to as receiver based or double end-host tools) and stand-alone tools (also referred to as sender-based or single end-host tools). Typically client-server based tools must be installed in both source host and destination host of the network path; on the other hand stand-alone tools need to be installed only in the source host.

Generally the client-server based tool consists of two programs, the sender program which is installed in the source host and the receiver program or the server program which is installed on the destination host. During estimation process the sender transmits a series of packet-pairs or packet trains at different rates, while the receiver receives the probe packets and uses the timestamp information of all the packets to calculate the AB. It is impossible to deploy the receiver-based algorithm without the destination's cooperation as it requires a server version of the estimation tool to be deployed at the destination. Users normally can install software in their own hosts, but they may not have administrative access to the destination host at the other end of the path. This may prevent the users from installing the receiver program on the destination host and hence may make available bandwidth estimation impossible.

On the other hand, for standalone algorithms, the measurement tool's program is required to be loaded on only the sender host. In this type of algorithm, the sender generally sends a series of ICMP echo-request packets and uses the timestamp information of the received echo-response packets to estimate the available bandwidth.

The standalone available bandwidth estimation algorithms can have several network applications where the sender has limited access or no access to the receiver host. For example, currently several streaming media websites host video or audio in different qualities or bit-rates. The web-sites can decide about the quality and the associated bit-rate to be sent to a user, after determining the available bandwidth from the streaming media host to the user's computer. As the web server may not have any access rights on a user's computer, it may use standalone available bandwidth estimation algorithm to first estimate the AB of the path from web server to the user's computer and then transmit the media of appropriate bit-rate so that the user can enjoy uninterrupted streaming media, without knowing any information about the network.

Almost all client-server based available bandwidth measurement algorithms are based on the following four basic assumptions:

- All routers along the path follow first-in-first-out (FIFO) queuing
- The cross traffic follows the fluid model.
- The cross traffic rate varies slowly and remains constant for the duration of available bandwidth estimation.
- The sender host is able to inject probe packets at a rate higher than the available bandwidth.

In addition to the four, mentioned above, the standalone algorithms are based on three more assumptions:

- The forward path from a sender to a receiver host and the returning path from the receiver to the sender host contain the same set of intermediate routers.
- The cross-traffic along the forward path determines the estimation result; the cross-traffic along the reverse path has a negligible effect on the returning probe packets.
- The receiver host can generate ICMP response packets.

Client-server based algorithms have less estimation error compared to the standalone algorithms as they use the cooperation of the hosts at both ends of the path, but they are less scalable because they need a server version of the measurement software

to be installed at the receiving host. On the other hand standalone algorithms are easy to deploy as they do not require any tool to be deployed at the destination hosts. However most of the stand-alone methods are less accurate than the receiver-based methods.

1.2. Thesis Contribution

In the last two decades a great deal of research has been done on available bandwidth estimation of a network path and a considerable number of algorithms have been proposed. Most of these algorithms use active probing approach and operate only in the client-server mode. The algorithms have been developed based on different theoretical and mathematical foundations and assumptions. All the algorithms pose some advantages but with some drawbacks. For example, some algorithm may perform better on high link utilization but it may fail under low traffic scenario. This thesis first presents a comprehensive survey of existing available bandwidth measurement algorithms and then proposes a new available bandwidth estimation algorithm PathAB which has been developed combining the concepts used in three different methods and can operate both in client-server mode and in standalone mode.

The rest of this thesis is organized as follows: Chapter II presents a comprehensive survey of existing available bandwidth measurement techniques. Chapter III gives a detailed description of the proposed algorithm PathAB, and its operation in client-server mode as well as in standalone mode. In Chapter IV we present the experimental results and analysis to verify the performance of PathAB and compare it with some existing methods such as IGI, Pathload, PathChirp, PoissonProb and Spruce. We have performed the comparison using extensive simulations in NS2 as well as on network test-bed under different traffic loads for both single-hop and multi-hop paths. Finally the future work and conclusion are presented in Chapter V.

CHAPTER II

SURVEY OF AVAILABLE BANDWIDTH ESTIMATE ALGORITHMS

All the existing available bandwidth measurement algorithms can be classified mainly in two categories: gap-based and rate-based algorithms. But because of different measurement approaches and network models used by different researchers in this survey the available bandwidth measurement algorithms have been divided into six categories. The six categories are: gap-based, rate-based, model-based, probabilistic, hybrid and Kalman filtering based approach.

Carter and Crovella [2] were the pioneer of available bandwidth measurement techniques. They introduced the first algorithm *cprobe*, a gap-based method, which estimates the available bandwidth based on the dispersion of long packet trains at the receiver. A similar approach is taken in *pipechar* [3]. Strauss *et al.* [4] introduce *spruce* which focuses on measurement accuracy, failure patterns, probe overhead and implementation issues of bandwidth measurement techniques. Kazantzidis *et al.* [5] use a new sampling formula to sample the probing packets in algorithm *ab-probe* introduced by them. Xuan and Zheng [6] introduce a new available bandwidth measurement algorithm *PoTRI*, that uses tri-packet-probe instead of packet-pair used in all other gap-based technique.

Most of the researchers have preferred rate-based approach to estimate available bandwidth and proposed various rate-based available bandwidth measurement algorithms. Melander *et al.* [7] [8] propose the technique *TOPP* which addresses the hidden bottleneck problem in the network path. NEPRI [9] focuses on the macroscopic behaviour of the probing packet queued at the bottleneck link. He *et al.* [10] introduce a measurement method which uses a curve matching technique to estimate the available bandwidth. Jain and Drovolis [11] [12] propose a new rate-based measurement method “Self Loading of Periodic Streams” and implements this method in a tool called *Pathload*. *PathChirp* [13] is based on the concept of “self-induced congestion” and uses exponentially spaced chirp probing train. The *PathMon* algorithm introduced by Kiwior *et al.* [14] calculates mean and standard deviation of inter-arrival jitter prior to bandwidth

measurement to improve accuracy of estimate of the curve matching technique. *Pathtrait* proposed in [15] uses three types of probing packets in the probing train and uses linear regression for bandwidth calculation. Xin [16] suggests a technique, *PoissonProb*, where the intervals between the probing packets are in Poisson distribution format. Kola and Vernon [17] propose a fast estimate method, *QuickProbe*, which calculates the available bandwidth in only two roundtrips with moderate accuracy. Xiao *et al.* [18] proposed a new algorithm which is based on Pathload's concept but uses exponential search instead of binary search for fast estimate and compares the average interval difference of source and received trains, rather than comparing the rates. The *eChirp* algorithm introduced by Suthaharan and Kumar [19] uses the concept of exponential packet trains used in PathChirp but increases the inter-packet intervals by even powers. The algorithm combines three different sub-trains within a packet train to obtain more information about the network path.

Some researchers have used model-based approaches to measure available bandwidth. The *Delphi* algorithm [13] uses the multifractal wavelet model introduced by the same authors in an earlier paper [20]. Hu and Steenkiste [21] develop a single-hop gap model for the competing cross-traffic and based on this model they introduce two available bandwidth measurement algorithms, *IGI* and *PTR*. Kang *et al.* [22] introduced an algorithm based on a stochastic queuing model for single congested path. Bhati [23] extends the previous idea to design a recursive queuing model for multiple congested links and presents the algorithm envelope.

Almost all the algorithms fail to correctly estimate the available bandwidth when the network utilization is very low. To overcome this problem two groups of researchers proposed algorithms based on probability and statistics. Min *et al.* [24] proposed a new probabilistic definition of available bandwidth and based on this, they introduced the *SMART* algorithm which, unlike all other methods, uses randomly distributed probing packets. Zhou *et al.* [25] proposed another probabilistic approach NBE to estimate the available bandwidth of a low utilization path. They have also established a new metric to calculate the busyness of the path and based on this metric, the authors have proposed a

new method *A_ABE* which dynamically uses NBE or IGI algorithm to estimate the available bandwidth.

Both gap-based and rate-based algorithms have some advantages as well as drawbacks which are described in section 2.5. To utilize the benefits of both of these approaches some researchers have proposed hybrid algorithms. Botta *et al.* [26] proposed a hybrid available bandwidth estimate tool called *BET* which integrates the three different concepts, the Packet Train Dispersion (PTD) technique of path capacity estimate methods, SLoEC (Self Loading Exponential Chirp) used by the PathChirp algorithm and the SLoPS (Self Loading of Periodic Streams) used by the Pathload algorithm. *MoSeab* [27] on the other hand uses several probing train with increasing rate in the first phase (rate-based) to get a rough estimate of available bandwidth and in the next phase it uses a gap-based approach for final estimate.

BART [28] and Abest [29] are the only two algorithms which use Kalman filtering method to estimate the available bandwidth. The only difference between them is that BART algorithm transmits probe packets at a rate higher than the available bandwidth and hence overloads the path. Abest on the other hand sends probe packets at a lower rate than AB without congesting the network path.

The following section briefly describes the concepts and measurement approaches of each of these available bandwidth estimation algorithms.

2.1. Gap-based Approach

Gap-based algorithms are usually facilitated by packet pair/train properties. They use the information about the time gap between the arrivals of two successive probes at the receiver. *"The advantage of this kind of algorithms is that they are very sensitive to the burstiness of cross-traffic because of fine-grained interaction between the probing packets and cross-traffic packets"* [16]. The main idea of gap-based approaches is that, if a pair of probe packet of size q is sent across a path of tight link capacity C with time gap Δ_{in}, such that Δ_{in} is not greater than q/C, then the cross-traffic packets will be queued up behind the first packet of the pair while it is being processed by the tight link. As a result

when the packet pair reaches the receiver, the output time gap Δ_{out} will be greater than the input time gap. Therefore, Δ_{out} is the time taken by the tight link to transmit the second probe packet in the pair and the cross traffic that arrived during Δ_{in} as shown in Figure 2-1.

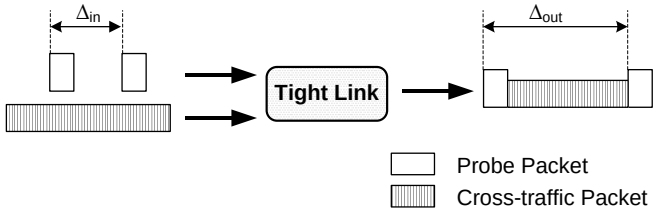

Figure 2-1. Gap-based Measurement

Thus the time to transmit traffic is $\Delta_{out} - \Delta_{in}$, and the rate of cross traffic is, $(\Delta_{out} - \Delta_{in})/\Delta_{in} \times C$, where C is the capacity of the bottleneck. The available bandwidth is:

$$A = C \times \left(1 - \frac{\Delta_{out} - \Delta_{in}}{\Delta_{in}}\right) \tag{2.1}$$

Most of the gap-based methods make the following assumptions: (i) a single bottleneck link, (ii) the bottleneck link to be the tight link of the path and (iii) the router queue does not become empty between the departure of the first probe in the pair and the arrival of the second probe.

2.1.1. Cprobe

Carter and Crovella [2] introduced the first algorithm *cprobe* to measure end-to-end available bandwidth. The measurement technique of cprobe is straightforward, it sends a short stream of echo packets, records the time between the receipt of the first packet and the receipt of the last packet, and then divides the number of bytes sent by this time to measure the available bandwidth. The underlying assumption is that the dispersion of long packet train is inversely proportional to the available bandwidth. The authors state that this method is applicable when the packets are sent at a higher rate than the bottleneck link speed, which can be measured using a separate method bprobe introduced by the authors in the same paper. Cprobe uses the results of four separate 10-packet

streams in order to tolerate packet drops and the possibility of re-ordering of packets. To eliminate some irregularities in the readings, cprobe discards the highest and the lowest inter-arrival measurements while calculating available bandwidth.

2.1.2. Pipechar

The algorithm *Pipechar* is proposed by Jin *et al.* [3] and it is implemented in the tool Network Characterization Services (NCS). It uses the same basic assumption about dispersion of long packet train like *cprobe*. The only difference is that *pipechar* can also operate in the passive mode through the deployment of NCS daemons on each subnet of the network infrastructure.

Though the algorithms cprobe and pipechar are straightforward, researchers are doubtful about some assumptions of these approaches. According to Dovrolis *et al.* [30] *"the dispersion of long packet train does not measure the available bandwidth in a path; instead, it measures a different throughput metric which is referred to as the asymptotic dispersion rate (ADR)"*.

2.1.3. Spruce

Spruce [4] algorithm uses a series of packet-pairs to estimate available bandwidth. It assumes single bottleneck link and bottleneck capacity C to be known. Spruce uses 1500 byte probe packets and sets the intra-pair time gap Δ_{in} to the transmission time of a probe packet on the bottleneck link. The main characteristic of spruce is that it sets the inter packet-pair gaps as Poisson distribution with an average τ which is much larger than Δ_{in}, so that it becomes less intrusive. For each packet-pair spruce calculates the available bandwidth using (2.1). By default it takes an average of 100 such samples to report the final estimate of available bandwidth. Authors claim that the value of τ is chosen in a way such that the average probe rate is within 5% of bottleneck capacity and the estimate error is less than 30% in almost all cases.

2.1.4. ab-probe

The ab-probe method was proposed by Kazantzidis *et al.* in [5]. Unlike existing gap-based methods, instead of calculating available bandwidth as the ratio of packet size and the inter-arrival time of two successive packets (referred to as "bytes over time", BoT), the authors suggested a new sampling formula for the probe packets in ab-probe.

Ab-probe sends multiple streams of N packets each of size S at equal time intervals assuming that the packets reach the bottleneck link with input rate P_b. The available bandwidth for each stream is calculated using the following equation:

$$A = C - \frac{C \times T - (N-1) \times S}{(N-1)\frac{S}{P_b}} \tag{2.2}$$

Where, C is the bottleneck capacity and T is the observed time separation between the first packet and the N-th packet at the receiver. Ab-probe takes the average of the available bandwidths calculated for all the streams to estimate the available bandwidth of the path. The nettimer tool is used to measure the bottleneck bandwidth prior to ab-probe.

The authors state that they have tested their algorithm on both long range and short range internet connections using both packet-pairs and packet-trains method. They claim that ab-probe can successfully measure the available bandwidth in all cases, even for long distance network with more than 20 hops, whereas the existing BoT techniques may sometimes fail.

2.1.5. PoTRI

Xuan and Zheng [6] introduced a new gap-based technique, *PoTRI* (PriOritized TRI-packets), to measure available bandwidth. Unlike all other gap-based methods, it sends tri-packets probes to measure the utilization of the link and the middle one packet of the tri-packets-probe is prioritized so that it can measure both the output time gap and the waiting time of the probe. According to the authors existing probe-gap-model only captures the competing cross traffic packets that are inserted between a probe packet pair, but cannot measure the packets that are already in the queue before the packet-pair

arrives which is a usual scenario for heavy cross traffic condition. The authors state that if a probe packet pair, with the second packet highly prioritized is transmitted-back to-back, when they arrive at the router, the second one will immediately go to the head of the service queue due to its high priority while the other one will wait at the end. Therefore the output gap of the two probe packets denote the waiting time in the queue. Based on this principle PoTRI sends three packets P_1, P_2 and P_3 in each probe and the prioritized packet P_2 is sent closely behind P_1. The first two packets P_1, P_2 are used to measure the mean waiting time in the queue and the other two packets P_1 and P_3 are used to measure the mean transmission time from the difference of their output and input gaps. This information is then used to accurately calculate the overall utilization as well as the available bandwidth of the link.

According to the authors, PoTRI's estimate for available bandwidth is quiet accurate for heavy cross traffic, but is unstable for low network utilization. Moreover, PoTRI needs the network facilities to support priority settings. If all routers in the network path do not support priority settings, the PoTRI becomes a usual probe gap method.

2.1.6. Summary

The advantage of the gap based algorithms is that they are less intrusive. Most of the gap-based methods, except ab-probe, use series of packet-pairs. As a result the overall probing rate can be kept very low by increasing inter packet-pair time gaps. But the main problem with these methods is that these methods assume that the bottleneck capacity of the path is known and that the bottleneck and the tight link are the same. This assumption makes these methods unusable to measure the available bandwidth of a completely unknown path. Also Xuan and Zheng [6] pointed out that existing gap-based approaches cannot capture the effect of cross-traffic packet that are already present at the router's queue. As a result, under high traffic utilization, they under-estimate the amount of cross-traffic and over estimate the available bandwidth; though they have satisfactory performance under low utilization. PoTRI is the first gap-based approach which tries to capture the effect of queued traffic packets along with the competing traffic, but it has a

special requirement that all the routers of the path have to support priorities. Table 2-1 presents a summary of gap-based methods discussed in this section.

Table 2-1. Summary of Gap-based Algorithms

Algorithm	Year	Main Contribution/Feature	Sender Based
Cprobe [2]	1996	First algorithm to measure available bandwidth	Yes
Pipechar [3]	2001	Similar to *cprobe* but can operate both in active and passive mode	No
Spruce [4]	2003	Interval between the packet-pairs are set in Poisson distribution format	No
ab-probe [5]	2003	Use of packet trains instead of packet-pairs. Introduces a new available bandwidth sampling formula	No
PoTRI [6]	2006	Use of tri-packet-probe with a prioritized central packet to capture the effect of traffic packets queued at the router	No

2.2. Rate-based Approach

Most of the researchers have preferred rate-based approach to measure the available bandwidth of a network path. This type of algorithms are based on the concept of self-induced congestion: *"If one sends probe traffic at a rate lower than the available bandwidth along the path, then the arrival rate of probe traffic at the receiver will match their rate at the sender. In contrast, if the probe traffic is sent at a rate higher than the available bandwidth, then queues will build up inside the network and the probe traffic will be delayed. As a result, the probes' rate at the receiver will be less than their sending rate"* [4]. Thus, the available bandwidth can be measured by searching for the turning point at which the probe sending and receiving rates start matching.

The advantage of rate-based algorithms is that they adapt widely to most of the network scenarios. They have better resistance to the cross-traffic effect and they can always report reasonable results. *"In comparison to the rate-based algorithms, the gap-based algorithms may deviate largely from the correct value because of the errors in estimating either the bottleneck capacity or the cross-traffic rate. The shortcoming of the*

rate-based algorithm is that the network overhead to converge to the turning point is too high" [16].

2.2.1. TOPP

Melander *et al.* proposed the measurement methodology TOPP to estimate the available bandwidth of a network path [7, 8]. TOPP sends many packet pairs at gradually increasing probing rates from sender to the target host. Suppose that a packet pair, each packet having a size of L bytes, is transmitted through a link of capacity C with inter packet interval Δ; thus, the offered rate of the probing packet-pair will be $R_O=L/\Delta$. If R_O is more than the end-to-end available bandwidth A, the link will become overloaded. Under this situation, if FCFS scheduling and random dropping of packets at buffer overflow is assumed, then the probe traffic will get a share of the link bandwidth proportional to the offered rate R_O and this is measured by the receiver as $R_m < R_O$. On the other hand if $R_O < A$, TOPP assumes that the packet pair will arrive at the receiver at the same rate as it had at the sender (i.e., $R_m = R_O$).

$$R_m = \begin{cases} R_O & \text{if } R_O \le A \\ \dfrac{R_O}{R_O + R_C} C & \text{if } R_O > A \end{cases} \tag{2.3}$$

where, $R_C = C - A$ is the average cross-traffic rate of the link. Equation (2.3) can be re-written as:

$$\frac{R_O}{R_m} = \begin{cases} 1 & \text{if } R_O \le A \\ \left(1 - \dfrac{A}{C}\right) + \dfrac{1}{C} R_O & \text{if } R_O > A \end{cases} \tag{2.4}$$

TOPP sends several trains of packet-pairs consisting of n pairs in each train with linearly increasing input rates for the trains. TOPP estimates the available bandwidth A to be the maximum possible input rate such that $R_O \approx R_m$. Equation (2.4) is used to estimate the capacity C from slope of R_O/R_m vs. R_O plot.

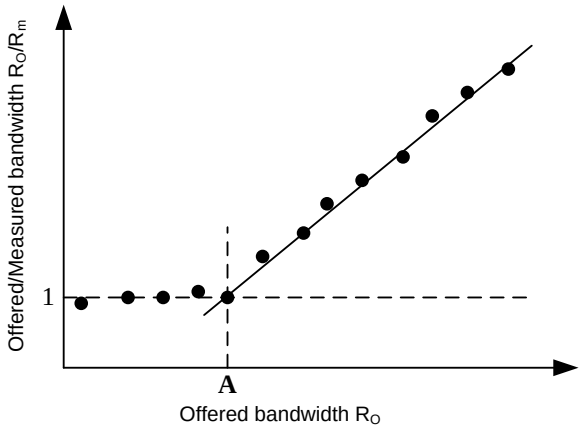

Figure 2-2. Offered bandwidth over measured bandwidth in TOPP for single-hop path

For a path consisting of multiple links, the R_O/R_m curve may show multiple slope changes due to queuing of probing packets at links having higher available bandwidth than A. To avoid this situation TOPP assumes that congested links are in Smallest Surplus First (SSF) order.

2.2.2. AB Estimate using Curve Matching

He *et al.* [10] proposed a new available bandwidth estimate method which uses curve matching technique. The proposed method sends trains of ICMP echo packets with decreasing time delays between two consecutive packets so that each packet requires higher bandwidth than the previous one. For each packet the transmission time and the reception time is noted. It then compares the curve for sending probe packets (sending curve) with the one for receiving acknowledgement packets (receiving curve). The sending curve is plotted using the transmission time against the packet number and the time of the first packet is set to 0. Similarly the receiving curve is plotted using the reception time against packet number with the time of first packet aligned to 0. To compensate the fluctuations in the receiving curve caused by burstiness of traffic, the method uses trend lines of receiving curve. The point where the trend line of receiving curve starts diverging from the sending curve is reported as the congestion point and the bandwidth requirement at that point is used to calculate the available bandwidth. To improve the correctness of result the method uses several packet trains. Once it finds the

congestion point, it automatically shrinks the bandwidth range around the estimated congestion point and probes the network again.

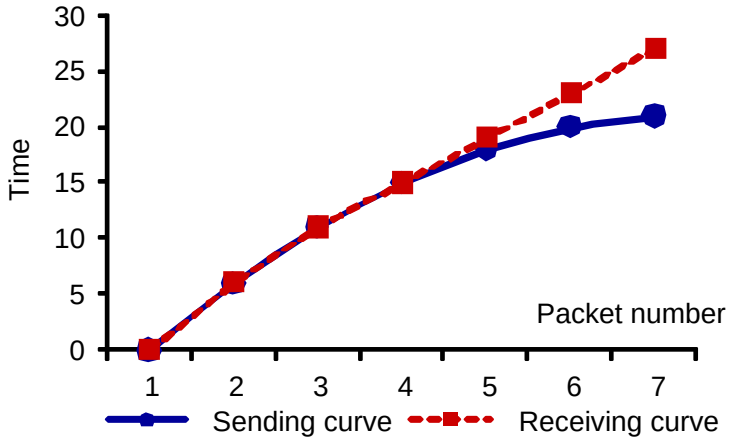

Figure 2-3. Sending curve Vs. Receiving curve

According to the authors this algorithm can calculate available bandwidth below 10Mbps with any desired accuracy. But for higher accuracy it requires more number of probing trains which increase the network overhead.

2.2.3. Pathload

Jain and Drovolis introduced the Pathload tool in [11] & [12]. Pathload uses Self-Loading Periodic Streams (SLoPS) to measure the available bandwidth. The basic idea of Pathload is that, if the stream rate R is greater than the available bandwidth A of the network path, the stream will cause a short term overload in the queue of the tight link. As a result the probe packets of the stream will queue up at the tight link and the One-way Delays of the probing packets will keep on increasing. On the other hand, if the stream rate is less than or equal to the path's available bandwidth, the one-way delays of the packets do not change.

In this method the source periodically sends streams of $K \approx 100$ equal-sized packets to the receiver at a certain rate R. Each packet of the stream is time-stamped and at the receiver *One-Way Delay (OWD)* for each packet is calculated. Pathload uses an iterative algorithm, similar to binary search mechanism, to bring the stream rate R closer to the available bandwidth of the path. Instead of reporting a single value for path's

available bandwidth, Pathload gives a range (AB_{min}–AB_{max}) in which the Available Bandwidth belongs. It uses several probe streams to narrow down the range. Assume that the sender sends the n th probe stream with rate $R(n)$. From the delay behavior of the received packets the receiver decides whether $R(n)>A$ or not and informs the sender. The sender then estimates the rate of the next probing stream $R(n+1)$ using the following method:

If, $R(n) > A$, $R^{max} = R(n)$

If, $R(n) \leq A$, $R^{min} = R(n)$

$R(n+1) = (R^{max} + R^{min})/2$

Initially R^{min} is set to zero and $R(n)$ & R^{max} both are kept same and sufficiently large so that $R(n) = R^{max} > A$. The algorithm terminates when $(R^{max}–R^{min})<\omega$, where ω is user defined estimate resolution. The algorithm needs $log_2\ (R(0))$ probing streams to converge.

The Pathload method assumes that there is zero packet loss at the bottleneck router, which means the router queue is large enough so that no cross-traffic packet is dropped during the probing. If this assumption is not satisfied, Pathload may underestimate the cross-traffic rate and over estimate the Available Bandwidth.

2.2.4. PathChirp

PathChirp is a novel available bandwidth estimate method introduced by Riberio *et al.* in [13]. Unlike all earlier measurement techniques it uses exponentially spaced probing packets in train to estimate path's available bandwidth. The inter-packet gaps within a chirp decreases exponentially by a factor γ resulting in a rapid increase of probing rate within each train.

At the receiver, PathChirp observes the queuing delay signature of the received packets for each train. Because of the burstyness of cross-traffic the delay signature consists of some excursions from the zero axis instead of monotonous increase in queuing delay.

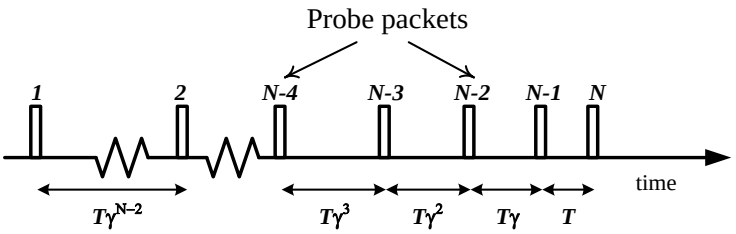

Figure 2-4. Exponentially distributed packets in PathChirp probe train

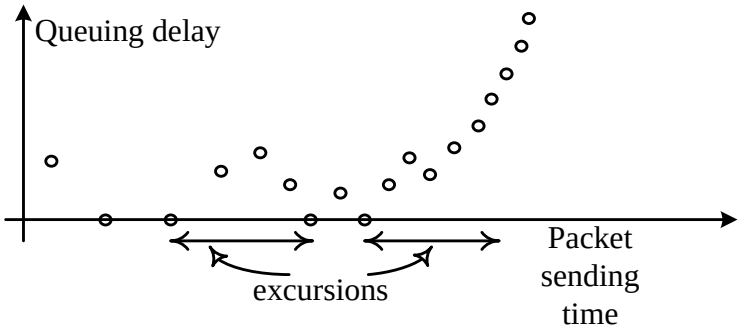

Figure 2-5. PathChirp queuing delay signature

The sender transmits *M* chirps each containing *N* exponentially separated packets. It first estimates per packet available bandwidth (E_k) for each packet *k* as follows:

i) $E_k = R_k$ if *k* belongs to an excursion that terminates and $q_k \leq q_{k+1}$

ii) $E_k = R_l$ if *k* belongs to an excursion that does not terminate, where *l* is the start of the excursion

iii) $E_k = R_l$ for all other cases

where, q_k is the queuing delay and R_k is the instanteneous rate of *k* th packet in the train. It then takes a weighted average of all the $E_k^{(m)}$'s to estimate per-chirp available bandwidth $D^{(m)}$ using equation:

$$D^{(m)} = \frac{\sum_{k=1}^{N-1} E_k^{(m)} \Delta_k}{\sum_{k=1}^{N-1} \Delta_k} \tag{2.5}$$

where, Δ_k is the inter-spacing time between packets *k* and *k+1*. Finally, by averaging all the estimates of $D^{(m)}$, it calculates the available bandwidth of the path.

The main advantage of PathChirp is that to probe a network over the range of rates [G_1, G_2]Mbps it requires only $log(G_2) - log(G_1)$ packets.

2.2.5. PathMon

PathMon is another algorithm, introduced by Kiwior *et al.* [14] to estimate available bandwidth, which follows almost similar curve matching technique inspired by the AB Estimation using Curve Matching method proposed by He in [10]. But to eliminate insignificant data and fluctuations of measurement, the algorithm first uses a single packet-train with a simple statistical evaluation.

The algorithm has two steps. In the first step, which is the jitter measurement step, PathMon sends one packet-train containing a series of N_j equally-spaced packets of the same size. The receiver collects the inter-arrival time gaps and uses statistical analysis to calculate the mean interval jitter, the standard deviation. It sends a large enough number of packets to obtain a good statistical sample of jitter.

In the second step, the algorithm sends a series of equal-sized packets, but with decreasing time interval, so that the instantaneous bandwidths of the packets are in increasing order and equally spaced between the lower and upper bounds of available bandwidth. The receiver records the receiving times of the packets in terms of cumulative time. PathMon calculates the available bandwidth by identifying the congestion point, i.e., the point of divergence between the inter-packet delays measured at the sender and the receiver.

PathMon takes a different approach from the method proposed in [10] to recognize the congestion point. It identifies the congestion point by starting at the upper bound endpoint and traversing backwards over the timestamp information for each packet in the train comparing the measured delay to the measured jitter statistics. The congestion point corresponds to the packet that has a time difference greater than the average jitter but is preceded by a packet with a time difference less than the average jitter.

2.2.6. Pathtrait

The *Pathtrait* method introduced in [15] can accurately locate the tight link and estimate the end-to-end available bandwidth of a network path. The method is based on a novel probing technique that uses three different types of probing packets in the probing train.

According to the authors, pathtrait technique is based on the assumptions that all the routers along the path follow FIFO queuing and generate ICMP packets and the cross traffic along the path follows a fluid model. Pathtrait uses three different types of packets, the Type-I packet can successfully reach the destination from the origin, Type-II packets are hop limited by setting a lower value for the TTL so that it is dropped at an intermediate router and Type-III packet which is hop limited ICMP packet that can generate ICMP response from an intermediate router. Pathtrait train consists of large load packets (Type-II) of size 1000 bytes, each of which is followed back to back by one backward packet (Type-III) or one forward packet (Type-I) of size 40 bytes alternatively. The Type-I packets are used to estimate the forward rate or output rate of a hop and Types-III packets are used to estimate the input rate or backward rate for the hop.

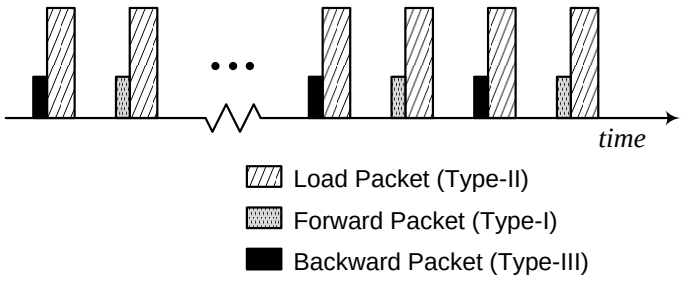

Figure 2-6. Pathtrait train structure

The method operates in three steps. In the first step Pathtrait sends a train with TTL 128 and finds the hop count of the path from the received TTL value and determines the maximum probing rate. The second phase is for locating the tight link. For each hop of the path it sends a pathtrait train with adjusted TTL value, and reports a link as tight link if the difference between the forward rate and backward rate is less than 5% of backward rate. After discovering the tight link the method proceeds to step three to

estimate the available bandwidth. In this step it probes the network path with 15 trains with different rates calculated as:

$$R_i = (1-(8-i)\varepsilon)R \tag{2.6}$$

where, R is probing rate used in locating the tight link, R_i is the rate of i-th probing train and the value of ε is set to 2%. After obtaining the receiving rates of all the trains, it uses linear regression to solve (2.7) in order to obtain tight link bandwidth C_t and cross traffic rate λ.

$$\frac{1}{R_O} = \begin{cases} \frac{1}{R_I} & \text{if } R_I < A \\ \frac{1}{C_t} + \frac{\lambda}{C_t}\frac{1}{R_I} & \text{if } R_I \geq A \end{cases} \tag{2.7}$$

where, R_I are R_O are the input and output rate respectively. Finally pathtrait calculates the available bandwidth A of the path as $C_t - \lambda$.

The authors state that they have verified Pathtrait estimate using NS2 simulation environments and found that this method accurately identifies the tight link location in both constant cross traffic environment and in bursty environment. However the available bandwidth estimate is less accurate in bursty traffic condition.

2.2.7. PoissonProb

The PoissonProb algorithm was introduced by Xin in [16]. The algorithm was designed based on the study in [31] that, current network traffic on the internet follows Poisson distribution. The key concept of this method is that in a probe stream, the intervals between probe packets are in Poisson distribution format.

PoissonProb can operate both in client-server mode (receiver-based) and in stand-alone mode (sender-based). In client-server mode, PoissonProb opens two connections between the server and the client, one TCP session, which is used for transferring control information, and a UDP session, which is used for probe packet transmission. In the first phase of measurement PoissonProb client sends probe packets back-to-back to the server

to estimate the bottleneck capacity of the path using histogram analysis of the timestamp information of the packets. In the next phase the client sends a train of Poisson distributed packets with mean inter-packet interval λ, set to 1/3 of the bottleneck separation gap. At the receiver, the average destination gap of the packets within a probing train is compared with the average source gap. If both gaps are the same ((*source gap-destination gap*)*/destination gap* $\leq$ 0.15), PoissonProb stops measurement and reports available bandwidth based on the probing rate of that train. Otherwise, it increases or decreases the value of λ by a factor of 1/5 and proceeds with the next round of measurement.

In the stand-alone or sender-based mode, PoissonProb requires only a UDP session between the sender and the receiver and sends UDP echo packets in Poisson distribution. The algorithm assumes that the packets are echoed back through the same route without being affected by the cross-traffic. The measurement strategy is similar to the client-server mode. The sending host observes the total initial gaps and the total gaps of the echo packets and stops measurement on reaching the turning point.

The main assumption of PoissonProb algorithm is that the network traffic pattern follows Poisson process. If the traffic pattern changes, this method fails to estimate the available bandwidth correctly.

2.2.8. QuickProbe

Kola and Vernon [17] introduced a rapid available bandwidth measurement technique, QuickProbe, which can estimate the available bandwidth in only two roundtrips. QuickProbe uses 19 probe packets on the first roundtrip to get a conservative estimate of the available bandwidth and then another 9-17 packets on the second roundtrip to refine the estimate.

According to the authors QuickProbe method sends a fixed-length train of maximum-size packets with fixed spacing. The sending rate is considered to be feasible if receiving rate of the probe packets is within 10% of the sending rate. QuickProbe uses two initial packet-pairs with two probe rates (6 Mbps and 80 Mbps) to measure bottleneck capacity of the path. It then uses this capacity information and initial probe

rate feasibility results to determine the next probe rate in order to perform a binary search for the maximum feasible transmission rate, which is reported as the available bandwidth, similar to the Pathload approach. The key difference is that QuickProbe uses trains of 9 packets (or 17 packets if the probe rate is more than 100 Mbps) unlike 100 packets per train in Pathload. This significantly reduces the traffic overload and estimation time. According to the authors, QuickProbe may underestimate the available bandwidth in some cases due to the granularity of binary search.

2.2.9. Algorithm proposed by Xiao et al.

Xiao *et al.* [18] proposed a new available bandwidth measurement method which is based on the Self-Loading of Periodic Stream (SLoPS) concept introduced in Pathload [11]. Instead of comparing the received probe rate with the sending rate the proposed method uses a new technique, called interval difference, to infer the congestion. Also unlike Pathload instead of only using binary search method, it first performs an exponential search to quickly find the rough range and then performs binary search to search for the actual range of available bandwidth.

In each probing train the method transmits m^2+1 probe packets of size L=100 bytes, where m is any integer. At the receiver the m^2 received intervals are separated into m groups. For example, the i-th group is $\{O_{m\times i}, O_{m\times i+1}, \dots, O_{m\times i+m-1}\}$ whose average is O_i^a, where $0 \le i < m$ and O_k is the received gap between k-th and (k+1)th packet. For each train, the value of a parameter δ is calculated using the following formula:

$$\delta = \frac{\sum_{i=0}^{m-1} F(X)}{m} \quad , F(X) = \begin{cases} 1 \, , \, O_i^a > \Delta_s \\ 0 \, , \, O_i^a \le \Delta_s \end{cases} \tag{2.8}$$

where, $\Delta_s=L/R$, R is the sending rate, Δ_s is the sending interval and Δ_r is the estimated receiving interval of the probing train. If $\delta \le 0.2$ the algorithm reports $\Delta_s=\Delta_r$ otherwise $\Delta_s<\Delta_r$.

To obtain the rough range of available bandwidth the algorithm sends several probing trains with exponentially increasing probing rate. For each train first the interval

difference between Δ_s and Δ_r is estimated. The rate of (n+1)-th probing train $R(n+1)$ is calculated as:

If, $\Delta_s<\Delta_r$, $R_{max} = R(n)$

If, $\Delta_s=\Delta_r$, $R_{min} = R(n)$

$R(n+1) = R(n) \times 2^n$

Once $R_{max}-R_{min}$ reaches a threshold value the algorithms enters into the second phase and obtains a finer range of available bandwidth using binary search method similar to Pathload's approach. Once the difference between R_{max} and R_{min} reaches the desired accuracy the algorithm reports the available bandwidth as $A=(R_{max}+R_{min})/2$.

The authors state that the proposed algorithm requires smaller number of probing packets, it has less estimation time compared to Pathload and the estimate is more accurate.

2.2.10. eChirp

Suthaharan and Kumar [19] introduced a new available bandwidth estimation algorithm eChirp which has the same basic concept as the exponential packet train used in PathChirp, but uses a modified train structure. In the modified train structure (Figure 2-7) every odd packet repeats the probing structure and inter-packet gap as the previous packet. Moreover the probing rate is increased exponentially with only even power. The advantage of this type of train structure is that it requires half the number of probe packets within a chirp train as compared to PathChirp.

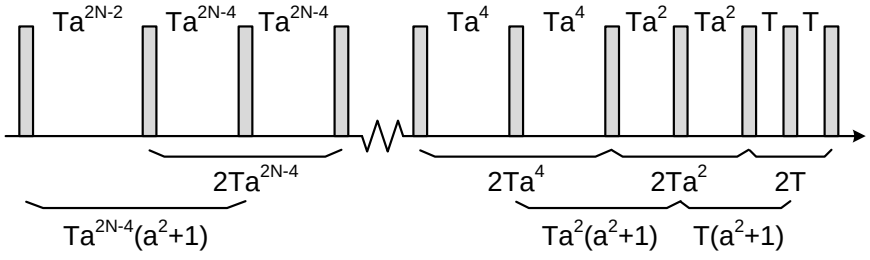

Figure 2-7. eChirp train structure

Each eChirp train can be seen as a combination of three different sub-trains with different probing structures. The first train is spaced by the probing rate increase of:

$$Ta^{2N-2}, Ta^{2N-4}, Ta^{2N-4}, \ \ldots, Ta^2, Ta^2, T, T$$

The second train is spaced by the probing rate increase of:

$$Ta^{2N-4}(a^2+1), Ta^{2N-6}(a^2+1), \ \ldots, Ta^2(a^2+1), T(a^2+1)$$

The third train is spaced by the probing rate increase of:

$$2Ta^{2N-4}, 2Ta^{2N-6}, \ \ldots, 2Ta^2, 2T$$

Hence the eChirp method can obtain more data than PathChirp to characterize the delay and excursion segmentation.

As each packet of the train belongs to three different sub-trains, each packet has three different instantaneous probing rate as well as per packet available bandwidth ($E_{k,j}^m$) associated with it. The overall per packet available bandwidth for the train is calculated as a linear combination of per-packet available bandwidths of three sub-trains as:

$$E_k^m = \left(3E_{k,1}^m + E_{k,2}^m + E_{k,3}^m\right)/5 \tag{2.9}$$

where, *m* is the train number, *k* is the packet number *j* is the sub-train number which can have a value of either 1, 2 or 3 indicating whether the packet belongs to first, second or third sub-train. Because of the equal spacing between two consecutive packets, the per-chirp available bandwidth for a chirp train is calculated as:

$$D^m = \frac{\sum\left(E_k^m + E_{k+1}^m\right)\Delta_k}{2\sum\Delta_k} \tag{2.10}$$

Finally the available bandwidth of the path is estimated as the average of all per-chirp available bandwidths.

2.2.11. Summary

Rate based algorithms can be used to estimate the available bandwidth of any completely unknown network. Unlike the gap-based methods, these methods do not require any prior information about the network path. As a result most of the researchers have followed this approach to estimate the AB. The major disadvantage of rate-based algorithms is that they inject a large number of probe packet trains at a higher rate than the AB and

overload the network path. This makes these methods much more intrusive compared to the gap-based algorithms. To overcome this problem, many researchers have focused on restructuring the packet trains to keep the probing rate and number of probe-packets as low as possible and gather more information about the network. Table 2-2 presents a summary of rate-based algorithms discussed in this section.

Table 2-2. Summary of Rate-based Algorithms

Algorithm	Year	Main Contribution/Feature	Sender Based
TOPP [7, 8]	2000	Assumes proportional share of probe-traffic and cross-traffic at the tight link. Uses trains of packet-pairs with linearly increasing input rate for the trains.	No
Curve Matching Algorithm [10]	2001	Uses curve matching technique to analyze the sending and receiving time curves of the probing packets.	Yes
Pathload [11, 12]	2002	Self-adaptive method that estimates the range of available bandwidth. Uses binary-search-like method to find out a range within which the actual AB may fall.	No
PathChirp [13]	2003	Usees exponentially spaced probing packets within a train. Requires only $log(G_2) - log(G_1)$ packets to probe a network over the range of rates $[G_1, G_2]$Mbps.	No
PathMon [14]	2004	Similar to the Curve Matching Algorithm [10] but, calculates the mean inter-arrival jitter and standard deviation prior to bandwidth estimate to improve accuracy	No
Pathtrait [15]	2005	Uses three different types of probing packets (load packet, forward packet & backward echo packet).	No
PoissonProb [16]	2005	Intervals between the probe packets are in Poisson distribution format	Yes
QuickProbe [17]	2006	Estimates available bandwidth with only two roundtrips.	No
Algorithm proposed by Xiao *et al.* [18]	2007	It is based on the SLoPS concept used in Pathload, but instead of only using binary search method, it first performs an exponential search to quickly find the rough range and then performs binary search to search for the actual range of available bandwidth.	No
eChirp [20]	2008	Instantaneous rates of even packets are increased exponentially with only even power and every odd packet repeats previous inter-packet gap. Each train consists of three sub-trains, which leads to more samples than PathChirp using less number of packets.	No

2.3. Model-based approach

This class of the available bandwidth measurement algorithms has been developed on the basis of the network traffic modeling research.

2.3.1. Delphi

The foundation of the Delphi algorithm [20] is based on the *multifractal wavelet model* (MWM). The core idea of the MWM is that the cross-traffic stream is a superposition of many data flows that share common link resources with the probe connections. The statistical analysis showed that such superposition has the characteristics of self-similarity, burstiness, long-range dependence (LRD) and even multifractal behavior (non-Gaussianity) [32]. This multifractal behavior makes it possible to present aggregated cross-traffic as a binary tree structure. In this structure, the β multiplier splits parent aggregate into two child aggregates at the next scale which increases or decreases β flow of traffic. The MWM also provides means to estimate the queuing behavior of a synthetic trace through the Multiscale Queuing Formula (MSQ) [32].

Following this model, the Delphi algorithm sends out chirps of $n+2$ probe packets within the time interval T_0, where T_i denotes the interval between the 1st and the $(n+2-i)$th probe packet. The initial interval between the packets is partitioned according to the exponential spacing and the interval is adjusted with the estimate of the previous result. Figure 2-8 depicts the exponential flight pattern used in Delphi and its relationship with the MWM tree. The tree coefficients $U_{j,k}$, $j \geq 0$, $k = 0,1,\ldots,2^j-1$, correspond to the total sum of cross-traffic bytes arriving at the model queue in the interval $[2^{-j}kT_0, 2^{-j}(k+1)T_0]$, where j denotes the scale of interest. Each parent coefficient $U_{j,k}$ is the sum of its two children $U_{j+1,2k}$ and $U_{j+1,2k+1}$ and $U_{j,k}$ is splits between its children by a random factor $B_{j,k}$ ($0 < B_{j,k} < 1$) such that $U_{j+1,2k} = B_{j,k} \times U_{j,k}$ and $U_{j+1,2k+1} = (1-B_{j,k}) \times U_{j,k}$. Therefore, MWM is essentially a parametric model for bursty non-Gaussian traffic with two parameters, a global mean-rate parameter or the scale of interest and the beta multiplier parameters. The initial estimate of beta multipliers is either based on previous measurements or is completely arbitrary. The gap change of two consecutive probing packets at the receiver is used to estimate the amount of traffic during that interval.

Delphi estimates the total cross-traffic arriving in the interval T_0 by recursive estimates of cross-traffics in the intervals T_1, T_2 and so on.

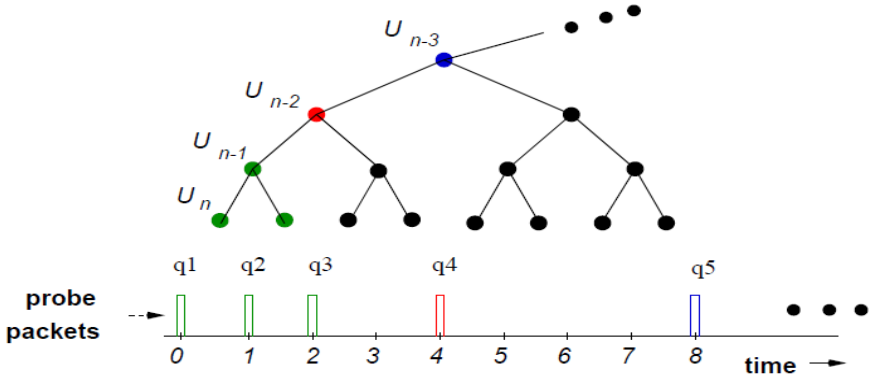

Figure 2-8. Exponential flight pattern and its relationship with the MWM tree

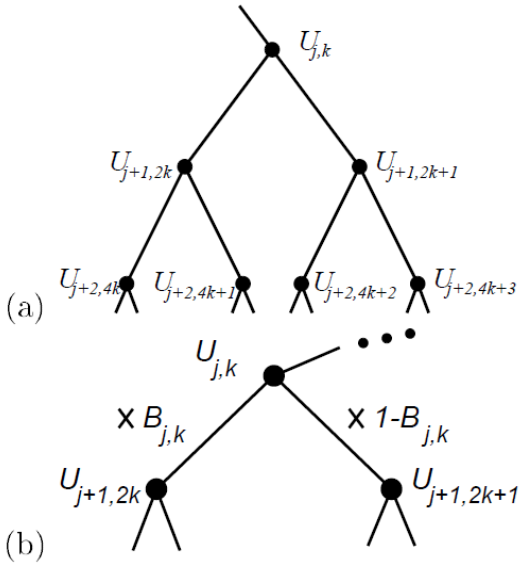

Figure 2-9. Multifractal wavelet model (MWM)
(a) Binary tree structure of aggregated traffic. (b) Beta multipliers split parent aggregate into two child aggregates at the next finer scale

Delphi assumes that the path can be well modeled by a single queue (single-hop model), However, this assumption is not applicable when the tight and bottleneck links are different. It also looks upon all the queuing delays in the path as delay at the tight link. This assumption, in some situations, leads to wrong estimate of the cross-traffic. Actually, the implementation of Delphi is similar to that of gap-based algorithms. But the two have different theoretical foundations [16].

2.3.2. IGI and PTR

Hu and Steenkiste [21] presented a single-hop gap model to establish the relationship between the competing traffic throughput and the change of packet pair gap for a single-hop network. The model can be represented as a 3-D graph as shown in Figure 2-10. It shows the output gap g_O as a function of the queue size Q and the competing traffic throughput B_C.

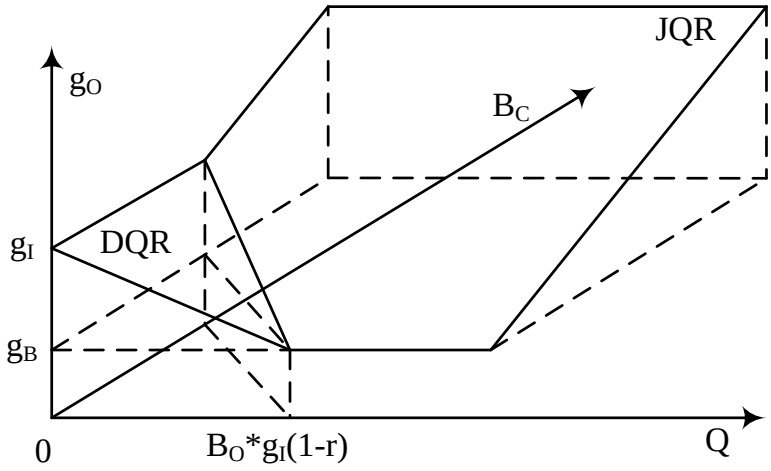

Figure 2-10. Single-hop Gap Model

Here, g_B is bottleneck separation gap, g_I is input gap between two packets P1 & P2 of a pair, g_O is output gap at the receiver, B_O is bottleneck capacity, B_C is cross-traffic rate, Q is queue size when the first packet P1 of the pair arrives at the router and $r=g_B/g_I$

The model assumes that the routers use FIFO queuing and all the probing packets have the same size. There are two regions in the model, the *joint queuing region* (JQR) where the router queue does not becomes empty during the period when both packets of the pair arrives at the router; and the *disjoint queuing region* (DQR) where the router queue becomes empty before arrival of the second packet. If the packet-pair operate in the DQR, the output gap will have no relationship with competing cross-traffic and can be represented as:

$$g_O = g_I - \frac{Q}{B_O} \tag{2.11}$$

On the other hand in JQR the output gap will have a linear relationship with competing traffic and can be represented by the following equation:

$$g_O = g_B + \frac{B_C \cdot g_I}{B_O} \tag{2.12}$$

Based on this model the authors proposed two available bandwidth estimation techniques, the gap-based method *Initial Gap Increasing* (IGI) and the rate-based method *Packet Transmission Rate* (PTR). Both IGI and PTR algorithms send a sequence of packet trains with increasing input gap. The measurement process terminates when average output gap is the same as the average input gap. The input gap is kept sufficiently small to ensure that all the probe packets within a train fall in the joint queuing region. Now in the probing train, consider that *M* probing gaps are increased, *K* are unchanged and *N* are decreased. The IGI algorithm then calculates the available bandwidth as,

$$A = B_O \times \left(1 - \frac{\sum_{i=1}^{M} \left(g_i^{+} - g_B \right)}{\sum_{i=1}^{M} g_i^{+} + \sum_{i=1}^{K} g_i^{=} + \sum_{i=1}^{N} g_i^{-}} \right) \tag{2.13}$$

Here, the gap values $G^{+} = \{g_i^{+} \mid i = 1, \ldots, M\}$, $G^{=} = \{ g_i^{=} \mid i = 1, \ldots, K \}$, and $G^{-} = \{g_i^{+} \mid i = 1, \ldots, N\}$. $B_O \sum_{i=1}^{M} \left(g_i^{+} - g_B \right)$ is the amount of competing traffic that arrives at the bottleneck router during the probing period. $\sum_{i=1}^{M} g_i^{+} + \sum_{i=1}^{K} g_i^{=} + \sum_{i=1}^{N} g_i^{-}$ is the total probing time.

The PTR algorithm on the other hand calculates available bandwidth using the equation,

$$\frac{\left(M + K + N \right) L}{\sum_{i=1}^{M} g_i^{+} + \sum_{i=1}^{K} g_i^{=} + \sum_{i=1}^{N} g_i^{-}} \tag{2.14}$$

Here, *L* is the size of probe packet.

2.3.3. Stochastic queuing model

Kang *et al.* [22] presented a generic stochastic queuing model of an internet router. The model assumes that the router introduces random delay noise ω to each arriving probe packet because of the cross-traffic. If the probing train consists of n equal sized packets of size q then the departure times d_n of the packets can be expressed as,

$$d_n = \begin{cases} a_1 + \omega_1 + \Delta & n = 1 \\ \max(a_n, d_{n-1}) + \omega_n + \Delta & n \geq 2 \end{cases} \tag{2.15}$$

Here, a_n is the arrival time of n-th packet and $\Delta = q/C$ is the process time of a packet by the router of capacity C.

The main idea of the model is to transmit the packets with inter-packet interval x in a way so that packet i arrives at the router before the departure time of packet $i-1$. This condition leads to $a_n \leq d_{n-1}$ and hence the inter-departure times y_n of the packets after the bottleneck router are given by:

$$y_n = d_n - d_{n-1} = \Delta + \omega_n, \quad n \geq 2 \tag{2.16}$$

In real networks such as the Internet, cross-traffic is bursty with a time-varying arrival rate. Considering the time varying nature of cross-traffic, the authors derive the mean output dispersion under arbitrary cross-traffic when the input spacing $x \leq q/C$ as:

$$E[y] = \Delta + \frac{x\bar{r}}{C} \tag{2.17}$$

where, $\bar{r}$ is the time-average of a cross-traffic arrival rate process $r(t)$ at the tight link:

$$\bar{r} = \lim_{t \to \infty} \frac{1}{t} \int_0^t r(u)du \tag{2.18}$$

Another important result in [22] shows that the variance of y decays to 0 as the packet-train length n (i.e., the number of packets in each train) increases. To estimate both capacity C and available bandwidth A from $E[y]$, the paper [22] defines W_n^a and W_n^b to be the average dispersion of two sets of measurements $\{y_i^a\}$ and $\{y_i^b\}$ (where the

index *i* represents the packet-train sequence number) with different initial spacings x_a and x_b:

$$W_n^a = \frac{1}{n}\sum_{i=1}^{n} y_i^a, \quad W_n^b = \frac{1}{n}\sum_{i=1}^{n} y_i^b \tag{2.19}$$

and calculates asymptotically accurate tight link capacity *C* and available bandwidth *A* of a single hop path as:

$$\lim_{n\to\infty} \frac{q(x_a - x_b)}{x_a W_n^b - x_b W_n^a} = C \tag{2.20}$$

$$\lim_{n\to\infty} q\left(\frac{x_a - x_b - W_n^a + W_n^b}{x_a W_n^b - x_b W_n^a}\right) = C - \overline{r} = A \tag{2.21}$$

The authors state that although their model is very accurate for single bottleneck link, but for a path with multiple congested links the estimation error increases significantly.

2.3.4. Envelope

Bhati [23] proposed a new algorithm Envelope, which is a recursive extension of the Kang's stochastic model introduced in [22], to estimate end-to-end available bandwidth of a multi-hop path.

Recursive extension is performed by treating inter-packet spacing x_k of probe traffic arriving at router R_k as the inter-departure delays y_{k-1} of the previous router R_{k-1} and the recursive relationship between the average output dispersions $E[y_k]$ and $E[y_{k-1}]$ can be expressed as:

$$E[y_k] = \begin{cases} E[y_{k-1}] & E[y_{k-1}] > \dfrac{q}{A_k} \\ \dfrac{q}{C_k} + \dfrac{\overline{r}_k E[y_{k-1}]}{C_k} & E[y_{k-1}] \le \dfrac{q}{A_k} \end{cases} \tag{2.22}$$

where, q is the packet size and C_k, A_k and r_k respectively are the capacity, available bandwidth and average cross-traffic rate of *k*-th hop.

To obtain the inter-packet spacing from router R_k, Envelope sends trains of n large probe-packets [P_1,P_2, …, P_n] surrounded by two small envelope packets E_1^k and E_2^k as show in Figure 2-11. The TTL value of the probe packets are adjusted in a way such that the probe traffic [P_1,P_2, … , P_n] is dropped at router R_{k+1} and the surviving envelope packets have a time spacing z_k that is (n + 1) times larger than y_k.

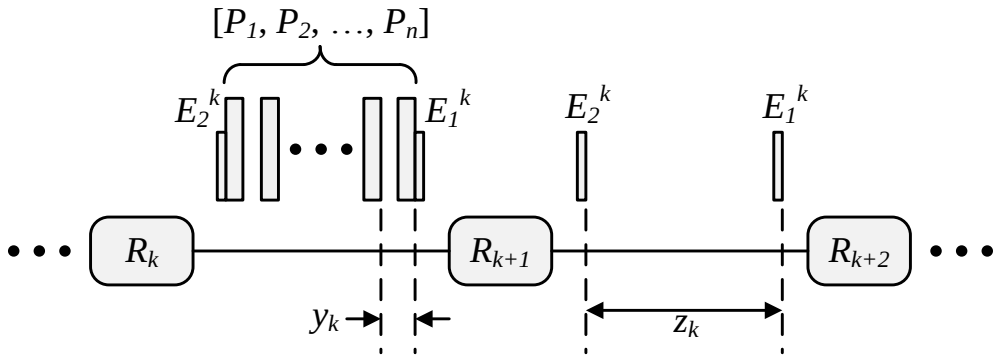

Figure 2-11. A probe-train [P_1, …, P_n] of n packets enveloped by two packets E_1^k and E_2^k at router R_k

At the receiver, the envelope-packets are sampled and then applied to the recursive queuing model to estimate the capacity as well as the available bandwidth for each link of the path using two sets of measurements with two different inter-packet spacings similar to the method used in the single-hop case in [22].

According to the author, the relative estimation error of Envelope is always less than 10%. But the error is high and it underestimates the available bandwidth when the bottleneck link precedes the tight link.

2.3.5. Summary

The model-based algorithms perform well when the network structure and cross-traffic follow exactly the same assumptions used to develop the algorithm. They have poor performance if the network or traffic pattern slightly deviates from the network model. Table 2-3 presents a summary of the model based algorithms discussed in this section.

Table 2-3. Summary of Model-based Algorithms

Algorithm	Year	Main Contribution/Feature	Sender Based
Delphi [20]	2000	Use of multifractal parametric model for cross-traffic estimate	No
IGI/PTR [21]	2003	Develop a "single-hop gap model" to relate between the competing traffic throughput and the change of the packet pair gap	No
Algorithm proposed by Kang *et al.* [22]	2004	Propose a stochastic queuing model for a single congested path	No
Envelope [23]	2004	Proposes a recursive extension of the stochastic queuing model for multiple congested links with arbitrary cross-traffic	No

2.4. Probabilistic Approach

Two groups of researchers have proposed probabilistic approaches SMART and A_ABE to estimate the available bandwidth of network. Both of these methods present a probabilistic definition of AB. Based on this definition; the two methods develop two new algorithms. This section presents a brief description of the two algorithms.

2.4.1. SMART

Min *et al.* [24] used a probabilistic approach to estimate the available bandwidth of an end-to-end network path. The authors defined available bandwidth in terms of probability and statistics and based on this definition they developed the new algorithm SMART (Statistics Measurement for Available-bandwidth by Random Train).

2.4.1.1. Probabilistic definition of Available Bandwidth

According to Min *et al.* [24], at any time instance, a network node can have only two states, it can either be idle or busy processing existing traffic. Therefore, the node can either process a new packet with its full capacity C when the node is free, or the packet can be queued up while the node is busy processing cross-traffic packets. Hence the available bandwidth of a link at any moment t can be defined as:

$$avail\_bw(t) = \begin{cases} C & \text{when node is free} \\ 0 & \text{when node is busy} \end{cases} \quad (2.23)$$

The available bandwidth of a link over the time period [t_1, t_2] can be estimated as the average of all the momentary available bandwidths during the interval. Hence,

$$avail\_bw(t_1, t_2) = \frac{1}{t_2 - t_1} \int_{t_1}^{t_2} avail\_bw(t) dt \quad (2.24)$$

The authors refer this as the non-intrusive available bandwidth of the link. For a multi-hop path, consisting of n links, the authors define the non-intrusive available bandwidth of the path at any moment t as:

$$na_n(t) = \begin{cases} C_{\min}(n) & \text{if } avail\_bw_k(t + \sum_{m=0}^{k-1} d_m) = C_k,\ 1 \le k \le n \\ 0 & \text{otherwise} \end{cases} \quad (2.25)$$

where, C_k is the capacity of k-th link, d_m is the transmission delay of a packet by m-th link and $d_0 = 0$. Finally the non-intrusive available bandwidth of a n-hop path for the period [t_1, t_2] is defined as:

$$NA_n(t_1, t_2) = \frac{1}{t_2 - t_1} \int_{t_1}^{t_2} na_n(t) dt \quad (2.26)$$

2.4.1.2. The SMART algorithm

Unlike all existing available bandwidth estimation algorithms, instead of using large probe packets, the SMART algorithm uses very small sized packets to probe the network. The packet size used by this method is only 40 bytes. Also it does not follow any specific pattern to transmit the packets. The algorithm sends a large number of small time-stamped packets at random intervals. The interval between two packets is kept large enough so that the front packet does not have any effect on the later packet. At the receiver the transmission delay of all the packets are recorded and the queuing delay for each packet is calculated by subtracting the minimum transmission delay (Mini-Transmission Delay) of all the packets from the transmission delay of the corresponding packet. If the queuing delay is zero, the algorithm assumes the path to be in available

state at that moment. Finally it estimates the available bandwidth of the path for the entire probing period as the average of all momentary available bandwidths.

One important assumption of this algorithm is that the packets with the Mini-Transmission Delay will not observe any queuing delay along the path. But in a heavily congested path, none of the probing packets may be transmitted without any queuing delay and the previous assumption may lead to error in estimation.

2.4.2. A_ABE

Zhou *et al.* [25] proposed a new probabilistic methodology to estimate the available bandwidth under "non busy assumption" which performs very accurately on a low utilization network path. The authors have also proposed a metric to weigh the busyness of a path based on the distribution of the output probe gaps. Finally using this metric, they introduced a new available bandwidth estimate method called Adaptive Available Bandwidth Estimate (A_ABE) which is suitable for both low utilization and high utilization path.

Under non busy scenario it is assumed that the inter-packet interval of the probe packets are set in a way that no more than one cross traffic packet arrive between two consecutive probe packets and the arrival time of a cross-traffic packet during a probe gap follows the Uniform distribution in the gap. Now the probability of probe gap increase because of a cross traffic (CT) packet is defined as:

$$P_{pgi} = P_{pgi|ctpa} \times P_{ctpa} \tag{2.27}$$

where,

$P_{pgi} = P\{\text{a probe gap increases}\}$

$P_{ctpa} = P\{\text{a cross traffic packet arrives during a probe gap}\}$

$P_{pgi|ctpa} = P\{\text{probe gap increases} \mid \text{a CT packet arrives during the gap}\}$

If the probe traffic consists of k kinds of packets each of size L_k and each type of packet arrives with probability P_k, then the probability that a probe gap increases because of CT packet of kind k is:

$$P_k \times \frac{L_k / C}{g_{in}}$$

where, C is the bottleneck link capacity and g_{in} is the input gap and $L_k/C.g_{in}$ is the probability that a packet of type k causes an increase in output probe gap (g_{out}). For all kinds of cross-traffic packets, (2.27) can be written as,

$$P_{pgi} = P_{ctpa} \times \sum P_k \times \frac{L_k}{C \times g_{in}} \tag{2.28}$$

If for a probing train consisting of n gaps, m probe gaps increase, then according to the authors, the probe gap increase frequency is equal to the probability P_{pgi}. Hence,

$$P_{ctpa} \times \sum P_k \times \frac{L_k}{C \times g_{in}} = \frac{m}{n} \tag{2.29}$$

Now, the mean of total cross-traffic that arrives during the probe gap g_{in} is $P_{cpta} \times \Sigma P_k \times L_k$. Hence the left side of (2.29) is the average cross-traffic rate during the interval g_{in}. Therefore the available bandwidth is calculated as:

$$A = C \times \left(1 - \frac{m}{n}\right) \tag{2.30}$$

The authors refer to this deduction process as NBE (non-busy estimate). To fulfill the non-busy assumption the size of inter-packet gap g_{in} for the probe packets is set as,

$$g_{in} = \frac{L_{probe}}{C} + \frac{1500Byte}{C} \tag{2.31}$$

where, L_{probe} is the size of probe packet. According to the authors, the NBE process is accurate in low utilization but it cannot estimate the available bandwidth when network utilization is high, whereas the IGI algorithm [21] gives a fairly good estimate for busy traffic.

To measure the busyness of the network path the authors have defined the metric Gap Symmetry (GS) as,

$$GS = \frac{\sum_{i=1;i<n;i=i+2} \left(g_{out}^{i} + g_{out}^{i+1} - 2 \times g_{in} \right)}{\sum_{i=1}^{n} \left(g_{in} \right)^2} \quad (2.32)$$

According to the authors, they have found from experiments that if *GS*>0.02, then the network path can be considered to be busy. Using this metric the authors introduced the A_ABE technique to estimate the available bandwidth both in non-busy and busy traffic conditions. The A_ABE tool first estimates the value of *GS*. If it is less than 0.02, it uses the NBE as estimate method. Otherwise it uses the IGI method, described in section 2.3.2, to estimate the available bandwidth of the network path.

2.4.3. Summary

The main objective of the probabilistic algorithms was to efficiently estimate the available bandwidth under low network utilization, where most gap-based algorithm fails. Table 2-4 presents the summary of probabilistic algorithms discussed in this section.

Table 2-4. Summary of probabilistic algorithms

Algorithm	Year	Main Contribution/Feature	Sender Based
SMART [24]	2003	Defines available bandwidth using probability and statistics. Transmits probe packets at random interval.	No
A_ABE [25]	2008	Presents a probabilistic definition of available bandwidth under low traffic condition and introduces a new metric to measure the business on network.	No

2.5. Hybrid Approach

Both the gap-based and rate-based available bandwidth measurement algorithms have some benefits and pitfalls. For example, gap-based algorithms need to know the bottleneck capacity of the path but they are less intrusive in nature than the rate-based

algorithms. On the other hand rate-based algorithms do not need any prior information about the path but send a large number of probe packets at a very high rate, which make them highly intrusive. To take advantages of both gap-based and rate-based algorithms some researchers have proposed hybrid methods which use a combination of ideas from both gap-based and rate-based algorithms. This section presents a brief description of the hybrid algorithms BET and MoSeab.

2.5.1. BET

Botta *et al.* [26] proposed a hybrid available bandwidth estimation tool called BET. The tool integrates the three different concepts, the Packet Train Dispersion (PTD) technique of path capacity estimate methods, SLoEC (Self Loading Exponential Chirp) used by the PathChirp [13] algorithm and the SLoPS (Self Loading of Periodic Streams) used by the Pathload [11] algorithm.

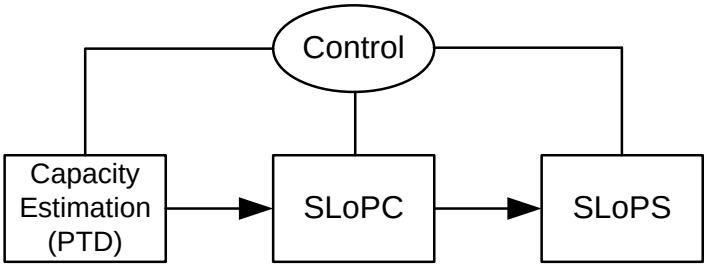

Figure 2-12. Modules of BET

In the first phase BET uses packet train dispersion technique to obtain the asymptotic dispersion rate (ADR) as well as an estimate of the capacity of the path. The ADR value found in this phase is passed as an input to the SLoPC module and used as the upper bound of the algorithm to make a fast estimate of available bandwidth. According to the authors this phase can estimate the available bandwidth up to 15% accuracy. The value obtained in the second phase is used as the initial value for the next phase, which is the SLoPS phase. In this phase the sender transmits several trains (fleet) consisting of 12 flows of packets with dynamically adjusted probing rate. For each train BET tool uses the Pairwise Comparison Test (PCT) and Pairwise Difference Test (PDT) to calculate the traffic trend. Based on the arrival rate of each fleet, a new probing rate is dynamically

calculated for the next fleet. The main advantage of this dynamic probing approach is that the probing traffic injected into the network is lower than the static choice and the estimation time is less.

2.5.2. MoSeab

The algorithm MoSeab [27], introduced by Zgang *et al.*, consists of two phases. The first phase is an iterative probing phase, where MoSeab obtains a rough estimate of available bandwidth. It starts to probe the network from an initial rate R_{min} = 200Kbps & train length 200, and doubles the probing rate at each subsequent run. At the receiver end the One Way Delay (OWD) trend is observed. If the OWD increases that means the probing rate R is higher than the available bandwidth A and at this point it stops probing and reports $\tilde{A} = R/\sqrt{2}$ as the rough estimate of AB.

In the second phase it sends four probing trains with rates 114%$\tilde{A}$, 133%$\tilde{A}$, 160%$\tilde{A}$ and 200%$\tilde{A}$ respectively and calculates the available bandwidth from the input probing rate and the OWD information of the received packets. If the probing rate is higher than the available bandwidth, then cross-traffic packets get queued up behind the probing packets and this causes the increase in the inter-packet intervals at the receiver side.

If C is the tight-link capacity, R_C is the cross-traffic rate, b is the size of probe packet, Δ_{in} is the inter-packet interval, $R_P = b/\Delta_{in}$ is the probing rate and A is the available bandwidth of the path, then the total amount of traffic arriving at the router during the period Δ_{in} is,

$$b + R_C\Delta_{in} = (R_P + R_C)\Delta_{in} > C\Delta_{in} \tag{2.33}$$

The amount of extra traffic, queued at the router is given by:

$$\delta Q = (R_P + R_C)\Delta_{in} - C\Delta_{in} = \frac{(R_P - A)b}{R_P} \tag{2.34}$$

Therefore, the increase in OWD between two successive packets is given by

$$\delta D = \frac{\delta Q}{C} = \frac{b}{C} - \frac{bA}{C} \cdot \frac{1}{R_P} = \alpha - \beta \cdot \frac{1}{R_P} \tag{2.35}$$

Now with four different measurements, first α and β are estimated using linear regression, and then the available bandwidth and the capacity of the tight link is calculated as:

$$A = \frac{\beta}{\alpha}, \quad C = \frac{b}{\alpha} \tag{2.36}$$

The authors have proved that the mathematical model of MoSeab is also valid for multiple tight link scenarios. The problem with MoSeab is that, it requires a considerable amount of time for probing trains in the first phase to iteratively estimate the rough available bandwidth.

2.5.3. Summary

The hybrid algorithms combine the concept and train structures of different existing algorithms and try to put together their advantages to improve the estimation process. Table 2-5 presents the summary of hybrid algorithms discussed in this section.

Table 2-5. Summary of hybrid algorithms

Algorithm	Year	Main Contribution/Feature	Sender Based
BET [26]	2005	Combines the concepts of Packet Train Dispersion technique of path capacity estimation methods, exponential chirp train used in PathChirp and the SLoPS technique used in Pathload algorithm.	No
MoSeab [27]	2006	Consists of two phases. First it uses a rate based approach and iteratively probes the network to obtain rough AB. In the second phase it uses a gap-based approach based on a new mathematical model to obtain actual available bandwidth.	No

2.6. Kalman Filtering based Algorithm

Two groups of researchers have proposed Kalman filtering (KF) based approach to address the problem of available bandwidth estimate. This section first briefly describes the Kalman filter and then discusses the two algorithms *BART* and *Abest* which use KF to estimate the available bandwidth of network path.

The Kalman filter is a set of sequential mathematical operations to iteratively estimate or predict the state of a system and then improve the estimate using a set of measurements. The detailed description of Kalman filter can be found in [33] and the references therein.

In general Kalman filter describes the system state $x \in \Re^n$ by the linear stochastic difference equation:

$$x_k = Ax_{k-1} + Bu_{k-1} + w_{k-1} \tag{2.37}$$

with a measurement $z \in \Re^m$ that is

$$z_k = Hx_k + v_k \tag{2.38}$$

Here u is the control input, w is the process noise with Gaussian probability distribution $N(0,Q)$ and v is the process measurement noise with Gaussian probability distribution $M(0,R)$ where Q and R are process and measurement noise covariance matrices respectively. The subscript k refers to discrete time and A and B relate the state and control input of previous step $(k-1)$ with that at the new step k, while H relates the state with measurement.

Each of the iterations of the Kalman filter works in two steps. In the first step ("time update") or the prediction step it obtains *a priori* estimate of the state ($\hat{x}_k^-$) and estimation error covariance (P_k^-) matrices. The predictor equations can be summarized as:

$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_{k-1} \tag{2.39}$$

$$P_k^- = AP_{k-1}A^T + Q \tag{2.40}$$

The second step of Kalman filter is the "measurement update" or correction step. In this step it first computes the Kalman gain, K_k and then uses the current measurements along with K_k to correct the *a priori* estimates and obtains improved estimates which are used in the next iteration. The correction equations of Kalman filter can be summarized as:

$$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1} \tag{2.41}$$

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-) \tag{2.42}$$

$$P_k = (I - K_k H)P_k^- \tag{2.43}$$

The key parameters of Kalman filter are Q and R. The measurement noise covariance R can be determined by taking some off-line sample measurements. Choosing the process noise covariance Q is more difficult as the system may be completely unknown. Higher value of Q means low stability but fast convergence of the filter, since the predictions will be considered less accurate while the measurements will be considered very accurate giving relatively greater weight to current measurement. Low values of Q, on the other hand, result in higher stability in presence of high measurement errors but slower step response.

2.6.1. BART

The BART (Bandwidth Available in Real-Time) method was proposed in [28] by Ekelin *et al.* The method uses the same network model used in TOPP [7] method, but uses a variation of Kalman filter, which the authors refer to as the BART filter, to estimate the value of available bandwidth instead of using linear regression.

The Kalman filter is an efficient recursive filter that estimates the state of a linear dynamic system from a series of noisy measurements. Similar to TOPP, the BART method uses trains of packet-pairs to probe the network. For each packet pair the inter-packet strain ε (instantaneous output rate decrease ratio of a packet-pair) is calculated as:

$$\varepsilon = \frac{u}{r} - 1 \tag{2.44}$$

where, u and r respectively are the input and received rates of probe packets. The BART filter assumes the network as a system having state vector x with input u and measured output ε which is affected by some measurement noise v. The state vector x is represented as:

$$x = \begin{bmatrix} \alpha \\ \beta \end{bmatrix} \tag{2.45}$$

where, α and β are the parameters of the sloping straight line $\varepsilon = \alpha u + \beta$ in the measurement model (Figure 2-13) and the measured output ε can is described by:

$$\varepsilon = f(x,u,v) = \begin{cases} 0 & (u \le A) \\ \alpha u + \beta & (u > A) \end{cases} \tag{2.46}$$

where, A is the available bandwidth of the path.

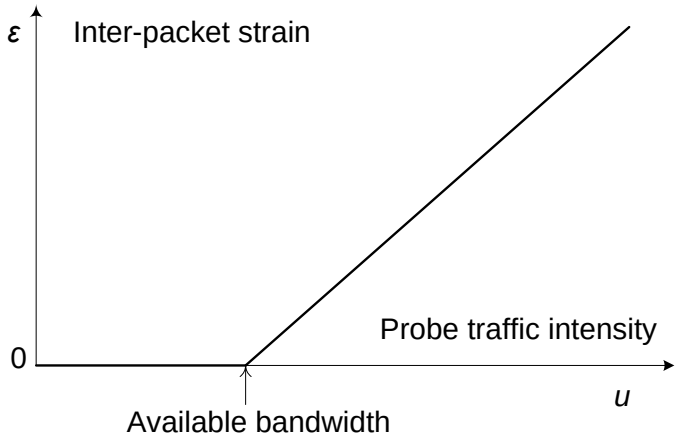

Figure 2-13. Asymptotic relation between available bandwidth, probe traffic rate and expected inter-packet strain

During the estimation process, the receiver first initializes the state vector estimate $\hat{x}$, available bandwidth estimate $\hat{A}$ and the error covariance matrix for $\hat{x}$. The sender sends a sequence of probe packet-pairs with input rate u. If $u > \hat{A}$ the receiver

computes average strain ε and its variance. It then passes these values to the Kalman filter which then updates the estimates of state vector $\hat{x}$ and error covariance matrix. The receiver then uses the updated $\hat{x}$ to compute the input rate u for the next sequence of probe packets.

According to the authors, given the points corresponding to current input rate u, Kalman filter attempts to find an approximate straight line L_1 (Figure 2-14) for the curve $\varepsilon(u)$ and estimates available bandwidth $\hat{A}_k$ as the intersection point of this curve with u-axis. Now assuming that the current estimate is an underestimate of A, in the next rounds Kalman filter is applied only with the values of u such that $u > \hat{A}_k$ and the filter attempts to find a new line L_2. This line will intersect the u-axis at a point $\hat{A}_{k+1}$, where $\hat{A}_k < \hat{A}_{k+1} < A$, indicating a better approximation of available bandwidth.

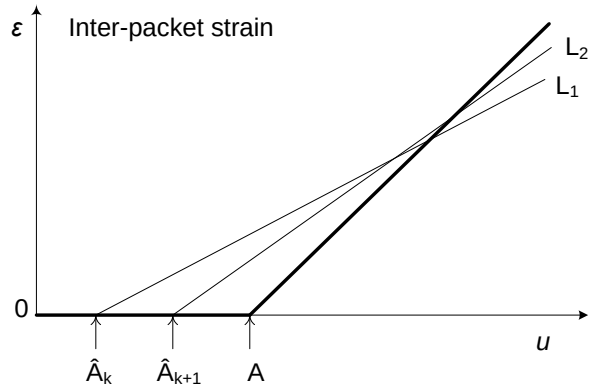

Figure 2-14. Convergence of the BART method

2.6.2. Abest

Cabellos-Aparicio *et al.* [29] propose another method Abest which also uses Kalman filtering method to estimate the available bandwidth of the network. The estimate methods of BART [28] and Abest are very similar. But the key difference is that BART is based on the fact that the inter-packet strain has a linear relation with probe traffic rate when the probing rate is higher than the path's available bandwidth; on the other hand Abest is based on the mathematical model proposed by Harfoush *et al.* in [34] which shows that there is a linear relation between the link utilization and probe traffic rate when the probing rate is less than the available bandwidth. Harfoush *et al.* showed that

for a multi-hop path the utilization $u_i(r)$ of link i under probe traffic rate r can be represented as:

$$u_i(r) \cong \min(1, ar + b) \tag{2.47}$$

where, a and b are constants of the straight line (Figure 2-15). The probing rate r_{ab} for which the utilization becomes 1 is reported as the available bandwidth of the path.

The Abest algorithm sends 200 packets of size 1500 bytes with exponentially distributed inter-packet intervals. It obtains the values of a and b using Kalman filtering method similar to BART approach. Finally it estimates the available bandwidth as:

$$AB = \frac{1 - \bar{b}}{\bar{a}} \tag{2.48}$$

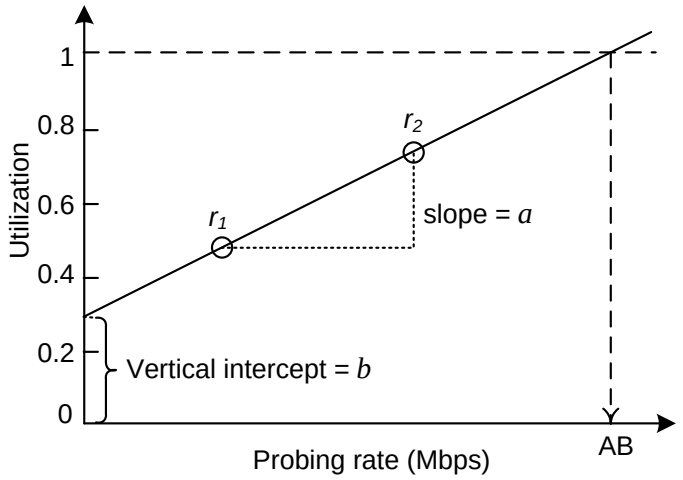

Figure 2-15. Linear model of Abest

The main advantage of Abest algorithm is that, unlike most of the available bandwidth estimation methods, it sends probe packets at a lower rate than AB and does not create congestion in the network. According to the authors the algorithm is very accurate when the network utilization is low, but it is less accurate for heavily utilized path.

2.6.3. Summary

Table 2-6 summarizes the Kalman filtering based available bandwidth estimation algorithms discussed in this section.

Table 2-6. Summary of Kalman filtering based algorithms

Algorithm	Year	Main Contribution/Feature	Sender Based
BART [28]	2006	Based on the fact that inter-packet strain has a linear relation with probe traffic rate when the probing rate is higher than path's available bandwidth. Calculates AB by finding the intersection point of this relationship curve with the traffic-rate axis.	No
Abest [29]	2008	Based on the fact that there is a linear relation between the link utilization and probe traffic rate when the probing rate is less than the available bandwidth. Probe packets are injected at a rate lower than the available bandwidth.	No

CHAPTER III

THE PROPOSED ALGORITHM: PATHAB

PathAB is a hybrid algorithm to measure the available bandwidth of a network path. It uses both rate-based and gap-based approaches in the estimation process. The rate-based approach allows it to operate without any information about the bottleneck capacity, whereas the gap-based approach enables PathAB to probe the network with probing trains of lower input rates. The algorithm has been developed using the concepts of three existing algorithms PathChirp, PoissonProb and MoSeab. Like MoSeab and PoissonProb, PathAB also consists of two phases. In the first phase, or initial probing phase it obtains a rough estimate of the available bandwidth of the path and in the second phase or direct probing phase it refines the estimate received from the previous phase. PathAB can be seen as an improvement as well as an extension of MoSeab. It uses the same mathematical model as MoSeab to calculate the final available bandwidth. However while MoSeab probes the network iteratively with several long probing trains, PathAB uses only one exponential packet train, as in PathChirp, to probe networks with a wide range of bandwidth. This reduces the duration and number of packets, sent in the initial phase. Since recent studies [31] [35] [36] have shown that the current Internet traffic follows Poisson pattern, PathAB uses Poisson distributed probing trains in its second phase. Finally, where MoSeab operates only in client-server mode, the proposed algorithm PathAB has the capability to operate both in client-server mode as well as in stand-alone mode without any help from the server. This chapter presents a detailed description of PathAB and its operating principles both in client-server mode and in stand-alone mode.

3.1. Client-Server Mode

In the client-server mode the bandwidth measurement tool has to be deployed both on the sender and the receiver side. The receiver side acts as the server. The client or the sending hosts transmits a sequence of probe packet-packet trains. Each packet of the train is time-stamped before transmission. The receiving host receives the probe packets and obtains

reception time of each packet. The receiver uses the transmission time and reception time of all the packets to calculate the available bandwidth of the path. In the client-server mode no calculation is done at the sending host.

3.1.1. Initial Probing Phase

Although PathAB uses the mathematical model of MoSeab to calculate the final available bandwidth, but unlike MoSeab, in the initial probing phase it does not use iterative probing trains with increasing probing rate. On the contrary, to reduce the number of probing trains, as well as the probing time, it uses exponentially spaced probe packets within a packet train. The concept of an exponential flight pattern of probe packets was first introduced by Ribeiro *et al.* [13]. The advantage of this approach is that, by exponentially increasing the packet spacing, the network over the range of rates [G1, G2]Mbps can be probed using just log(G2) – log(G1) packets.

The exponential probing train consists of N probe packets of size Q^E resulting in a total of $N-1$ inter-packet intervals. The inter-packet intervals of two consecutive packets are decreased by a factor γ, which is referred to as the spread factor of the algorithm. The probe packets of the exponential probing train are spaced by:

$$T\gamma^{N-2}, T\gamma^{N-3}, ..., T\gamma^{3}, T\gamma^{2}, T\gamma, T$$

where, $T\gamma^{N-2}$ is the 1st and T is the (N-1)-th or the last input gap. This leads to probe packets' instantaneous rate increase from min_rate = $Q^E / T\gamma^{N-2}$ to max_rate = Q^E / T. PathAB uses probe packets of 1200 bytes for the exponential probing train. The instantaneous probing rate is increased from 100 Kbps to 100 Mbps by default.

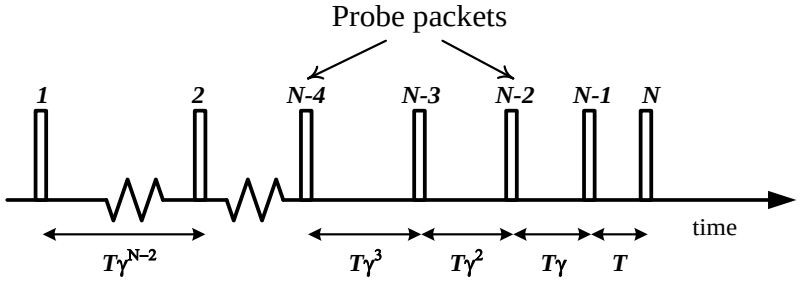

Figure 3-1. Exponentially spaced probing train

The receiver receives all the packets, calculates the output gaps of consecutive received packets and then determines the One-Way-Delay (OWD) increase ratio (R^{OWD}) as:

$$R^{OWD} = \frac{OutputGap - InputGap}{InputGap} \times 100 \tag{3.1}$$

If δ_i is the i-th input gap, then the i-th instantaneous probe rate is $r_i = Q^E / \delta_i$. Now, if r_i greater than the available bandwidth A, then the extra traffic ($r_i - A$) will cause an increase in the OWD and the i-th OWD increase ratio can be expressed as:

$$R_i^{OWD} = \frac{r_i - A}{C} \Big/ \delta_i = \frac{1}{C}\left(Q^E \frac{1}{\delta_i^2} - A\frac{1}{\delta_i} \right) \tag{3.2}$$

This leads to the conclusion that the OWD increase ratio is inversely proportional to the input gap. As the instantaneous input rates are increased exponentially by decreasing the input gaps, from equation (3.2), it is obvious that the OWD increase ratio will increase exponentially within a train in an ideal scenario. But in reality because of the traffic fluctuation and packet drops by the intermediate routers, some spikes are observed when R^{OWD} is plotted against the packet number.

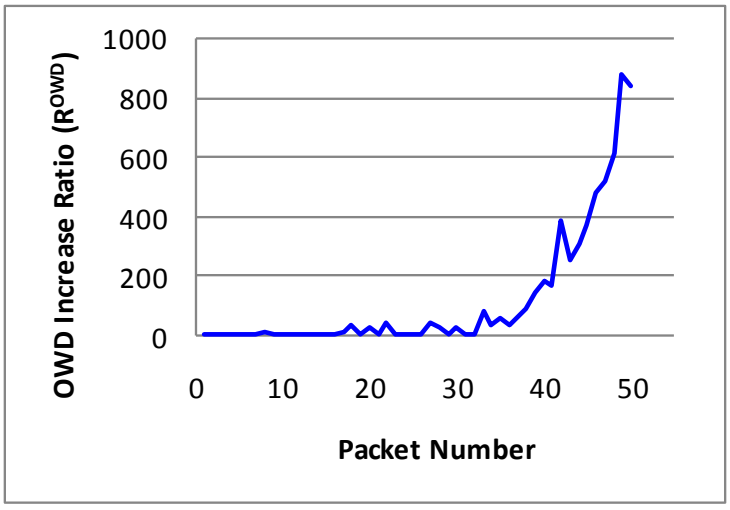

Figure 3-2. OWD Increase Ratio vs. Packet numner

To remove these spikes first the curve is smoothened so that for each packet p_i, $R_i^{OWD} \geq R_{i-1}^{OWD}$. The pseudo code for smoothing the curve and removing the spikes is given below:

```
Algorithm RemoveSpikes:
for i = 2 to number of packets
    if(R^OWD_i < R^OWD_i-1)
        /* decrease all R^OWD prior to i'th packet by certain
        percentage (decreaseRatio) and make R^OWD_i = R^OWD_i-1,
        such that the sum of R^OWD remains the same */
        sum = 0;
        for j=1 to i-1
            sum = sum + R^OWD_j;
        decreaseRatio = (R^OWD_i-1 - R^OWD_i)/(sum + R^OWD_i-1);
        for j=1 to i-1
            R^OWD_j = (1 - decreaseRatio) * R^OWD_j;
        R^OWD_i = R^OWD_i-1;
```

Figure 3-3. Pseudo code to smoothen OWD increase ratio curve

After the spikes are removed, an exponential trend line of the form $y = Ae^{Bx}$ is constructed for the curve. Here, x is the packet number and y is the OWD increase ratio for that packet. The values of A and B are obtained using the following exponential best curve fitting equation:

$$a = \frac{\sum_{i=1}^{n} \ln y_i \sum_{i=1}^{n} \ln x_i^2 - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} x_i \ln y_i}{n\sum_{i=1}^{n} \ln x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2}$$

$$b = \frac{n\sum_{i=1}^{n} x_i \ln y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} \ln y_i}{n\sum_{i=1}^{n} \ln x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2} \qquad (3.3)$$

where, $B \equiv b$ and $A \equiv \exp(a)$

After smoothening and then fitting the curve exponentially we obtain a curve similar to the one shown in Figure 3-4:

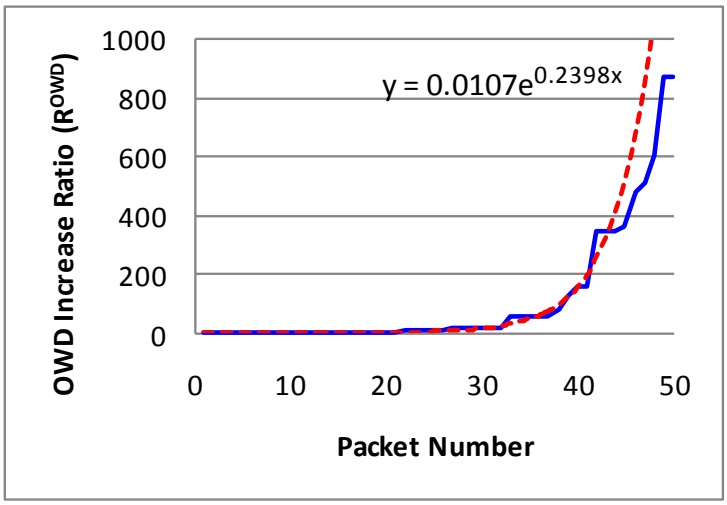

Figure 3-4. OWD Increase Ratio vs. Packet numner (after removing spikes and fitting exponential curve)

Once the values of A and B of the equation $y = Ae^{Bx}$ are calculated, the rough available bandwidth ($\tilde{A}$) of the path is estimated using the following equation:

$$\tilde{A} = min\_rate \times spread\_factor^{(ln\ 10\ -\ ln\ A)/B} \tag{3.4}$$

The point where the OWD increase ratio is just greater than 10% is used to estimate the rough available bandwidth $\tilde{A}$ so that the available bandwidth in the initial probing phase is never underestimated.

3.1.2. Direct Probing Phase

The second step of PathAB is to find out the actual available bandwidth of the path. In this phase the sender transmits several probe packet-trains with different input rates in each train. PathAB uses the value of rough available bandwidth $\tilde{A}$, obtained from previous initial probing phase to determine the input rates of the probing trains in the second phase. The direct probing phase is similar to the second phase of MoSeab. It uses the same assumptions and mathematical model as MoSeab. The only difference is that, instead of equally spacing the probe packets within each train, the inter-packet gaps are set in such a way that they are in Poisson distribution format (as shown in Figure 3-5) with mean probing rate R_P. To reduce the overall rate of the entire probing phase, the

inter-train intervals, τ_i, are adjusted in such a way that the average probing rate during the entire direct probing phase remains within 10% of the rough available bandwidth $\tilde{A}$. If R_{Pi} is the mean rate of i-th probing train, then the inter-train interval τ_i between i-th and (i+1)-th train is calculated as:

$$\begin{aligned}\tau_i &= \frac{N \times Q^P}{0.1 \times \tilde{A}} - \sum_{i=1}^{N-1} t_i \\ &\approx \frac{N \times Q^P}{0.1 \times \tilde{A}} - (N-1)\frac{Q^P}{R_{Pi}}\end{aligned} \tag{3.5}$$

where, Q^P is the size of probe packet, N is the number of packets within a train, t_i is the inter-packet interval between i-th and (i+1)-th packet. The inter-packet interval t_i's are distributed in Poisson distribution format with mean Q^P/R_{Pi} resulting in mean probing rate R_{Pi}.

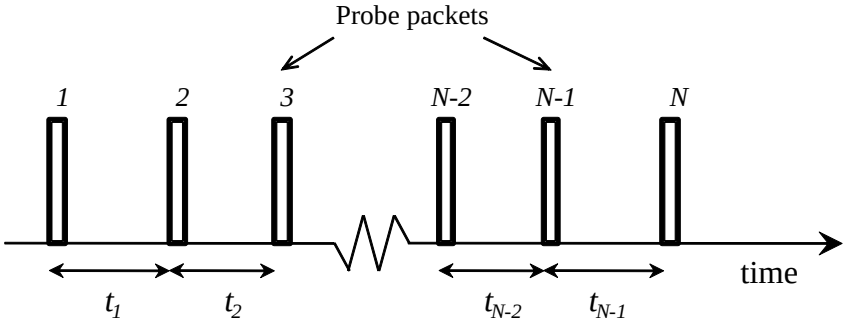

Figure 3-5. Packet distribution within a train. Inter-packet gaps t_1, t_2,…, t_{N-1} are in Poisson distribution

The sender sends 15 Poisson distributed probing trains of length N=30 with the probing rate increasing uniformly from 85%$\tilde{A}$ to 200%$\tilde{A}$. All the probe packets are of size Q^P=1200 bytes and are time-stamped by the sender.

At the receiver for each train the average input gap, output gap and input rate are collected. These ten sets of values are used to solve Eq. (2.35) with the help of linear regression and finally the available bandwidth is calculated using Eq. (2.36), as described in section 2.5.2 for the MoSeab algorithm.

3.1.3. Complete client-server algorithm

The PathAB algorithm has been implemented using C++ on Linux environment. In client-server mode it opens two connections to measure the available bandwidth between the client and server. It opens one TCP socket to transfer control information and available bandwidth information between the sender and receiver. Another connection is the UDP socket to transmit the probe packets from client to server. To achieve nano-second level time resolution for times-stamping the packets we have used the ***clock_gettime(clockid_t*** *clk_id,* ***struct timespec*** **tp)* function of Unix "time.h" library. Also to set the inter-packet intervals at nano-second resolution we have used the ***nanosleep(const struct timespec*** **req,* ***struct timespec*** **rem)* function. To perform any measurement the PathAB server has to be started first. The server opens a TCP port and waits for measurement request from the client. When the measurement starts, the PathAB client first establishes a TCP connection with the server and sends a measurement request. Upon receiving measurement request the server opens a UDP port and informs the client though the TCP connection to begin measurement process. After receiving response from the server the client creates a UDP connection with the server. In the first phase PathAB client sends exponential probing train. By default the size of the probe packet is kept 1200 bytes, the instantaneous probe rate is increased from *min_rate* = 100 Kbps to *max_rate* = 100 Mbps with a *spread_factor* γ=1.2. Each packet is time-stamped by the client before transmission. The server also time-stamps each arrived packet. After receiving all the packets of the exponential train it calculates the rough available bandwidth $\tilde{A}$ of the path and informs the client through the TCP connection. The next step of the algorithm is to transmit the probing trains for measurement of the available bandwidth. When the client receives $\tilde{A}$ from the server, it calculates the rates of 10 probing trains to be transmitted and also the inter-train intervals. The client transmits 15 probing trains of length L = 30 with Poisson distributed probe packets, with mean probing rate of the trains increasing uniformly from 85%$\tilde{A}$ to 200%$\tilde{A}$. For each received train the server calculates and stores the average input gap, average input rate and the average output gap. Finally it uses all this information to estimate the available bandwidth A of the path and informs the client through the TCP connection.

We have performed extensive experiments on the network test-bed as well as NS2 simulations to observe the performance of PathAB. It has been found that on network test-bed, when the available bandwidth of the path is less than 2 Mbps the first phase of the algorithm cannot report any value for rough estimate. The reason is that in the first phase, PathAB transmits packets in exponentially increasing rate. So a considerable number of packets get dropped by the router at tight link. This leads to failure of PathAB to estimate the available bandwidth of the path. We have found that the first phase fails if more than 20% packets are lost in the exponential train. To prevent this situation if the first phase of PathAB fails, it assumes that the available bandwidth is less than 2 Mbps and reports a random value around 2 Mbps as the rough available bandwidth. The second phase then can proceed with the estimation process, using the value obtained from the previous phase. In simulation experiments, we have used infinite queue length for the routers to ensure zero packet loss, hence the packet loss scenario was not considered in NS2 implementation of PathAB.

3.2. Stand-alone Mode

The stand-alone mode of PathAB is developed using the help of ICMP echo protocol. The primary requirement of this mode is that the UDP echo port (port 7) should be opened at the destination host. The sender maintains all the timestamp information of transmitted probe packets and performs calculations after receiving back the echo packets, keeping the load on the target host to a minimum. The stand-alone mode of PathAB may be used in situations when the target host is out of sender's administrative domain and it is not possible to install the server software on the target host. Unlike the stand-alone mode of PoissonProb algorithm, described in section 2.2.7, instead of echoing all the large probe packets, PathAB sends very small UDP echo packets back-to-back behind the large probe packets. Because of the small size, the echo packets will have negligible effect on the cross-traffic in the returning path. PathAB transmits ICMP echo request packets with the minimum size of 28 bytes, which is the total size of IP header and ICMP header without any message body. A brief description of ICMP echo protocol is given in Appendix-B. During the measurement process PathAB algorithm bounces the echo packets at the UDP port 7 of the target host. The large probe packets are

dropped at target host. The returning path of the echo response packets may be different from the forward path, but we assume that all the echo packets follow the same returning path without being affected by cross-traffic on the returning path. That means the time required to travel the path from target host back to the sending host is the same for all the echo response packets and the inter-arrival time between two consecutive packets are independent of the transmission time from target host back to the sending host. The stand-alone mode of PathAB also consists of two phases, the initial probing phase and the direct probing phase. The following part of this section describes the probing train structure used for the stand-alone mode of PathAB.

3.2.1. Initial Probing Phase

In the initial probing phase, PathAB sends large probe packets in exponentially increasing probing rate. So if the probing packets are echoed back as it is, there is a high probability that at least some of the packets will be affected by the cross-traffic or the bottleneck link of the returning path. To alleviate this problem the proposed method does not echo back the large probe packets. Instead during this phase, each probe packet is followed back-to-back by a very small echo packet. The size of the probe packets is 1200 bytes, whereas the size of echo packets is only 28 bytes. The probe packets are dropped or ignored at the destination host. The structure of the exponential probing used in the initial probing phase of PathAB is shown in Figure 3-6.

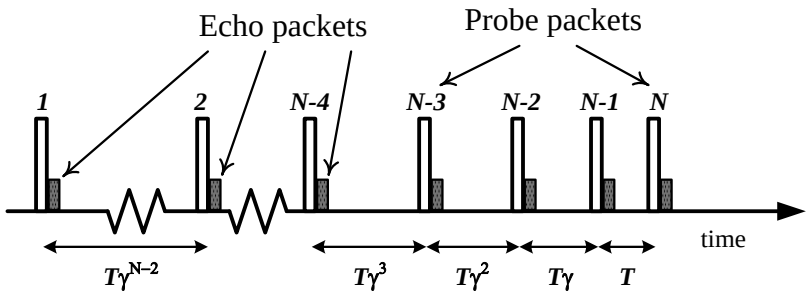

Figure 3-6. Exponentially spaced probing packets with back-to-back echo packets

Before transmitting the echo packets the sender timestamps the packets and keeps track of all the transmission times. After receiving the echo response packets back, the

sender calculates the rough available bandwidth $\tilde{A}$ of the path, in the same way as in the client-server mode as described in section 3.1.1.

3.2.2. Direct Probing Phase

In the direct probing phase all the probe packets are not followed by echo packets; instead the first and last packet of the Poisson distributed probing trains are followed back-to-back by 28-byte echo packets. The packets distribution for the direct probing phase of stand-alone mode is shown in Figure 3-7

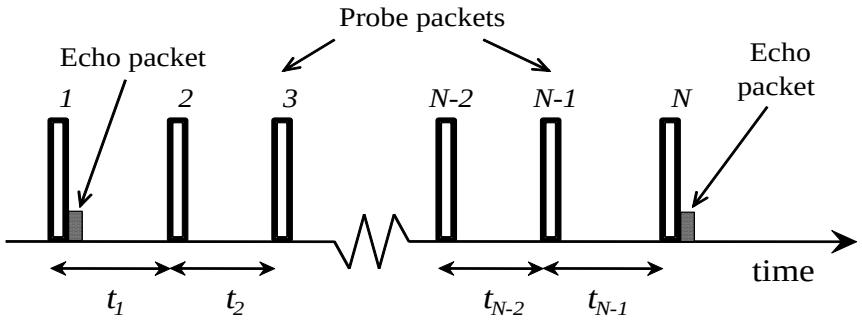

Figure 3-7. Packet distribution within a train in Stand-alone mode. Inter-packet gaps t_1, t_2, … , t_{N-1} are in Poisson distribution

The sender, after receiving the echo packets, calculates the average output gap for the train. If the gap between two echo packets is g, then the average output gap for the train will be $g/(N-1)$, where N is the number of packets in the train. The sender also keeps track of the input rates and average input gaps of the probing trains.

After receiving all the echo packets, the sender calculates the available bandwidth by solving Eq. (2.35) and (2.36), as in the client-server mode.

3.2.3. Complete stand-alone mode algorithm

In the stand-alone mode PathAB does not require any help from the target host and completely relies on ICMP echo packets. The requirement for this mode is that the UDP echo port 7 should be open at the target host. In stand-alone mode the sender program creates two threads. The first is the sender thread. It is used to transmit the probe and echo packets. The second is the receiver thread. It is used to receive the echo response

packets. We have used POSIX thread library to create the threads. The sender thread first sends one ICMP echo packet to the target host to check whether the echo port is open at the destination. If it receives the echo response back, then it proceeds with the measurement phase. First the sender thread transmits exponential probe train. Each packet of the train is followed back-to-back with small 28 byte ICMP echo packet. The large probe packets are dropped at the destination host. The receiver thread, after receiving the echo response packets, calculates the rough available bandwidth $\tilde{A}$ of the path. The sender thread then uses the value of rough available bandwidth to calculate the rates of Poisson distributed probing trains in the next phase and transmits 10 Poisson distributed probe trains. The first and the last probe packets of each train is followed back-to-back by 28 byte ICMP echo packets. The receiver thread receives the echo response packets and calculates the available bandwidth using the transmission time and reception time of all the ICMP packets. The sequence number field of UDP echo packet header is used to send the train number and packet sequence number with each echo packet.

3.2.4. Position of the Echo Packet

A packet of size q takes a time of q/C to arrive at the router after traversing a link of capacity C. Therefore a probe packet of size 1200 bytes takes $t_1 = (1200*8/C)$ seconds to arrive at the router after traversing a link of capacity C. The echo packet of size 28 bytes takes $t_2 = (28*8/C)$ second, which is negligible compared to t_1. We assume that the router takes negligible time to inject any packet to the next link regardless of the size of the packet.

If the echo packet is placed before the large probe packet, the echo packet will first arrive at the router immediately and leave the router, whereas the probe packet will take t_1 second to arrive at the router through the bottleneck link before it can leave the router. So after both the packets leave the router, a gap of t_1 second will build up between the packets. This gap might keep on increasing at the next router because of the cross-traffic packets that arrive during the interval t_1 in the next link.

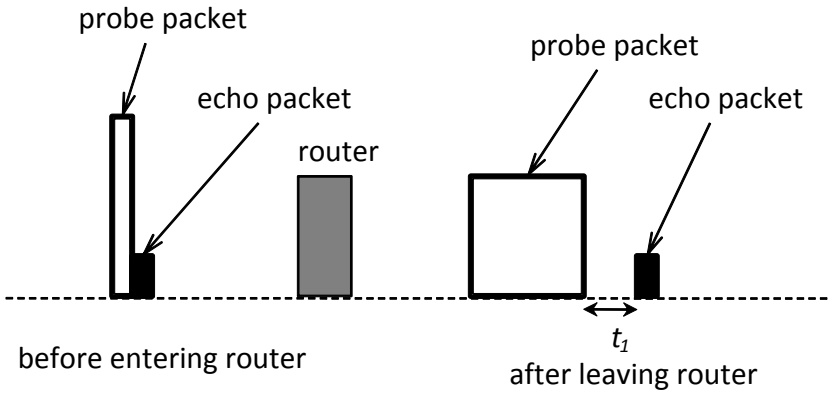

Figure 3-8. Gap builds up between two packets if the echo packet is followed by probe packet

On the other hand, if the echo packet is placed behind the probe packet, the probe packet will first arrive at the router and as soon as the probe packet arrives, the echo packet will also arrive at the router immediately. Therefore when the two packets leave the router, there will be no gap between the packets.

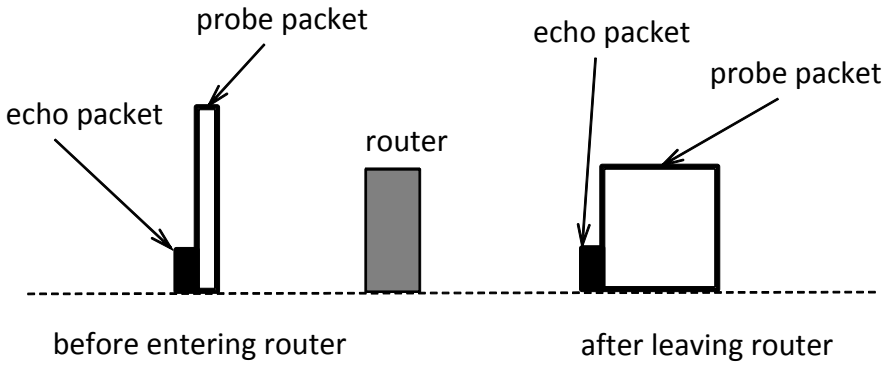

Figure 3-9. No gap builds up between two packets if the probe packet is followed by echo packet

In our proposed method PathAB we have placed the echo packet behind the probe packet in the stand-alone mode so that no gap can build up between the probe packet and the echo packet.

CHAPTER IV

EXPERIMENT AND ANALYSIS

To study the performance of PathAB we have performed extensive experiments using the network simulator NS-2 and as well as the network testbed in our laboratory and compared the performance of PathAB with some existing available bandwidth estimation methods. This chapter describes the experimental setup used for the simulations and the experimental results.

4.1. Experiments using NS-2 simulator

In simulation experiments we have observed the performance of PathAB both for single tight-link and multiple tight-link scenarios and compared with some existing available bandwidth estimation algorithms namely Pathload [11], PoissonProb [16], IGI [21], spruce [4], PathChirp [13], and the stochastic model [22]. As Pathload reports the range of available bandwidth instead of a single value, we have averaged the high and low values of the two estimates.

Our proposed algorithm PathAB is a combination of ideas from PathChirp, PoissonProb and MoSeab. So we have compared its performance with PathChirp and PoissonProb. We could not compare it with MoSeab because MoSeab was developed at Microsoft Asia research lab and the authors could not provide us with its implementation due to their corporate regulations (The e-mail communication with the authors has been given in Appendix C). Both PathChirp and PoissonProb are rate-based algorithms. PathAB is a hybrid algorithm. Therefore it has been also compared with the well-known rate-based algorithm Pathload as well as the well-established gap-based algorithms IGI, spruce and the stochastic model [22].

4.1.1. Single Tight-Link Scenario

The network model used for single bottleneck experiments is shown in Figure 4-1. Available bandwidth is measured along the path *Snd* to *Rcv*. The link *R2-R3* is the bottleneck link. We have tested the available bandwidth measurement algorithms with

bottleneck capacity C = 1.5, 5, 10 & 15 Mbps. The bottleneck link has 20ms delay. All the other links have 100 Mbps capacity with 5ms delay. Cross-traffic packets flow from Cs2 to Cd2. To generate cross traffic we have attached 50 Poisson traffic sources with Cs2. If the total cross-traffic rate across the bottleneck link is *r*, then each Poisson traffic source generates traffic with mean rate *r*/50. The packet size of each traffic source is randomly generated between 64 and 1500 bytes (as the minimum size of a UDP packet is 64 bytes and the maximum size is 1500 bytes). Cross-traffic on the returning path is generated from Cs1 to Cd1. To ensure zero packet loss we have used a very high value for the queue length of all routers.

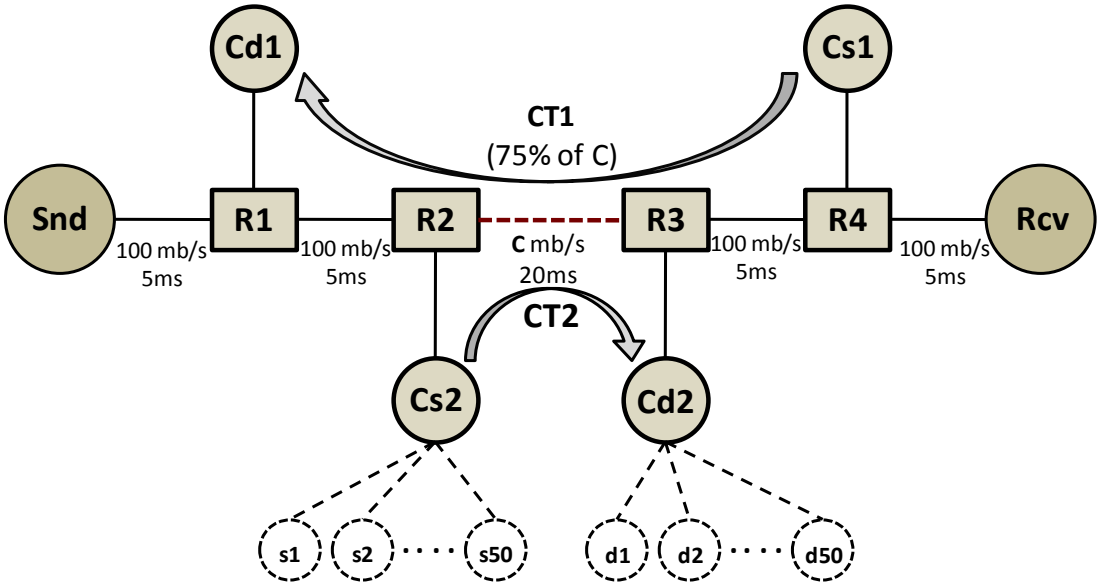

Figure 4-1. Network model for single bottleneck experiments

For each value of the bottleneck capacity, experiments were run for 20, 50, 75 & 90 percent utilization of the bottleneck link.

To avoid synchronization among the cross-traffic packets generated by the 50 Poisson traffic sources, each traffic source started traffic generation at a random instance between 0 and 10 second of the simulation. In each case the available bandwidth estimation process was started at the 10th second of simulation.

We ran all the experiments with the default values of the parameters of the available bandwidth measurement algorithms. For the stochastic model, the algorithm

generally converges after 200 samples in case of CBR traffic, but in Poisson traffic scenario it takes more time to converge. We have observed that in Poisson traffic scenario the available bandwidth estimate by stochastic model becomes stable generally after around 300 samples. So in our experiments for the stochastic model we have taken the available bandwidth estimate value after 300 samples.

We have repeated each experiment 15 to 20 times and have taken the Root Mean Square (RMS) value of the estimated error percentage. The percentage of the estimated error for each experiment has been calculated as:

$$\tilde{E} = \frac{A - \tilde{A}}{A} \times 100 \tag{4.1}$$

where, A is the actual available bandwidth and $\tilde{A}$ is the estimated value of available bandwidth.

- **Estimated error for 1.5 Mbps bottleneck link:**

The comparison of RMS estimated error of PathAB with existing methods on a path with 1.5 Mbps bottleneck capacity has been presented in Table 4-1 and Figure 4-2.

Table 4-1. RMS error % for 1.5 Mbps bottleneck Capacity under different utilization

AB estimation algorithm	% of RMS Error for **1.5 Mbps** bottleneck Capacity under different utilization			
	20%	50%	75%	90%
PathAB (CS)	6.60	8.66	13.11	18.75
PathAB (SA): CT=0	6.73	8.99	14.14	22.31
PathAB (SA): CT= 75%C	7.12	10.15	16.15	33.62
PathChirp	13.22	12.94	15.66	30.63
Spruce	10.33	19.40	22.71	39.47
PoissonProb	22.54	11.78	34.03	52.95
Stochastic model	11.47	11.98	36.12	31.60
IGI	13.14	34.91	64.38	219.06
Pathload	10.05	50.51	149.43	443.75

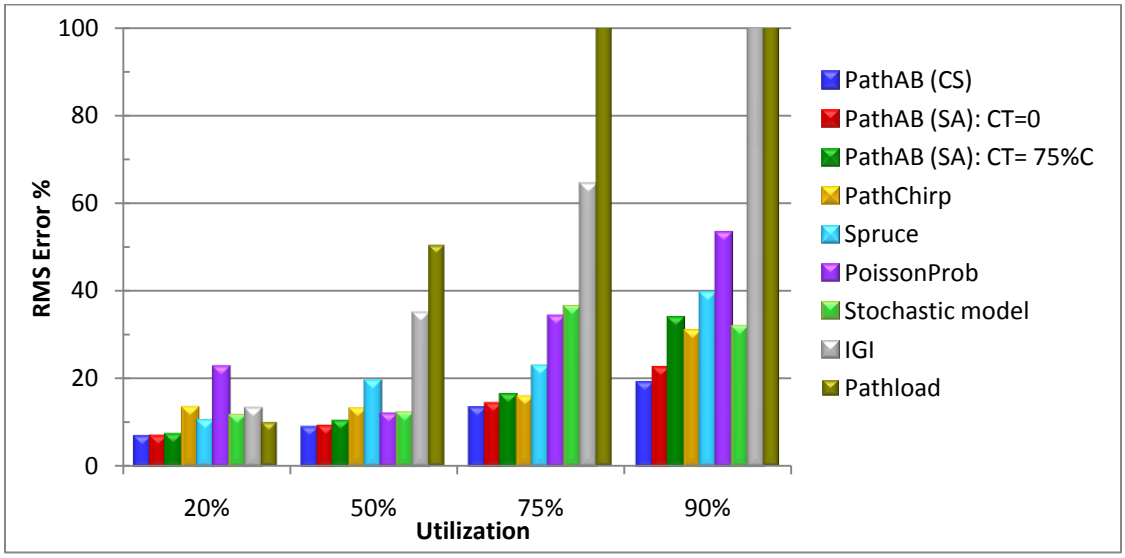

Figure 4-2. RMS error % for 1.5 Mbps bottleneck Capacity under different utilization

- **Estimated error for 5 Mbps bottleneck link:**

The comparison of RMS estimated error of PathAB with existing methods on a path with 5 Mbps bottleneck capacity has been presented in Table 4-2 and Figure 4-3.

Table 4-2. RMS error % for 5 Mbps bottleneck Capacity under different utilization

AB estimate algorithm	% of RMS Error for **5 Mbps** bottleneck Capacity under different utilization			
	20%	50%	75%	90%
PathAB (CS)	6.38	7.03	9.22	13.68
PathAB (SA): CT=0	6.64	9.96	13.35	14.38
PathAB (SA): CT= 75%C	8.77	10.89	11.52	25.17
PathChirp	25.01	10.06	13.01	17.39
Spruce	9.66	23.81	15.55	33.47
PoissonProb	7.45	22.01	22.52	62.47
Stochastic model	10.57	15.44	24.34	25.27
IGI	10.16	40.39	65.06	204.78
Pathload	9.67	22.21	71.29	263.50

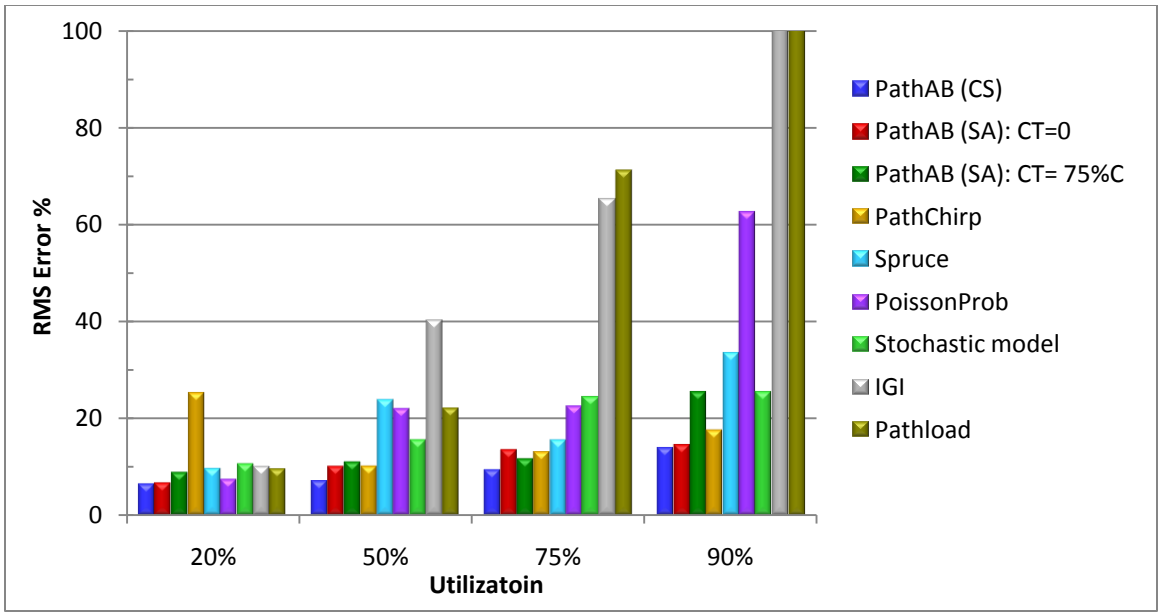

Figure 4-3. RMS error % for 5 Mbps bottleneck Capacity under different utilization

- **Estimated error for 10 Mbps bottleneck link:**

The comparison of RMS estimated error of PathAB with existing methods on a path with 10 Mbps bottleneck capacity has been presented in Table 4-3 and Figure 4-4.

Table 4-3. RMS error % for 10 Mbps bottleneck Capacity under different utilization

AB estimate algorithm	% of RMS Error for **10 Mbps** bottleneck Capacity under different utilization			
	20%	50%	75%	90%
PathAB (CS)	7.61	6.63	8.41	10.55
PathAB (SA): CT=0	8.57	6.89	7.17	13.09
PathAB (SA): CT= 75%C	9.16	8.88	9.47	15.33
PathChirp	10.31	10.54	18.80	23.86
Spruce	13.58	11.18	29.32	33.77
PoissonProb	7.22	26.56	14.04	57.77
Stochastic model	7.95	8.91	30.00	36.36
IGI	11.88	43.76	53.58	135.11
Pathload	8.44	31.80	63.37	192.03

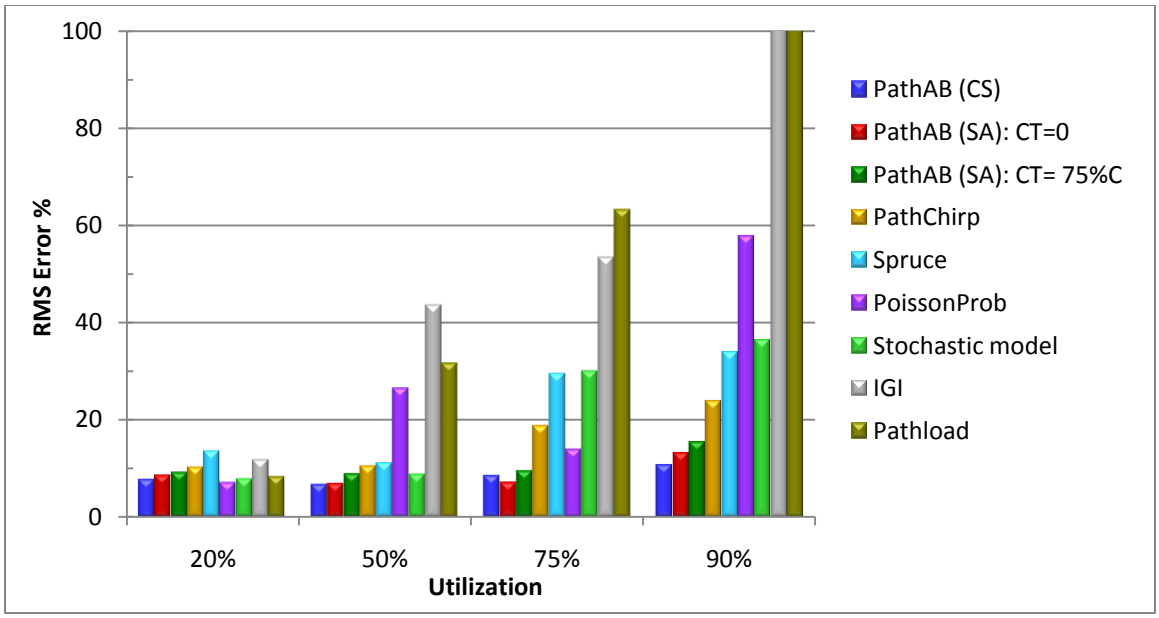

Figure 4-4. RMS error % for 10 Mbps bottleneck Capacity under different utilization

- **Estimated error for 15 Mbps bottleneck link:**

The comparison of RMS estimated error of PathAB with existing methods on a path with 15 Mbps bottleneck capacity has been presented in Table 4-4 and Figure 4-5.

Table 4-4. RMS error % for 15 Mbps bottleneck Capacity under different utilization

AB estimate algorithm	% of RMS Error for **15 Mbps** bottleneck Capacity under different utilization			
	20%	50%	75%	90%
PathAB (CS)	7.44	7.75	10.22	8.65
PathAB (SA): CT=0	8.32	9.74	10.05	10.15
PathAB (SA): CT= 75%C	9.61	9.75	10.50	15.19
PathChirp	11.08	14.82	24.33	17.90
Spruce	11.24	18.71	25.40	29.10
PoissonProb	9.61	12.64	17.61	35.63
Stochastic model	6.50	8.33	30.71	25.46
IGI	14.32	31.91	49.82	93.92
Pathload	7.46	18.20	57.28	176.74

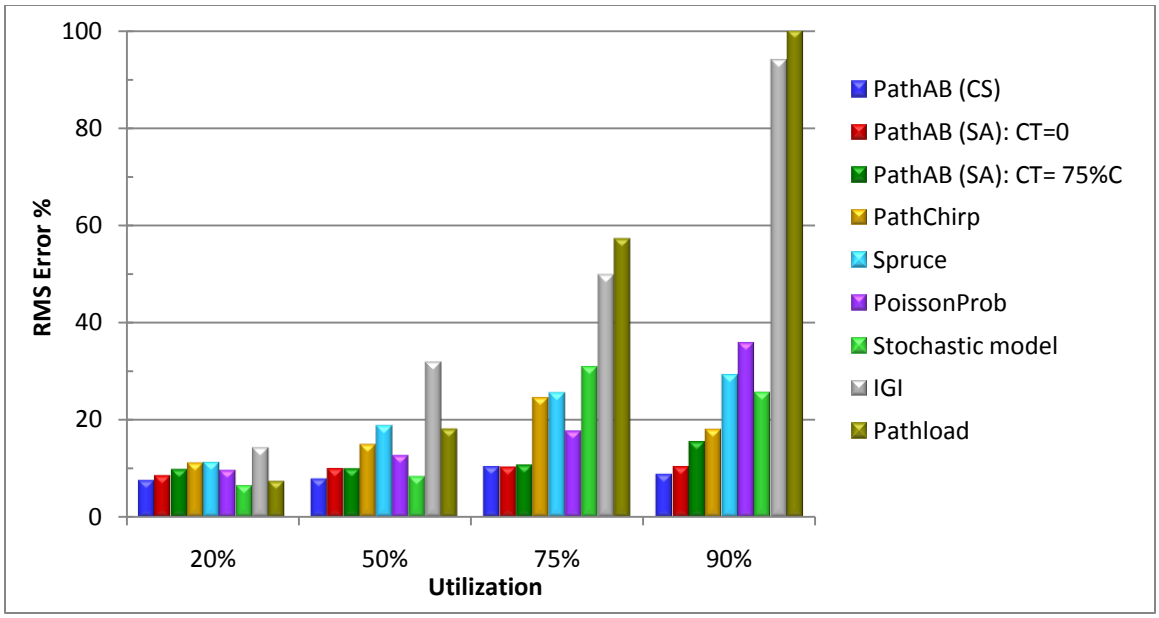

Figure 4-5. RMS error % for 15 Mbps bottleneck Capacity under different utilization

From the simulation results for single bottleneck link scenarios presented in Table 4-1 - Table 4-4, we observe that PathAB exhibits less error compared to all other methods almost in all the cases in both client-server mode and in stand-alone mode. The RMS value of error of PathAB is within 10% in most of the cases except the case when the link utilization is more than 75%. Only in the path with 1.5 Mbps bottleneck capacity, its error is more than 10% in case of 50% link utilization. Only in some cases when link utilization is 50% or less, the estimates obtained using PathChirp, PoissonProb, Spruce and the stochastic model are comparable to those obtained with PathAB. IGI and Pathload can report reasonably good estimates only if the link utilization is less than 50%. These algorithms fail to converge if the path is heavily loaded. As expected, PathAB performs better in the client-server mode than in the stand-alone mode. Also in the stand-alone mode it produces relatively better estimates when there is no cross-traffic in the returning path. Although the estimate of PathAB is slightly worse than that of PoissonProb, PathChirp and the stochastic model under 20% link utilization condition on network path with 10 and 15 Mbps bottleneck capacity, it outperforms all the algorithms we have tested in all other conditions and produces reliable estimates.

4.1.2. Multiple Tight Link: Pre and Post Bottleneck Cross-Traffic Effect

We have performed extensive simulation experiments on NS-2 simulator to observe the effect of pre-bottleneck and post-bottleneck cross-traffic on the available bandwidth measurement algorithms. The objective of these experiments can be summarized as follows:

- Multiple tight links – The second tight link is located *before* the bottleneck link and has the same available bandwidth as the bottleneck link.

- Multiple tight links – The second tight link is located *after* the bottleneck link and has the same available bandwidth as the bottleneck link.

- The tight link is different than the bottleneck link and is located *before* the bottleneck link.

- The tight link is different than the bottleneck link and is located *after* the bottleneck link.

For the pre and post bottleneck simulation experiments we have used the four-hop network topology shown in Figure 4-6. The link *R2-R3* is the bottleneck link of the path with bottleneck capacity C = 10 Mbps and 20ms delay. Both the pre-bottleneck link *R1-R2* and the post-bottleneck link *R3-R4* have 20 Mbps link capacity and 5ms delay. All other links of the topology have 100 Mbps capacity and 5ms delay. The traffic along the bottleneck link is generated from Cs2 to Cd2. Pre-bottleneck traffic is generated from Cs1 to Cd1 and post-bottleneck traffic is generated from Cs3 to Cd3. The available bandwidth is measured across the path *Snd* to *Rcv*. To generate cross traffic we have attached 50 Poisson traffic sources with each of the nodes Cs1, Cs2 and Cs3. If the total cross-traffic rate across any link is r, then each Poisson traffic source attached to that link generates Poisson traffic with mean rate $r/50$.

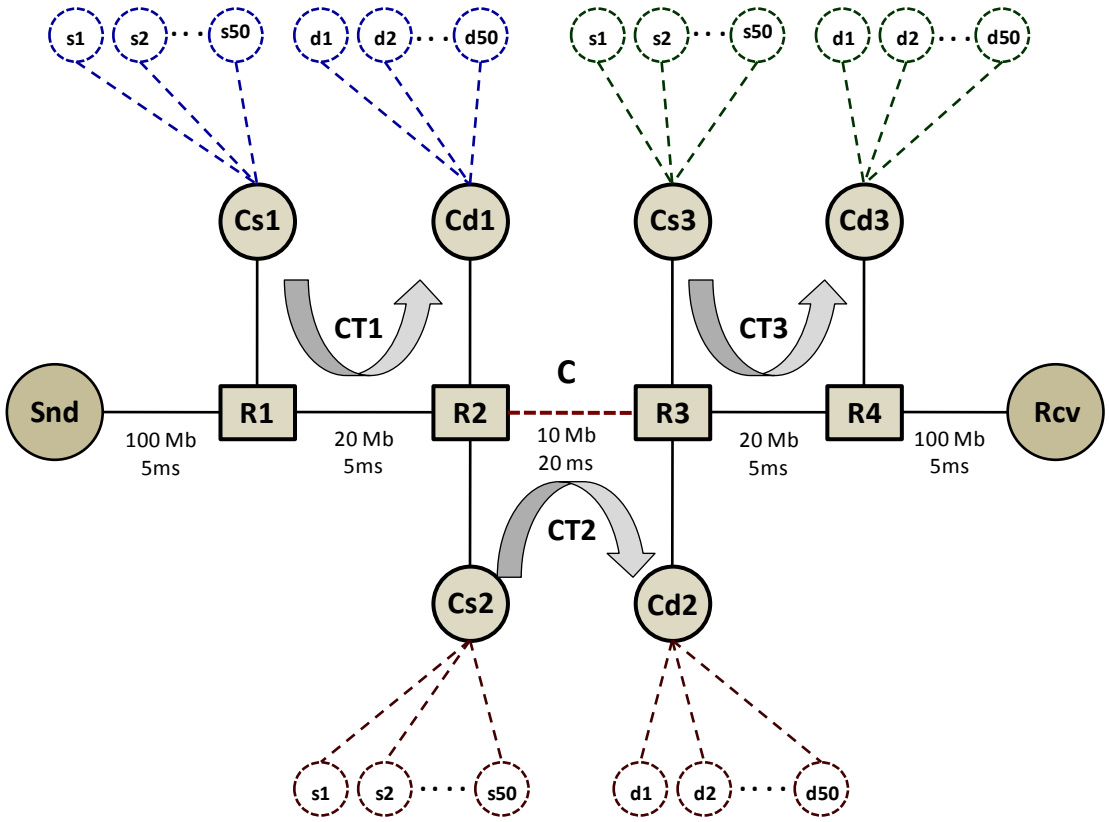

Figure 4-6. Simulation topology for Pre-bottleneck and Post-bottleneck experiments

4.1.2.1. Pre-bottleneck experiment

To observe the pre-bottleneck effect on the available bandwidth estimation algorithms we have kept the cross-traffic rate across the bottleneck link *R2-R3* constant at 3Mbps (i.e. *CT2* = 3Mbps) throughout all the experiments. This makes the AB at the bottleneck fixed at 7 Mbps (as the capacity of the link is 10 Mbps). Post-bottleneck cross-traffic rate across the link *R3-R4* was set to *CT3=0*, that means there was no cross-traffic after the bottleneck link. The cross-traffic rate across the pre-bottleneck link *R1-R2* was increased from 0 Mbps to 19 Mbps. When the pre-bottleneck cross-traffic is less than 13 Mbps, the path has only one tight link which is the bottleneck link *R2-R3*. When pre-bottleneck traffic rate is 13 Mbps both the links *R1-R2* and *R2-R3* have 7 Mbps available bandwidth, resulting in presence of multiple tight links in the path. If pre-bottleneck traffic exceeds beyond 13 Mbps, the links *R1-R2* turns into the tight link as its available bandwidth becomes less than that along the bottleneck link.

We have compared the stand-alone algorithm of PathAB with Pathload, PathChirp, PoissonProb, IGI, Spruce and the stochastic model [22] to observe the pre-bottleneck cross-traffic effect. We have repeated each experiment 10 times and taken the average value of estimated available bandwidths. The experimental results with pre-bottleneck cross-traffic are shown in Table 4-5 and Figure 4-7

Table 4-5. Average estimate of Available Bandwidth with Pre-Bottleneck Cross-traffic

Cross-Traffic	Actual AB	Pathload (Avg.)	IGI	Stochastic Model	PoissonProb	Spruce	PathChirp	PathAB (Stand-alone)
0	7	8.01	6.87	8.00	6.25	7.18	7.45	6.73
1	7	7.96	7.08	6.99	5.86	5.84	7.30	6.60
2	7	7.80	7.34	6.93	6.48	7.22	7.21	6.57
3	7	7.60	8.50	7.45	6.97	7.28	7.76	6.55
4	7	8.00	8.50	7.63	7.20	6.81	6.50	6.65
5	7	7.99	7.72	7.71	6.34	6.70	7.52	7.33
6	7	8.05	7.37	6.66	7.35	6.77	8.40	6.99
7	7	8.00	7.67	7.78	8.01	5.44	7.87	6.58
8	7	8.06	7.79	7.28	5.59	5.57	7.11	6.69
9	7	7.83	7.25	6.57	6.08	5.65	5.74	6.46
10	7	7.64	7.35	7.12	6.39	6.39	6.09	6.54
11	7	7.62	7.78	6.46	6.91	5.35	6.45	6.34
12	7	7.75	7.22	6.68	7.26	5.55	5.13	6.43
13	7	7.48	7.45	6.74	6.53	5.53	6.26	6.45
14	6	6.96	6.87	6.70	6.77	4.71	5.35	5.50
15	5	6.79	6.68	4.70	4.63	5.93	4.65	4.84
16	4	4.97	6.26	5.15	3.80	4.55	3.17	3.58
17	3	5.09	6.00	4.65	2.18	4.55	3.33	2.56
18	2	5.03	6.01	5.07	3.80	4.76	2.62	2.33
19	1	4.50	4.83	4.61	3.70	3.67	2.11	1.43

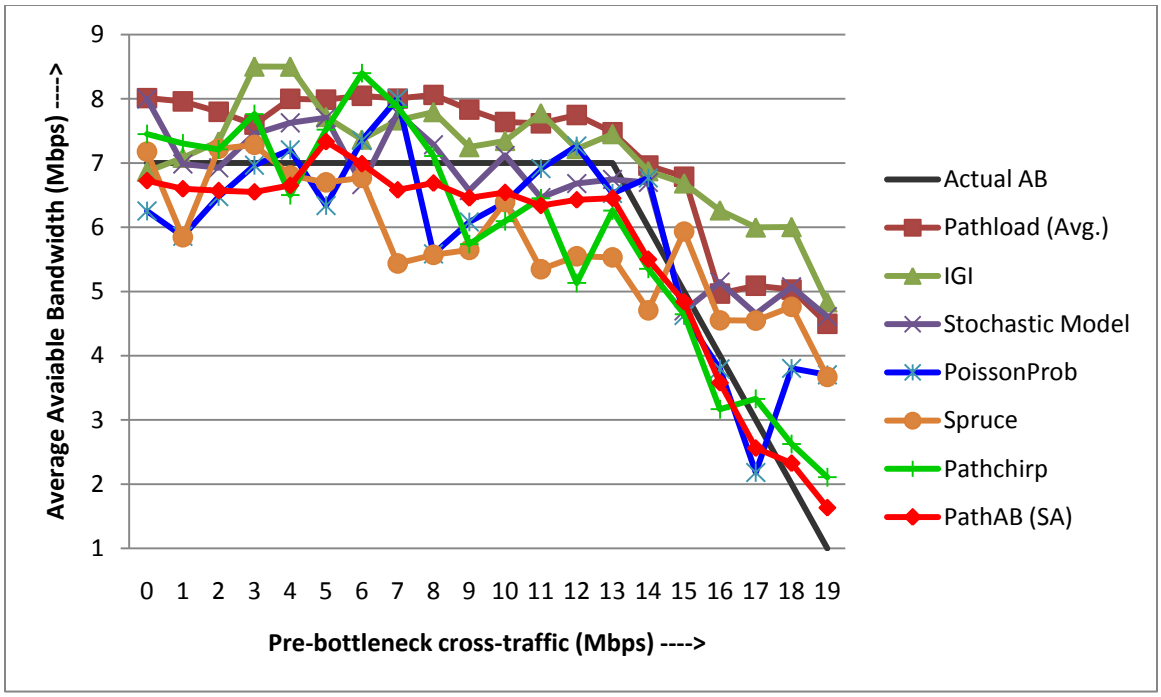

Figure 4-7. Average estimate of Available Bandwidth with Pre-Bottleneck Cross-traffic

From the graph presented in Figure 4-7 we can see that the average estimate of PathAB is much closer to the actual available bandwidth line than other algorithms and it provides a conservative estimate in all cases except when the available bandwidth is 2Mbps or less. Both Pathload and IGI constantly over-estimate the available bandwidth and deviate significantly from the actual AB line in pre-bottleneck tight link scenario. Only the estimate by PoissonProb and PathChirp are comparable to PathAB in pre-bottleneck tight link scenario.

4.1.2.2. Post-bottleneck experiment

To observe the post-bottleneck effect on the available bandwidth estimation algorithms we have kept the cross-traffic rate across the bottleneck link *R2-R3* constant at 3Mbps (i.e. *CT2* = 3Mbps) throughout all the experiments. This makes the AB at the bottleneck fixed at 7 Mbps (as the capacity of the link is 10 Mbps). Pre-bottleneck cross-traffic rate across the link *R1-R2* was set to *CT1=0*. That means that there was no cross-traffic prior to the bottleneck link. The cross-traffic rate across the post-bottleneck link *R3-R4* was increased from 0 Mbps to 19 Mbps. When the post-bottleneck cross-traffic is less than 13 Mbps, the path has only one tight link which is the bottleneck link *R2-R3*. When post-

bottleneck traffic rate is 13 Mbps both the links *R2-R3* and *R3-R4* become tight links as both have 7 Mbps available bandwidth, resulting in the presence of multiple tight links in the path. If post-bottleneck traffic exceeds beyond 13 Mbps, the link *R3-R4* turns into the tight link as its available bandwidth becomes less than that along the bottleneck link.

We have compared the stand-alone algorithm of PathAB with Pathload, PathChirp, PoissonProb, IGI, Spruce and the stochastic model to observe the pre-bottleneck cross-traffic effect. Each experiment has been repeated 10 times and the average values of estimated available bandwidths have been taken. The experimental results with pre-bottleneck cross-traffic are shown in Table 4-6 and Figure 4-8.

Table 4-6. Average estimate of Available Bandwidth with Post-Bottleneck Cross-traffic

Cross-Traffic	Actual AB	Pathload (Avg.)	IGI	Stochastic Model	PoissonProb	Spruce	PathChirp	PathAB (Stand-alone)
0	7	8.01	6.87	8.00	6.57	7.18	7.45	6.73
1	7	7.48	7.95	6.68	7.04	8.22	8.36	6.44
2	7	8.02	7.10	7.32	6.23	7.37	7.91	6.37
3	7	8.06	7.40	7.12	6.82	6.02	7.22	6.48
4	7	8.10	7.10	7.73	5.56	7.78	6.25	6.59
5	7	8.02	6.90	5.82	7.36	7.90	8.18	6.98
6	7	7.54	7.66	7.41	6.45	7.35	7.47	6.23
7	7	8.41	7.08	6.86	5.79	6.55	7.82	6.58
8	7	8.15	7.28	6.84	7.63	7.48	5.95	6.33
9	7	7.92	6.76	6.64	6.25	8.02	6.18	6.87
10	7	8.18	7.05	6.07	6.17	5.65	6.59	6.50
11	7	8.08	6.72	6.25	7.23	5.22	7.54	6.13
12	7	7.87	6.88	6.63	5.19	6.22	6.43	6.55
13	7	7.55	7.70	4.80	5.45	5.45	6.40	6.12
14	6	6.79	5.32	6.97	5.60	4.93	5.16	5.46
15	5	6.69	5.04	6.21	4.34	4.91	5.47	4.49
16	4	6.27	6.12	5.28	3.12	5.49	2.99	4.23
17	3	5.64	4.52	5.90	3.76	4.19	2.50	2.52
18	2	5.52	4.21	4.70	3.01	3.55	2.99	2.41
19	1	4.66	5.05	5.14	3.82	3.99	2.18	1.74

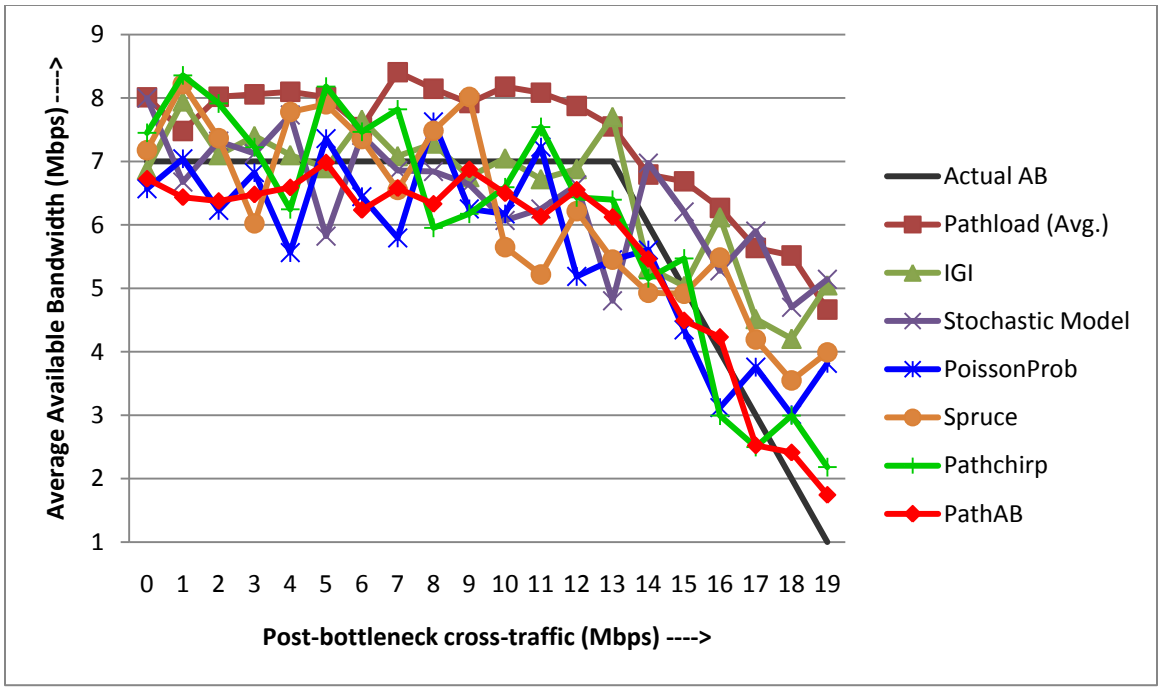

Figure 4-8. Average estimate of Available Bandwidth with Post-Bottleneck Cross-traffic

All the algorithms perform better in the post-bottleneck tight link scenario and the estimates are much closer to the actual available bandwidth line. The reason for this is that cross-traffic of the link closest to target host significantly affects the probe traffic. If there is significant amount of cross-traffic after the tight link, then the inter-packet intervals created within the probing train might be altered by the traffic after and the probe traffic may not be able to preserve the tight link's traffic information. Similar to the pre-bottleneck experiments, we have found that estimates by PathAB are more accurate than those obtained by other algorithms. Also in this scenario the estimates by PathChirp and PoissonProb are comparable to those by PathAB.

4.2. Experiments on Network TestBed

Beside the NS-2 simulation experiments, we have also tested the performance of PathAB on a network test-bed in our Lab and compared its performance with PathChirp, PoissonProb, IGI, spruce and Pathload. The implementations of these algorithms were obtained from the authors' website. We have observed the performance of the above algorithms for both single-hop and multi-hop path with multiple congested links. For the

single-hop path, the experiments were performed with 10 Mbps link. For multi-hop paths, we have performed experiments with two different network setups, one with 10 Mbps bandwidth range and another with only 100 Mbps links. Due to the limitation of the routers available in our lab, we could not perform experiments on networks with higher link capacity. In the simulation experiments, we used infinite length router queue to ensure zero packet loss. But in reality zero packet loss is almost impossible to achieve. Length of packet queue is limited by the amount of memory available in the routers. Therefore if the traffic rate across a link is higher than its capacity, there is a high probability that some of the probe packets might be dropped by the router. This in turn affects the performance of bandwidth measurement algorithms. In all our experiments we have used Cisco 2651xm routers with 256MB DRAM to setup the network test-beds. In the following part of this section we describe the network topology and the experimental results obtained in the above three scenarios.

4.2.1. Single-hop Experiments

4.2.1.1. Description of Network TestBed

The topology of network test-bed used for the single-hop experiments is shown in Figure 4-9. The link between routers R1 and R2 has 10 Mbps link capacity. All the links connecting a router with a host have 100 Mbps capacity. The cross traffic packets are generated from host H3 to host H4. Available bandwidth is measured along the path from H1 to H2. The server programs of AB estimate tools are installed on host H2 and the client programs is installed on host H1.

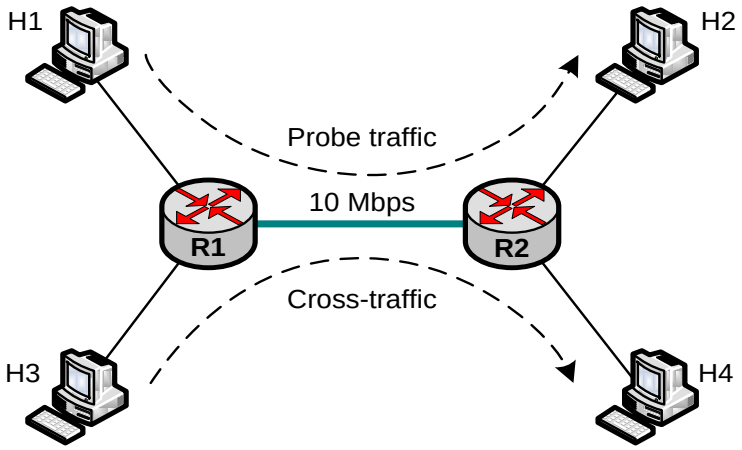

Figure 4-9. Network topology for single-hop experiments

4.2.1.2. Results of Single-hop Experiments

For single-hop experiments we have compared both stand-alone and client-server algorithms of PathAB with PathChirp, PoissonProb, IGI, spruce and Pathload. The experiments were run with cross-traffic rates of 0, 2, 4, 6, 8 and 9 Mbps resulting in 10, 8, 6, 4, 2 and 1 Mbps of available bandwidth respectively along the path. Each experiment was repeated 20 times and the RMS value of the estimated error percentage has been considered for comparison. The estimated errors for single-hop experiments are presented in Table 4-7 and Figure 4-10.

Table 4-7. Comparison of AB estimate algorithms for single-hop path with 10Mbps capacity

AB Estimate Algorithm	RMS Error % for different values Available Bandwidth					
	10	**8**	**6**	**4**	**2**	**1**
PathAB (SA)	5.72	7.26	9.62	8.87	16.59	24.39
PathAB (CS)	5.59	6.76	7.87	7.98	14.58	23.99
PoissonProb	9.73	12.10	10.36	15.93	24.99	-
PathChirp	4.37	7.91	16.36	34.17	25.44	-
IGI	18.79	15.55	10.38	17.01	71.32	-
Spruce	5.24	6.22	24.53	60.45	240.53	-
Pathload	14.00	4.55	14.20	49.16	209.77	-

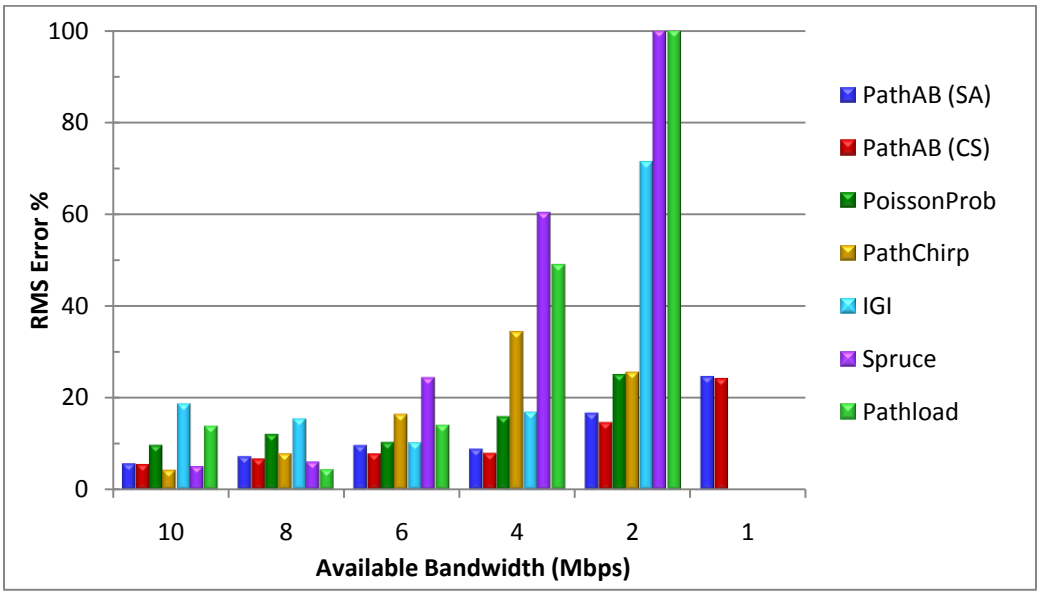

Figure 4-10. Comparison of AB estimate algorithms for single-hop path with 10Mbps capacity

From the experimental results we observe that the estimated error of PathaAB is always less than 10%, if the available bandwidth is greater than 2 Mbps. In some cases, for example when AB is 10Mbps PathChirp and spruce perform slightly better than PathAB. Although spruce and Pathload have relatively less estimated error than PathAB when available bandwidth is 8Mbps, their estimated error increases rapidly with the increase of cross-traffic rate and is more than 50% when the available bandwidth becomes less than 6Mbps. Other than PathAB, only the PoissonProb's estimate error is almost steady in all cases. Also we have observed that all the algorithms except PathAB fail to report any value of bandwidth when the available bandwidth becomes less than 2 Mbps. The reason of their failure is that all these algorithms transmit probe packet trains at a very high rate which is much higher than the available bandwidth of the path in this case. Due to the high value of cross-traffic, the arrival rate of packets, combining cross-traffic and probe packets, exceeds the capacity of router queue resulting in too much loss of probe packets. PathAB on the other hand transmits only one train with exponentially increasing probe rate and sets the input rates of subsequent trains according to the rough estimate obtained in the first phase. We have found that like all other algorithms, PathAB also fails to report any value for the rough AB in the first phase because of packet loss in

the exponential train. If the first phase fails to report any value, PathAB assumes that the available bandwidth is less than 2 Mbps and randomly chooses a value around 2 Mbps as the rough AB. Once it obtains some value from the first phase, the second phase can continue and estimate the available bandwidth of the path.

4.2.2. Multi-hop Experiment: 10 Mbps range

4.2.2.1. Description of Network TestBed

The network test-bed used for multi-hop experiments is shown in Figure 4-11. The link between routers R2 and R3 is the bottleneck link with capacity of 8 Mbps. Both pre-bottleneck link R1−R2 and post-bottleneck link R3−R4 have 10 Mbps capacity. All the links connecting any router with a host have 100 Mbps link capacity. Cross-traffic along the bottleneck link (CT2) is generated from host H2 to H3. Pre-bottleneck cross-traffic CT1 is generated from host H1 to H2 and post-bottleneck traffic is generated from host H3 to H4. The available bandwidth is measured along the path from H_S to H_D where H_S is the sending host and H_D is the destination host.

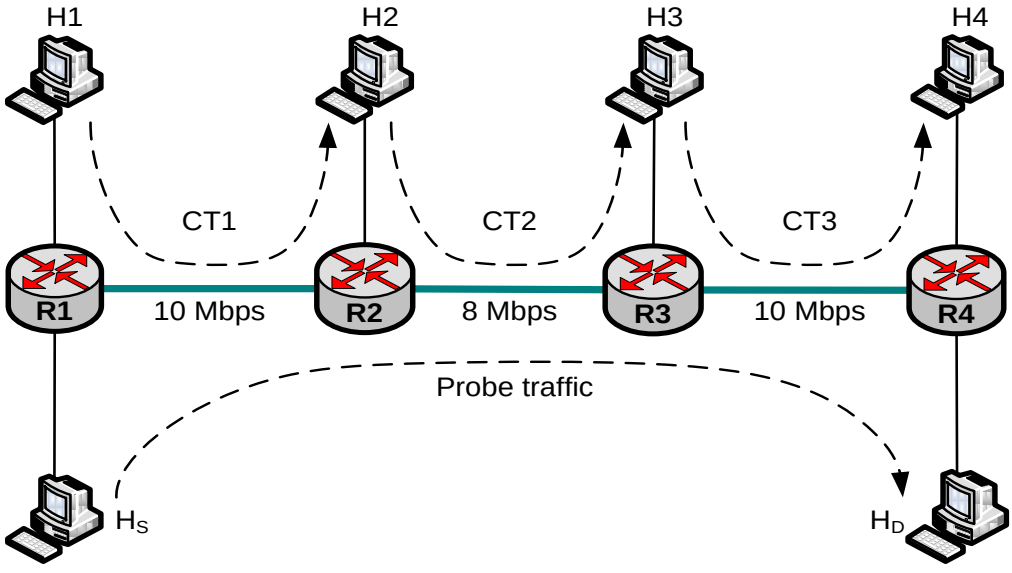

Figure 4-11. Network topology for multi-hop experiments

4.2.2.2. Experimental Results

The objective of multi-hop experiments on the network test-bed are the same as in the NS-2 simulation experiments, to observe the performance of PathAB under pre-bottleneck and post-bottleneck cross traffic conditions and in multiple tight link condition and compare it with PathChirp , PoissonProb, IGI, spruce and Pathload. Cross-traffic CT2 along the bottleneck link R2−R3 is generated from host H2 to H3 and the rate is kept constant at 3 Mbps. This makes the available bandwidth of the bottleneck link as 5 Mbps.

- **Pre-bottleneck traffic:**

To observe pre-bottleneck cross-traffic effect, the cross-traffic CT3 along the post-bottleneck link R3−R4 is kept 0. Pre-bottleneck traffic CT1 across the link R1−R2 is generated from host H1 to H2. We have run experiments with pre-bottleneck traffic CT1= 0, 2, 4, 5, 6, 8 Mbps resulting in 10, 8, 6, 5, 4, 2 Mbps available bandwidth in the pre-bottleneck link. When the cross-traffic is less than 5 Mbps, the bottleneck link is the tight link. At 5 Mbps traffic at the pre-bottleneck link, both the R1−R2 and R2−R3 become tight links resulting in multiple tight links. When traffic increases beyond 5 Mbps the pre-bottleneck link becomes the tight link. Each experiment was repeated 20 times and the RMS values of all the estimated errors were considered for comparison. Table 4-8 and Figure 4-12 presents the estimated errors obtained from pre-bottleneck experiments.

Table 4-8. RMS error % of pre-bottleneck experiments

Algorithm	Pre-bottleneck cross-traffic (Mbps)					
	0	**2**	**4**	**5**	**6**	**8**
PathAB (SA)	6.86	7.96	9.35	10.91	13.96	25.15
PathAB (CS)	6.75	6.96	8.10	9.49	13.37	24.78
PoissonProb	9.24	9.09	11.87	13.63	8.33	38.87
PathChirp	12.32	14.21	14.87	20.12	42.25	123.94
IGI	5.92	7.83	9.64	12.69	18.04	60.04
Spruce	10.20	20.19	19.68	17.61	20.58	100.36
Pathload	22.46	25.46	28.32	25.07	49.26	158.82

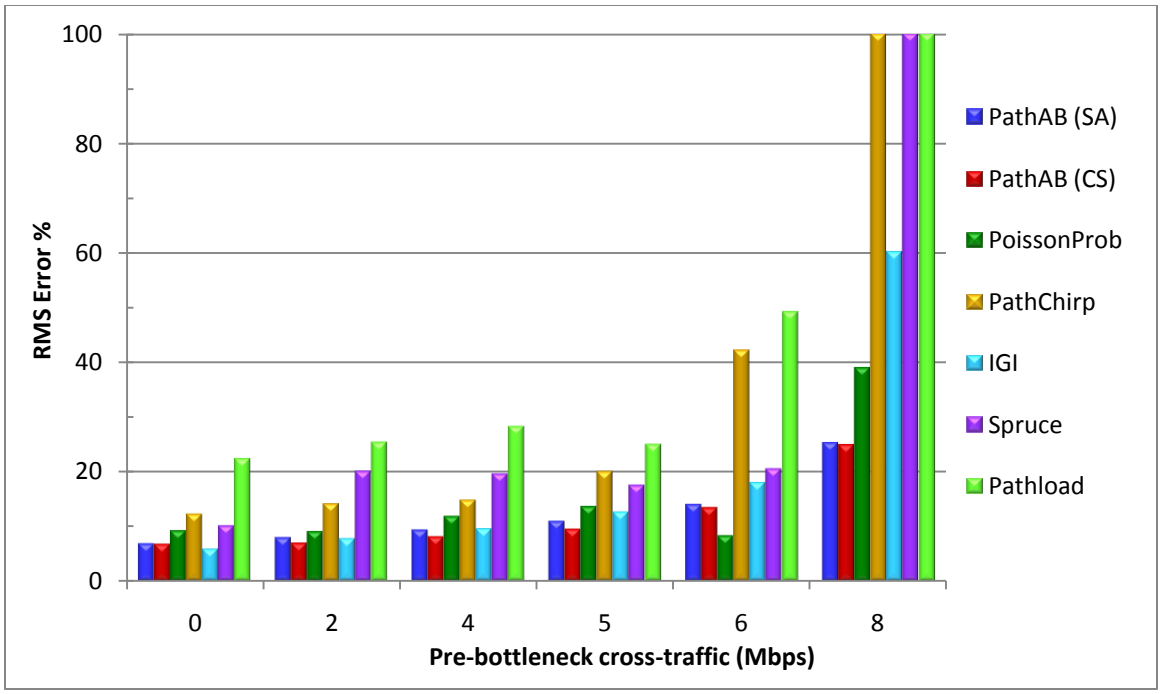

Figure 4-12. Comparison of RMS error % for pre-bottleneck experiments

From the above figure, we can observe that PathAB performs better than all the other algorithms in almost all cases. As expected the client-server algorithm of PathAB has slightly better performance than the stand-alone mode. Only in zero pre-bottleneck condition IGI and at 6 Mbps pre-bottleneck traffic, PoissonProb give better estimates than PathAB. In all the cases only PoissonProb's estimate is closer to that of PathAB. The estimated error of Pathload is very high than other algorithms in all the cases. Surprisingly the PathChirp algorithm performs poorly and the estimated error increases rapidly as the available bandwidth decreases below 5 Mbps. The explanation is that PathChirp injects packet trains with exponentially increasing probing rate and as the links have low capacity, higher number of probe packets gets dropped by the routers.

- **Post-bottleneck traffic:**

For post-bottleneck experiments the pre-bottleneck traffic CT1 across link R1−R2 is kept zero throughout all the experiments. The post-bottleneck traffic CT3 is increased gradually. Post-bottleneck experiments were run with the values of CT3 as 0, 2, 4, 5, 6 and 8 Mbps. The RMS estimate errors of post-bottleneck experiments are shown in Table 4-9 and Figure 4-13.

Table 4-9. RMS error % of post-bottleneck experiments

Algorithm	Post-bottleneck cross-traffic (Mbps)					
	0	**2**	**4**	**5**	**6**	**8**
PathAB (SA)	6.89	8.41	11.32	11.55	12.93	22.69
PathAB (CS)	6.75	7.85	11.13	10.61	10.23	18.03
PoissonProb	9.24	13.79	12.33	14.03	15.38	-
PathChirp	12.32	11.50	12.35	12.94	21.94	120.91
IGI	5.92	8.04	8.50	12.62	13.37	46.78
Spruce	10.20	16.66	13.08	21.54	22.49	32.36
Pathload	22.46	21.45	19.50	16.75	35.50	124.92

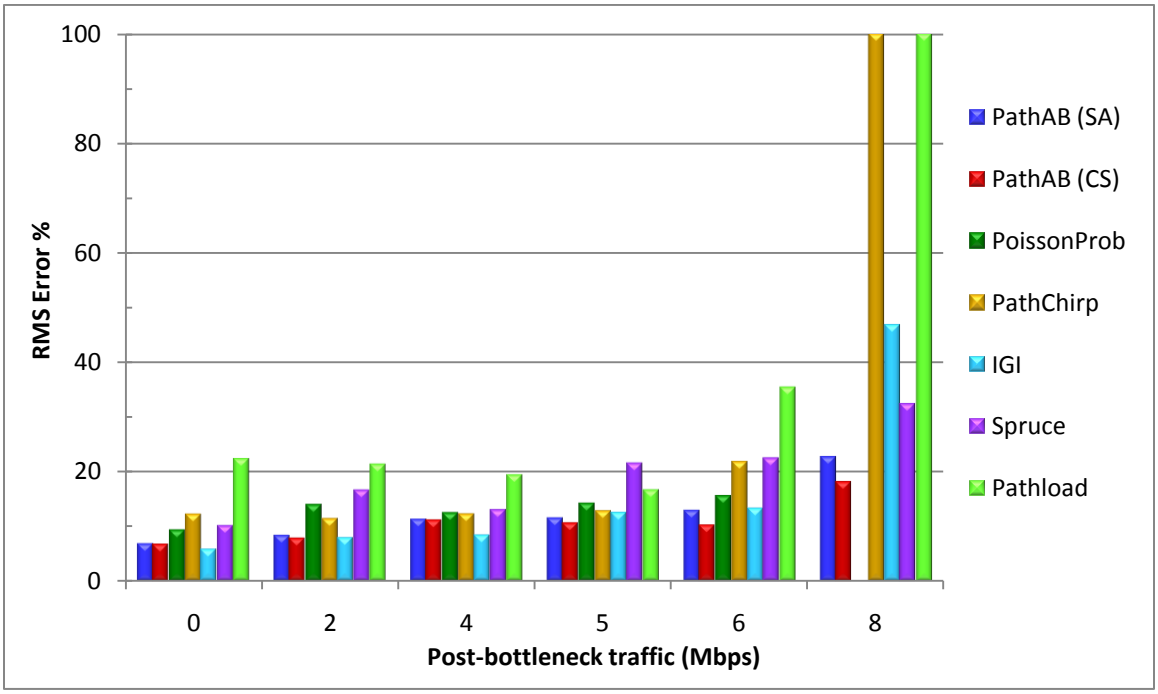

Figure 4-13. Comparison of RMS error % for post-bottleneck experiments

From figures Figure 4-13 and Figure 4-12 we can observe that all the available bandwidth measurements perform better in post-bottleneck conditions than in pre-bottleneck cross-traffic conditions, which is an expected scenario. The reason behind this is the same as explained in section 4.1.2.2. We can see that in post-bottleneck scenario as well, PathAB performs better than all other algorithms in almost all cases. Only IGI has similar or better performance than PathAB when post-bottleneck traffic is equal or less than 5 Mbps, i.e., when the bottleneck link is the tight link. The estimated errors of all

other algorithms except Pathload are also close to PathAB in these cases. But their performance drops when post-bottleneck traffic increases and the bottleneck link no longer remains the tight link. Similar to pre-bottleneck condition, PathChirp shows very high estimated error when cross-traffic is 8 Mbps. We have noticed that the PoissonProb algorithm cannot at all estimate the available bandwidth and stops execution, reporting "too much link congestion" when post-bottleneck traffic is 8 Mbps.

Both pre and post bottleneck experiments show that PathAB outperforms most of the available bandwidth estimate algorithms in almost all cases. The RMS estimated error is within 10%, when available bandwidth is more than 4 Mbps and always within 25%.

4.2.3. Multi-hop Experiment: 100 Mbps range

4.2.3.1. Description of Network TestBed

The network topology used for these experiments is the same as the one used for experiments in section 4.2.2 which is shown in Figure 4-11. The only difference is all the links, connecting any two routers or a host with a router, have 100 Mbps link capacity. As all the links are of the same capacity, there is actually no bottleneck link. But to perform the multi-hop experiments we have assumed the middle link R2−R3 to be the bottleneck link and performed pre-bottleneck and post-bottleneck experiments. For both types of experiments the cross-traffic rate CT2 across the link R2−R3 was kept constant at 30 Mbps, leading to 70 Mbps available bandwidth at the bottleneck link. To observe the pre-bottleneck effect the cross-traffic rate CT1 across the link R1−R2, generated from host H1 to H2, has been increased from 0 to 90 Mbps, while keeping no cross-traffic across link R3−R4. Obviously when CT1 becomes higher than 30 Mbps, the link R1−R2 becomes the tight link of the path. For post-bottleneck scenario the cross-traffic CT1 across link R1−R2 was kept zero and the cross-traffic CT3 from host H3 to H4 across the link R3−R4 was increased from 0 to 90 Mbps. Again, when the cross-traffic increases above 30 Mbps, the link R3−R4 becomes the tight link. To simulate the Internet traffic we have used Poisson traffic and for generating Poisson traffic we have used the *Distributed Internet Traffic Generator* (D-ITG version 2.6.1d) [37] obtained from the website http://www.grid.unina.it/software/ITG/.

4.2.3.2. Experimental Results

We have performed extensive experiments to observe the performance of PathAB on 100 Mbps multi-hop path for pre and post bottleneck scenarios and compared with PoissonProb, PathChirp and IGI. Each experiment was repeated 20 times and the average of all estimated available bandwidth along with the Root-Mean-Square (RMS) error percentage of the estimates were considered to compare the performances of these algorithms. We have observed that PathAB performs better in 100Mbps path when the size of probe packet is 1500 bytes. Therefore for these experiments we have run the PathAB algorithm with 1500 byte probe packets. Also for the initial probing phase the instantaneous probing rate of exponential train was increased from 1Mbps to 200Mbps with spread factor 1.2. The necessity for using larger probe packets has been discussed in section 4.3.

- **Pre-bottleneck effect**

The average estimated available bandwidths by PathAB in stand-alone (SA) mode and client-server (CS) mode, PoissonProb, PathChirp and IGI algorithms for pre-bottleneck experiments for different cross-traffic rates (CT) are presented in Table 4-10 and Figure 4-14. The comparison of RMS error % of these algorithms is shown in Figure 4-15.

Table 4-10. Average estimated AB by different algorithms in 100Mbps multi-hop path under pre-bottleneck traffic

CT	Actual AB	PathAB (SA)	PathAB (CS)	PoissonProb	PathChirp	IGI
0	70	67.76	69.58	73.25	67.38	60.88
10	70	68.23	68.32	72.22	66.51	64.91
20	70	69.52	68.99	70.97	72.12	62.37
30	70	68.95	70.11	71.87	68.87	61.05
40	60	57.05	58.13	56.97	64.33	53.11
50	50	47.31	47.32	48.37	54.85	55.77
60	40	39.60	38.97	37.73	45.94	50.19
70	30	30.17	29.37	27.53	33.21	48.37
80	20	19.16	19.27	22.28	23.67	37.19
90	10	14.56	13.67	13.76	17.13	28.47

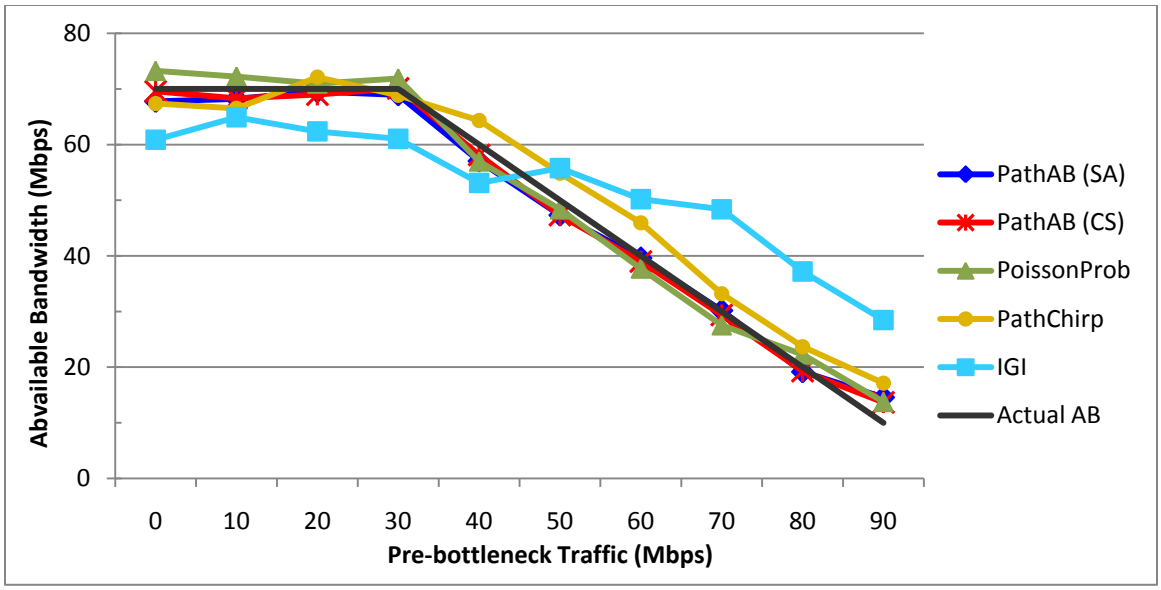

Figure 4-14. Comparison of average estimated AB by different algorithms in 100Mbps multi-hop path under pre-bottleneck traffic

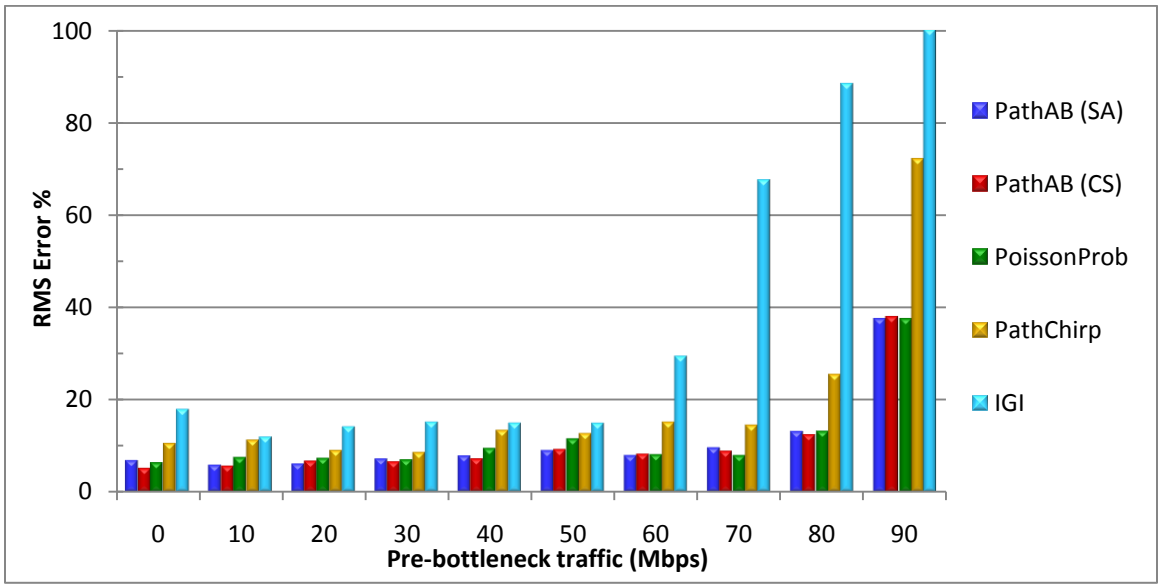

Figure 4-15. Comparison of RMS Error % of estimated AB by different algorithms in 100Mbps multi-hop path under pre-bottleneck traffic

From Figure 4-14 we can observe that the average estimate of PathAB both in the stand-alone mode and in the client-server mode and PoissonProb algorithms are very close to the actual available bandwidth line. PathChirp performs better when pre-bottleneck traffic is less than bottleneck traffic but when the traffic rate increases, it continuously over estimates the available bandwidth. The IGI algorithm under-estimates

the available bandwidth of bottleneck link in the presence of pre-bottleneck traffic but highly over-estimates path's AB when the pre-bottleneck link becomes the tight link. These results match the results presented in [16]. Figure 4-15 shows that the RMS error for PathAB (both in stand-alone and client-server modes) and PoissonProb have similar error rates in all cases and are much less than the other algorithms. In almost all cases PathAB's estimated error is less than PoissonProb, except the cases when pre-bottleneck traffics are 70 and 90 Mbps where PoissonProb performs slightly better than PathAB. As expected, the client-server version of PathAB has a little better performance over the stand-alone version.

- **Post-bottleneck effect**

Table 4-11 and Figure 4-16 presents the average estimates by different algorithms with increasing post-bottleneck traffic rates (CT). The comparison of RMS error percentages of these estimates is shown in Figure 4-17.

Table 4-11. Average estimated AB by different algorithms in 100Mbps multi-hop path under post-bottleneck traffic

CT	Actual AB	PathAB (SA)	PathAB (CS)	PoissonProb	PathChirp	IGI
10	70	68.95	68.32	72.43	76.75	50.44
20	70	72.15	71.55	72.85	76.99	49.12
30	70	70.71	69.07	73.21	75.12	55.78
40	60	57.55	61.13	56.14	65.67	56.89
50	50	46.88	48.01	47.97	53.09	55.30
60	40	37.67	37.92	33.58	46.11	37.09
70	30	27.07	28.25	23.14	37.87	36.59
80	20	18.19	21.22	22.21	17.15	28.70
90	10	8.92	12.34	12.76	15.55	21.77

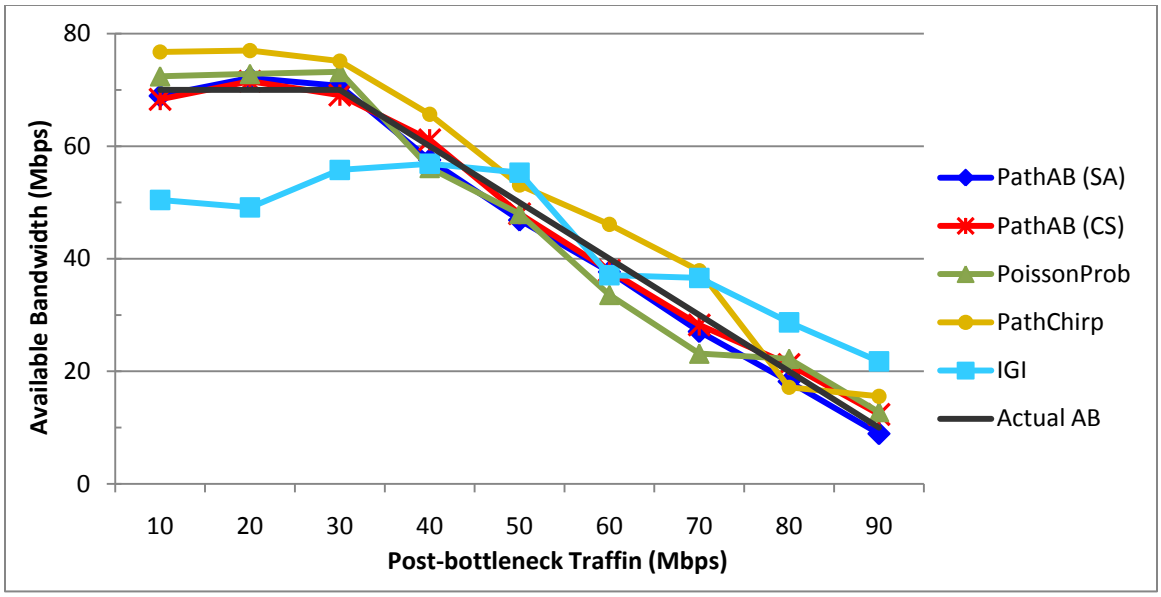

Figure 4-16. Comparison of average estimated AB by different algorithms in 100Mbps multi-hop path under post-bottleneck traffic

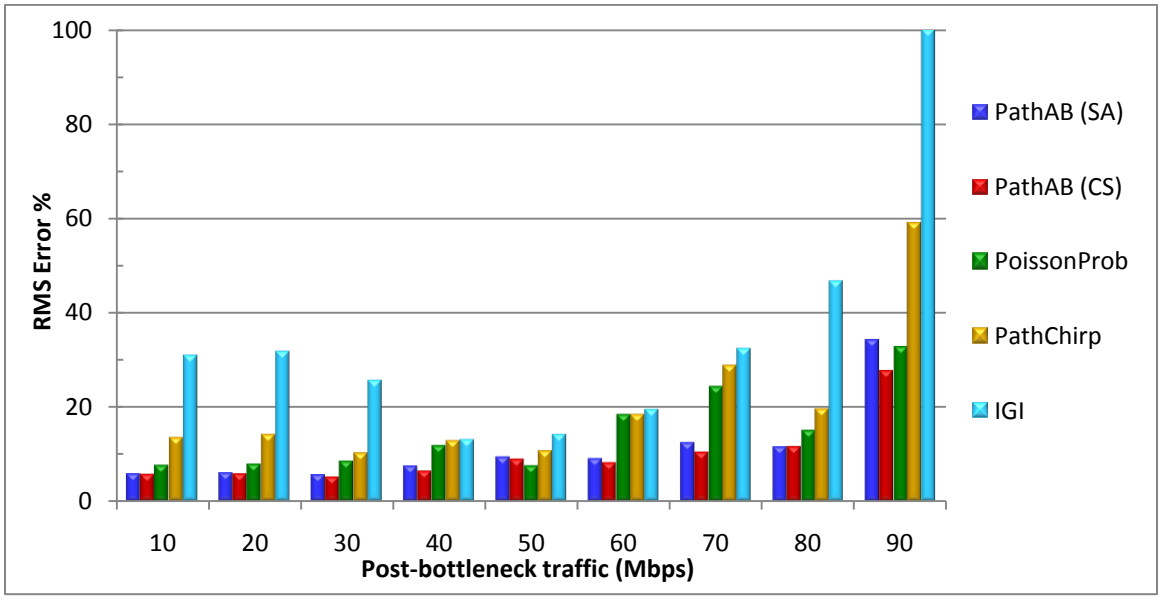

Figure 4-17. Comparison of RMS Error % of estimated AB by different algorithms in 100Mbps multi-hop path under post-bottleneck traffic

From Figure 4-16 we can see that similar to pre-bottleneck experiments, in this case also the average estimate of PathAB (both in stand-alone and in client-server modes) and PoissonProb are very close to the actual available bandwidth line. The PathChirp algorithm constantly over-estimates the available bandwidth. Similar to the previous case,

IGI again under-estimates the AB when post-traffic is less and over-estimates for higher post-bottleneck traffic rates, but the estimated errors for the later cases are a little less than those for the pre-bottleneck scenario. Figure 4-17 shows that the RMS error % of PathAB is always within than 10-12% except when the cross-traffic rate is 90 Mbps. Although the average estimates of PathAB and PoissonProb are almost similar, PoissonProb has slightly more RMS error in all cases except in 50Mbps cross-traffic condition. It can, therefore, be inferred that PathAB performs better than other algorithms and has less RMS estimated error in almost all cases. Also as in all other previous experiments, in this case of 100 Mbps path also, we have found that PathAB presents a conservative estimate of the available bandwidth for both pre and post bottleneck scenarios.

4.3. Effect of Probe-Packet Size on Estimation Accuracy

Size of the probe packet is an important parameter for almost all available bandwidth measurement algorithms, especially for those which are based on inter-packet gaps for the estimation process. The main idea of this kind of algorithm is that, if the probing rate is higher than the available bandwidth, then some cross-traffic packets will be queued up behind the first packet, while it is being processed by the link with capacity C, before the next probing packet arrives. This in turn will cause an increase in the gap between those packets at the receiver. The probing rate r is calculated as,

$$r = \frac{q}{\Delta} \tag{4.2}$$

where, q is the size of probe packet and Δ is the inter-packet gap and the processing time of the packet by link's router is q/C. Generally for this type of algorithms Δ should be less than q/C.

$$\Delta \leq \frac{q}{C} \tag{4.3}$$

It is obvious that to increase the probing rate we have two choices; decreasing the inter-packet gap Δ or increasing the packet size q. Now if we assume fluid model for the cross-traffic, i.e., the cross-traffic packets are of infinitely small size and the inter-packet intervals are almost zero; then it will not affect the available bandwidth measurement

process whether we increase the probe rate by increasing probe packet size or by decreasing the inter-packet gaps, because no matter how small the gap is, at least some cross-traffic packets will arrive within that interval. But in reality the Internet traffic does not follow the fluid model and the arrival process of traffic packets is discrete in nature [36]. So, if the inter-packet gap is too small then no cross-traffic packet may arrive at all during that interval which may lead to wrong estimation. Again, from (4.3) we can see that for same probe-packet size, if the link capacity increases, the inter-packet gap will decrease. Therefore choosing appropriate probe packet size is very important for available bandwidth estimation algorithms. Pasztor and Veitch [37] showed that the correctness of bandwidth estimation algorithms has a linear relationship with the size of probe packet and the accuracy of estimation improves with the increase in packet size upto a certain size and the accuracy saturates after that (as shown in Figure 4-18).

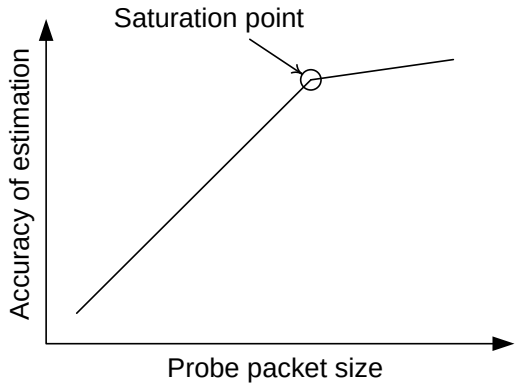

Figure 4-18. Probe-packet size vs. Accuracy of estimation

They found that for link with capacity less than 2 Mbps the saturation point is around 500 bytes and for 10 Mbps links it is around 1100 bytes. We have also observed the same kind of nature of probe size dependence in our experiments. All the available bandwidth estimation algorithms generally use 1200 or 1300 byte probe packets by default. For PathAB we have used 1200 byte probe packets for network path with 10 Mbps rage links. But we found that it provides better estimation for 100 Mbps links when the probe packet size is 1500 bytes.

CHAPTER V

CONCLUSION AND RECOMMENDED FUTURE WORK

In this thesis we have presented a new algorithm, called PathAB, to estimate the available bandwidth of an end-to-end network path. PathAB is a hybrid algorithm which is mainly based on the strong mathematical foundation of MoSeab, but also borrows ideas from two other methods, PathChirp and PoissonProb to improve its performance and reduce traffic overload. It is a hybrid algorithm in the sense that it uses both rate-based and gap-based approaches for the measurement process. The algorithm operates in two phases; the first phase is a rate-based approach where it transmits a single exponential packet train to rapidly obtain a rough estimate and in the next phase, which is a gap-based approach, it transmits several Poisson distributed probing trains with different mean inter-packet intervals to obtain the final estimate. Another attractive feature of PathAB is that, it can also operate in stand-alone mode without any assistance from the target host. Unlike all other existing stand-alone algorithms, instead of echoing the large probe packets, PathAB uses very small 28 byte ICMP echo packets which are transmitted right behind the large probe packets. The probe packets are dropped at the target host and the sender estimates available bandwidth after receiving back the echo packets.

The client-server and stand-alone algorithms of PathAB have been compared with some existing algorithms, such as PoissonProb, PathChirp, IGI, Pathload and spruce using NS-2 simulations and on the network test-bed under different topology and cross-traffic scenarios. We have observed that PathAB performs better and poses relatively less RSM error both in the client-server and stand-alone modes than the other algorithms in almost all the test cases. For both 10Mbps and 100Mbps paths the RMS error of PathAB is within 10% in most of the cases and within 15% in a few cases. But the error is more than 20% when utilization of the path is 90% or more.

We have found that PathAB requires different values for its parameters, the probe packet size for both phases and the min_rate & max_rate of exponential train in the first phase, to produce better estimates in 10Mbps and 100Mbps paths. For 10Mbps path 1200 byte probe packets were used and the rate of exponential train was increased from 10kbps

to 100Mbps, whereas for 100Mbps path the size of probe packets was 1500 bytes and the rate of exponential train was increased from 1Mbps to 200Mbps. In the current implementation of PathAB we have to change these parameters manually to fit the algorithm appropriately for 10Mbps and 100Mbps path. One possible improvement of PathAB can be to run some prediction algorithms to first predict bandwidth range of the path and adjust the parameters automatically.

PathAB calculates the path's available bandwidth based on all the 15 samples received in the second phase. As the utilization of the path increases, there is a high possibility that some of these samples may become affected due to packet drops or sudden unexpected traffic burst and this in turn may affect the final estimate. PathAB's estimate in such scenario can be further improved by applying some filtering mechanism to ignore the noisy samples.

Also we have assumed Poisson traffic pattern across the path. We have not observed the performance of PathAB when the traffic pattern changes to self-similar, pareto, exponential or something else. Although the performance of PathAB should be similar to that with Poisson traffic condition, if the traffic is CBR or uniform, but it may fail in other situations. The open area of further research is to observe the performance of PathAB under different traffic patterns and adjust the structure of probing train to adapt to different scenarios.

REFERENCES

[1] K. C. Claffy and S. McCreary, "Trends in Wide Area IP Traffic Patterns," CAIDA, Technical Report, 2000.

[2] R. L. Carter and M. E. Crovella, "Dynamic server selection using bandwidth probing in wide-area networks," Boston University, Computer Science Department, Technical Report TR-96-007, 1996.

[3] G. Jin, G. Yang, B. R. Crowley, and D. A. Agarwal, "Network Characterization Service (NCS)," in *Proceedings of 10th IEEE International Symposium on High Performance Distributed Computing*, 2001, pp. 289-299.

[4] J. Strauss, D. Katabi, and F. Kaashoek, "A Measurement Study of Available Bandwidth Estimation Tools," in *Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement*, 2003, pp. 39-44.

[5] M. Kazantzidis, D. Maggiorini, and M. Gerla, "Network Independent Available Bandwidth Sampling and Measurement," in *Proceedings of Second International Workshop on Quality of Service in Multiservice IP Networks*, 2003, pp. 117-130.

[6] G. W. Xuan and Y. S. Zheng, "Prioritized Tri-Packets Probes for Available Bandwidth Measurement," in *Proceedings of International Conference on Communications, Circuits and Systems*, vol. 3, 2006, pp. 1777-1781.

[7] B. Melander, M. Bjorkman, and P. Gunningberg, "A New End-to-End Probing and Analysis Method for Estimating Bandwidth Bottlenecks," in *IEEE Global Telecommunications Conference, GLOBECOM '00*, 2000, pp. 415-420.

[8] B. Melander, M. Bjorkman, and P. Gunningberg, "Regression-Based Available Bandwidth Measurements," in *Proceedings of the 2002 International Symposium on Performance Evaluation of Computer and Telecommunications*, 2002.

[9] M. Adachi, S. Kikuchi, and T. Katsuyama, "NEPRI: Available Bandwidth Measurement in IP Networks," in *IEEE International Conference on Communications*, vol. 1, 2000, pp. 511-515.

[10] J. He, C. E. Chow, J. Yang, and T. Chujo, "An Algorithm for Available Bandwidth Measurement," in *In Proceedings of the First International Conference on Networking-Part 1*, 2001, pp. 753-761.

[11] M. Jain and C. Dovrolis, "Pathload: A measurement tool for end-to-end available bandwidth," in *Proceedings of Passive and Active Measurements (PAM) Workshop*, 2002.

[12] M. Jain and C. Dovrolis, "End-to-end available bandwidth: measurement methodology, dynamics, and relation with TCP throughput," *IEEE/ACM Transactions on Networking (TON)*, vol. 11, no. 4, pp. 537-549, 2003.

[13] V. J. Ribeiro, R. H. Riedi, R. G. Baraniuk, J. Navratil, and L. Cottrel, "pathChirp: Efficient Available Bandwidth Estimation for Network Paths," in *Proceedings of Passive and Active Measurement Workshop*, 2003.

[14] D. Kiwior, J. Kingston, and S. A., "PATHMON, A Methodology for Determining Available Bandwidth over an Unknown Network," in *IEEE/Sarnoff Symposium on Advances in Wired and Wireless Communication*, 2004, pp. 27-30.

[15] D. Zhang, Y. Wu, and J. Xu, "Pathtrait: A Tool for Tight Link Location and End-to-End Available Bandwidth Measurement," in *Proceedings of International Symposium on Parallel and Distributed Processing and Applications*, 2005, pp. 78-89.

[16] L. Xin, "PoissonProb: A new rate-based available bandwidth measurement algorithm," M.Sc. Thesis, University of Windsor, 2005.

[17] G. Kola and M. K. Vernon, "QuickProbe: available bandwidth estimation in two roundtrips," in *ACM SIGMETRICS Performance Evaluation Review*, 2006, pp. 359-360.

[18] Y. Xiao, S. Chen, X. Li, and Y. Li, "A New Available Bandwidth Measurement Method Based on Self-Loading Periodic Streams," in *International Conference on Wireless Communications, Networking and Mobile Computing (WiCom 2007)*, Shanghai, 2007, pp. 1904-1907.

[19] S. Suthaharan and S. Kumar, "Measuring Available Bandwidth: pathChirp's Chirp Train Structure Remodeled," in *Australasian Telecommunication Networks and Applications Conference, 2008. ATNAC 2008.*, Adelaide, SA, 2008, pp. 379-384.

[20] V. Ribeiro, *et al.*, "Multifractal Cross-Traffic Estimation," in *Proceedings of the ITC Specialist Seminar on IP Traffic Measurement, Modeling and Management.*, 2000.

[21] N. Hu and P. Steenkiste, "Evaluation and Characterization of Available Bandwidth Probing Techniques," *IEEE Journal on Selected Areas in Communications*, vol. 21, no. 6, pp. 879-894, 2003.

[22] S. R. Kang, X. Liu, M. Dai, and D. Loguinov, "Packet-Pair Bandwidth Estimation: Stochastic Analysis of a Single Congested Node," in *Proceedings of the 12th IEEE International Conference on Network Protocols, ICNP*, 2004, pp. 316-325.

[23] A. Bhati, "Envelope: Estimation of Bottleneck and Available Bandwidth over Multiple Congested Links," MSc Thesis, Texas A&M University, 2004.

[24] L. Min, S. Jinglin, L. Zhongcheng, K. Zhigang, and M. Jian, "A new end-to-end measurement method for estimating available bandwidth," in *Proceedings of the Eighth IEEE International Symposium on Computers and Communication (ISCC'03)*, vol. 2, 2003, pp. 1393-1400.

[25] A. Zhou, *et al.*, "A New Method for End-to-End Available Bandwidth Estimation," in *IEEE Global Telecommunications Conference, 2008. IEEE GLOBECOM 2008*, 2008, pp. 1-5.

[26] A. Botta, Dapos, S. Antonio, A. Pescape, and G. Ventret, "BET: A Hybrid Bandwidth Estimation Tool," in *Porceedings of the 11th International Conference on Parallel and Distributed Systems (ICPADS'05)*, vol. 2, 2005, pp. 520-524.

[27] M. Zhang, C. Luo, and J. Li, "Estimating Available Bandwidth Using Multiple Overloading Streams," in *IEEE International Conference on Communications, 2006. ICC '06.*, vol. 2, Istanbul, 2006, pp. 495-502.

[28] S. Ekelin, *et al.*, "Real-Time Measurement of End-to-End Available Bandwidth using Kalman Filtering," in *Proc. 10th IEEE/IFIP Network Operations and Management Symposium*, 2006.

[29] A. Cabellos-Aparicio, F. J. Garcia, and J. Domingo-Pascual, "A Novel Available Bandwidth Estimation and Tracking Algorithm," in *IEEE Network Operations and Management Symposium Workshops, 2008. NOMS Workshops*, 2008, pp. 87-94.

[30] C. Dovrolis, P. Ramanathan, and D. Moore, "What do packet dispersion techniques measure?," in *Proceedings of Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies, IEEE INFOCOM 2001*, vol. 2, 2001, pp. 905-914.

[31] J. Cao, W. S. Clevel, D. Lin, and D. X. Sun, "Internet Traffic Tends Toward Poisson and Independent as the Load Increases," in *Nonlinear Estimation and Classification*, New York, 2002, pp. 83-109.

[32] V. J. Ribeiro, R. H. Riedi, M. S. Crouse, and R. G. Baraniuk, "Multiscale queuing analysis of long-range-dependent network traffic," in *Proc. IEEE INFOCOM*, 2000, pp. 26-30.

[33] G. Welch and G. Bishop, "An Introduction to the Kalman Filter," Department of Computer Science, University of North Carolina at Chapel Hill, Chapel Hill, NC, USA, Technical Report TR95-041, 1995.

[34] M. Neginhal, K. Harfoush, and H. Perros, "Measuring Bandwidth Signatures of Network Paths," in *Proceedings of IFIP Networking 2007*, Atlanta, GA, 2007.

[35] A. Botta, A. Dainotti, and A. Pescapè, "Multi-protocol and multi-platform traffic generation and measurement," in *INFOCOM 2007 DEMO Session*, Anchorage (Alaska, USA), May 2007.

[36] R. Prasad, C. Dovrolis, M. Murray, and K. Claffy, "Bandwidth Estimation: Metrics, Measurement Techniques, and Tools," *IEEE Network*, vol. 17, no. 6, pp. 27-35, Nov. 2003.

[37] A. Pasztor and D. Veitch, "The packet size dependence of packet pair like methods," in *Tenth IEEE International Workshop on Quality of Service*, 2002, pp. 204-213.

[38] A. Broido, Y. Hyun, R. Gao, and K. C. Claffy, "Their share: diversity and disparity in IP traffic," in *Proceedings of Passive and Active Network Measurement (PAM), 5th International Workshop*, Antibes Juan-les-Pins, France, 2004.

[39] T. Karagiannis, M. Molle, M. Faloutsos, and A. Broido, "A Nonstationary Poisson View of Internet Traffic," in *Proceedings of IEEE INFOCOM*, 2004, pp. 1558-1569.

[40] H. Zhou, Y. Wang, and Q. Wang, "Measuring Internet bottlenecks: location, capacity, and available bandwidth," in *Proceedings of International Conference on Computer Network and Mobile Computing*, 2005, pp. 1052-1062.

[41] H. Zhou, Y. Wang, X. Wang, and X. Huai, "Difficulties in Estimating Available Bandwidth," in *IEEE International Conference on Communications*, vol. 2, 2006, pp. 704-709.

[42] G. Urvoy-Keller, T. En-Najjary, and A. Sorniotti, "Operational Comparison of Available Bandwidth Estimation Tools," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 1, pp. 39-42, Jan. 2008.

[43] A. Shriram and J. Kaur, "Empirical Evaluation of Techniques for Measuring Available Bandwidth," in *26th IEEE International Conference on Computer Communications. IEEE INFOCOM '07*, Anchorage, AK, 2007, pp. 2162-2170.

[44] V. J. Ribeiro, R. H. Riedi, and B. R. G., "Spatio-temporal available bandwidth estimation with STAB," in *Proceedings of the Joint International Conference on Measurement and Modeling of Computer Systems*, 2004, pp. 394-395.

[45] V. Paxson, "Measurements and Analysis of End-to-End Internet Dynamics," Ph.D. Thesis, University of California, 1997.

[46] X. Liu, K. Ravindran, and D. Loguinov, "A Stochastic Foundation of Available Bandwidth Estimation: Multi-Hop Analysis," *IEEE/ACM Transactions on Networking*, vol. 16, no. 1, pp. 130-143, Feb. 2008.

[47] L. Lao, C. Dovrolis, and M. Y. Sanadidi, "The probe gap model can underestimate the available bandwidth of multihop paths," *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 5, pp. 29-34, 2006.

[48] K. Lai, "Measuring the Bandwidth of Packet Switched Networks," Ph.D. thesis, Department of Computer Scienc, Stanford University, 2002.

[49] K. Lai and M. Baker, "Measuring bandwidth," in *In Proceedings of IEEE INFOCOM '99*, 1999, pp. 21-25.

[50] S. R. Kang, X. Liu, A. Bhati, and D. Loguinov, "On Estimating Tight-Link Bandwidth Characteristics over Multi-Hop Paths," *26th IEEE International Conference on Distributed Computing Systems, ICDCS 2006*, pp. 55-59, 2006.

[51] M. Jain and C. Dovrolis, "Ten fallacies and pitfalls on end-to-end available bandwidth estimation," in *Proceedings of the 4th ACM SIGCOMM Conference on internet Measurement IMC '04*, 2004, pp. 272-277.

[52] M. Jain and C. Dovrolis, "End-to-end estimation of the available bandwidth variation range," *Proceedings of the 2005 ACM SIGMETRICS international conference on Measurement and modeling of computer systems SIGMETRICS '05*, vol. 33, no. 1, pp. 265-276, 2005.

[53] X. Hei, D. H. K. Tsang, and B. Bensaou, "Available Bandwidth Measurement using Poisson Probing on the Internet," in *IEEE International Conference on Performance, Computing, and Communications*, 2004, pp. 207-214.

[54] J. He, "On Available Bandwidth Measurement Implementation and Experiment," in *Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks (LCN'04)*, 2004, p. 772–773.

[55] J. He, "Available bandwidth measurement, implementation and evaluation," in *Proceedings of 12th IEEE International Conference on Networks, ICON 2004*, vol. 1, 2004, pp. 226-230.

[56] A. Botta, A. Pescape, and V. G., "On the Performance of Bandwidth Estimation Tools ," in *Proceedings of the 2005 Systems Communications (ICW'05)*, 2005, pp. 287-292.

[57] V. Paxson and S. Floyd, "Wide-Area Traffic: The Failure of Poisson Modeling," *IEEE/ACM Transactions on Networking*, vol. 3, no. 3, pp. 226-244, Jun. 1995.

[58] W. E. Leland, M. S. Taqq, W. Willinger, and D. V. Wilson, "On the self-similar nature of Ethernet traffic," *IEEE/ACM Transactions on Networking (TON)*, vol. 2, no. 1, pp. 1-15, Feb. 1994.

[59] J. C. Hoe, "Improving the start-up behavior of a congestion control scheme for TCP," in *Proceedings of ACM SIGCOMM*, Stanford, CA, USA, 1996, p. 270–280.

[60] F. Qi, J. Zheng, W. Jia, and G. Wang, "Available Bandwidth Measurement Schemes over Networks," *Lecture Notes in Computer Science*, vol. 3619, pp. 931-940, Aug. 2005.

[61] S. Seshan, M. Stemm, and R. H. Katz, "SPAND: Shared Passive Network Performance discovery," in *Proceedings of 1st Usenix Symposium on Internet Technologies and Systems (USITS'97)*, Monterey, CA, USA, 1997.

APPENDIX A

A.1. Poisson Process and Poisson Traffic

The Poisson distribution is a discrete distribution which describes the number of times that some known event has occurred as a function of time, where events can occur at random times (such as, the number of telephone calls at a business or the number of accidents at an intersection). In network research, it has been widely used to model the packet arrivals and packets queuing time for a system. The probability mass function for the Poisson process is:

$$p(x,\lambda) = \frac{e^{-\lambda}\lambda^{x}}{x!} \quad \text{for } x = 0,1,2,...$$

where, λ denotes the average number of packets that arrives in a given time period. Also referred to as intensity, x is the number of packets we are currently interested in and e is the base of natural logarithmic function *ln*. Under Poisson modeling, network traffic is usually considered as a random arrival process using non-homogeneous Poisson process. The difference is that in non-homogeneous Poisson process, instead of taking a stationary value of intensity, it is considered as a deterministic function of time as $\lambda(t)$. Figure A-1 shows an example of non-homogeneous Poisson Process.

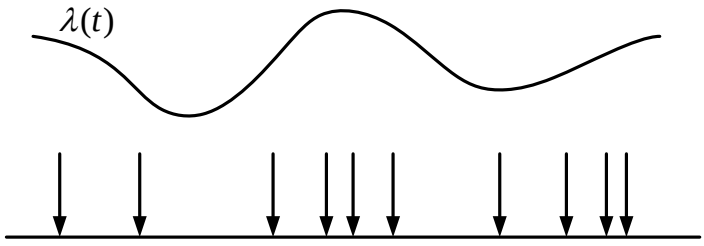

Figure A-1. Non-homogeneous Poisson Process

There are a number of interesting mathematical properties exhibited by Poisson processes. Primarily, superposition of independent Poisson processes results in a new Poisson process, whose rate is the sum of the rates of the independent Poisson processes. Further, the independent increment property makes a Poisson process memoryless. Poisson processes are common in traffic applications scenarios that comprise of a large

number of independent traffic streams. The reason is that, under suitable conditions, a large number of independent multiplexed streams approach a Poisson process as the number of processes grows, but the individual rates decrease in order to keep the aggregate rate constant. Nevertheless, it is to be noted that traffic aggregation need not always result in a Poisson process. The Poisson model is primarily based on two assumptions:

1. The number of sources is infinite.

2. The traffic arrival pattern is random.

The Poisson model was widely applied in network engineering in the early-90's. But studies [40, 41] during that period had shown that the LAN and WAN traffic diverge considerably from the Poisson pattern as the exponential distribution underestimates the burstiness of traffic and can better be modeled by self-similar process because of the long-range dependence.

Within the last decade Internet has grown rapidly in diversity and disparity, and the nature of traffic has changed significantly. The speed of links has increased several orders of magnitude, up to Giga-byte per second order, and each link had much more connectivity. Another important phenomenon that affects the traffic modeling is network multiplexing. A recent study [38] has shown that the network traffic on Internet can again be modeled by Poisson distribution. The reason is that the statistical properties of packet traffic on the internet link dramatically change because of the presence of a large number of simultaneous active connections. The high speed links have the capacity to drain the packets so fast that "*the increasing connection load can bring the traffic to Poisson and independence before substantial upstream queuing occurs; the onset of queuing does not resurrect the long-range dependence*" [31]. Also the burstiness of single network traffic cannot change the nature of traffic of highly multiplexed connections, even though they may still be bursty as an individual connection. Researchers [39] have found that for a heavily loaded link, the packets arrive back-to back and the distribution of arrival times depends on the packet size from the transmitter's point of view. Also from the analysis of large-scale packet dataset, the packet sizes have been found to be independent. Although

the edge links with limited traffic load may show burstiness, self-similarity and long-range dependence characteristics; the very high speed internet backbone links carry a huge amount of traffic which is made up of traffic from a large number of different connections. This makes the traffic on the internet backbone links close to Poisson distribution pattern.

The measurement time scale is another important factor of traffic modeling. It has been found that Internet traffic becomes self-similar and long-range dependent at large time scale, but at the time scale of millisecond or minute level the traffic is usually non-stationary and show completely different properties compared to the average properties of large time scale. Karagiannis *et al.* [39] have shown "*packet arrivals appear Poison at sub-second time scale; Internet traffic is nonstationary at multi-second time scales; Internet traffic exhibits long-range dependence (LRD) at large time-scale*". These findings have immense importance for designing network measurement algorithms to achieve high accuracy. Usually most applications require the bandwidth information at the time scale of millisecond to minute level, where the network traffic follows Poisson pattern. Therefore the available bandwidth measurement algorithms which follow Poisson traffic assumption have higher possibility to provide better estimates.

APPENDIX B

B.1. Internet Control Message Protocol (ICMP)

Internet Control Message Protocol (ICMP) is part of the Internet Protocol Suite as defined in RFC 792. ICMP protocol is used to allow network devices to report errors and other conditions in data transmission. Some of the ICMP's functions are to:

- ***Announce network errors***, such as a host or entire portion of the network being unreachable, due to some type of failure. A TCP or UDP packet directed at a port number with no receiver attached is also reported via ICMP.

- ***Announce network congestion***. When a router begins buffering too many packets, due to an inability to transmit them as fast as they are being received, it will generate ICMP Source Quench messages. Directed at the sender, these messages should cause the rate of packet transmission to be slowed. Of course, generating too many Source Quench messages would cause even more network congestion, so they are used sparingly.

- ***Assist Troubleshooting***. ICMP supports an Echo function, which just sends a packet on a round--trip between two hosts. Ping, a common network management tool, is based on this feature. Ping will transmit a series of packets, measuring average round--trip times and computing loss percentages.

- ***Announce Timeouts***. If an IP packet's TTL field drops to zero, the router discarding the packet will often generate an ICMP packet announcing this fact. TraceRoute is a tool which maps network routes by sending packets with small TTL values and watching the ICMP timeout announcements.

Like TCP and UDP, ICMP uses IP to communicate across network. Internet Protocol encapsulates the appropriate ICMP message with a new IP header (to get the ICMP message back to the original sending host) and transmits the resulting datagram in the usual manner.

Each ICMP message is encapsulated directly within a single IP datagram, and thus, like UDP, ICMP uses connectionless approach, so packet delivery is unreliable.

The IP packets identify the next layer protocol contained in the data section using the ***protocol*** type field. ICMP packets are identified with protocol type value of ***1***. The following figure shows how ICMP packet fields are placed in an IP packet:

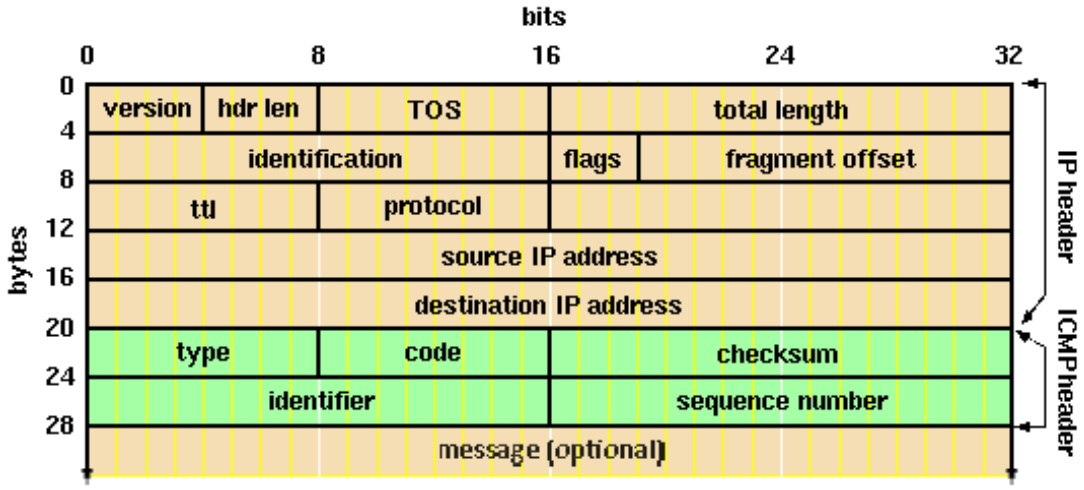

Figure B-1. ICMP header with IP header

- **Type and Code fields**

There are different types of messages that ICMP packet can carry. These different messages are grouped into types. The 1-byte type field is used to specify the type of message that is enclosed in the packet. Some of the types are further divided into sub-types. The next 1-byte code field is used to specify the sub-type. Table B-1 shows some of the types and some of the codes used in ICMP packets.

- **Checksum:**

The 2-byte checksum is used to ensure that the packet has arrived without corruption. The checksum is computed based on the ICMP portion of the packet, using a specific algorithm defined in RFC792.

- **Identifier and a Sequence number**

These two fields are used to uniquely identify an ICMP message.

- **Message:**

The message part is a variable size component that represents the message being sent. The message part contains various other fields that are unique to individual ICMP message types.

Table A-1. Some types and codes used in ICMP header

Type	Code	Description
0	0	for echo reply message (also see Type 8)
3	0	net unreachable
	1	host unreachable
	2	protocol unreachable
	3	port unreachable
	4	fragmentation needed and DF set
	5	source route failed
	6	destination network unknown
	7	destination host unknown
	8	source host isolated
	9	communication with destination network administratively prohibited
	10	communication with destination host administratively prohibited
	11	network unreachable for type of service
	12	host unreachable for type of service
4	0	source quench message
5	0	Redirect datagrams for the Network
8	0	for echo request message (see Type 0)
11	0	time to live exceeded in transit
12	0	pointer indicates the error (identifies the octet where an error was detected.)
13	0	for timestamp message
14	0	for timestamp reply message
15	0	for information request message
16	0	for information reply message

B.2. Use of ICMP packet in PathAB

The stand-alone mode of PathAB relies on the ICMP protocol for the estimation process. In the initial probing phase each probe packet of the exponential train is followed back-to-back by an ICMP echo request packet (type 8). The algorithm calculates the rough available bandwidth after receiving back the echo response packets. In the direct probing

phase the first and the last packet of each probing train is followed by ICMP echo request packets and the algorithm calculates available bandwidth after receiving all the response packets.

To separate the ICMP packets generated by PathAB from other ICMP packets the process id of PathAB program is used as the identifier field of all echo request packets. The sequence number field is used to send the train number and packet number of each echo request packet. The first 8 bits of sequence number field are used to send train number and the following 8 bits are used for sending packet number. The ICMP echo request are sent without any message body, hence the size of each ICMP packet used in PathAB is 28 bytes.

APPENDIX C

C.1. E-mail communication with the authors of MoSeab

From: Chong Luo <Chong.Luo@microsoft.com>
To: Roy Debashis <roy17@uwindsor.ca>,
"ben_zhang@zju.edu.cn" <ben_zhang@zju.edu.cn>,
Jiang Li <jiangli@microsoft.com>
Date : Wed, Jun 4, 2008 at 1:45 AM
Subject: RE: Request for MoSeab program

Dear Debashis,

Thanks for your interest in MoSeab. However, I regret to tell you that we cannot give you the code. This work is done in Microsoft Research Asia. As a corporate research lab, we need to follow the company regulations. Sorry for that.

Thanks,

Chong

From: Roy Debashis [mailto:roy17@uwindsor.ca]
Sent: 2008年6月4日 11:27
To: ben_zhang@zju.edu.cn; Chong Luo; Jiang Li
Subject: Request for MoSeab program

Dear Sir/Ma'm,

I am a Masters' student at University of Windsor, Canada and I am doing my research in the area of available bandwidth estimation of network path under supervision Dr. A.K. Aggarwal. Recently I have gone through your paper "*Estimating Available Bandwidth Using Multiple Overloading Streams*" in which you have introduced a new method **MoSeab** to estimate the available bandwidth. I will be very thankful if you could provide me the programs for MoSeab (if possible both NS2 simulation program and the actual implementation). It will be very much helpful towards my research.
Looking forward for your response.

With due regards,

Debashis Roy

High Performance Grid Computing Research Group
School of Computer Science
University of Windsor, ON, Canada
Phone: (519)253-3000 ext. 4406

VITA AUCTORIS

NAME	:	Debashis Roy
PLACE OF BIRTH	:	West Bengal, India
YEAR OF BIRTH	:	1983
EDUCATION	:	Bachelor of Engineering in Computer Science & Technology, 2005. Bengal Engineering and Science University, Shibpur, WB, India. Master of Science, 2009. School of Computer Science, University of Windsor, ON, Canada
WORK EXPERIENCE	:	Graduate Research and Teaching Assistant, University of Windsor, January 2007 − April 2009. ESB Java Developer, IBM Canada Limited, September 2008 − December 2008. Systems Engineer, Siemens Information Systems Ltd., India, July 2005 − December 2006.
AWARDS	:	International Graduate Student Scholarship, 2007-2009. University of Windsor, ON, Canada.
PUBLICATION	:	J.Lu, Y. Yu, D. Roy and D. Saha, "Web service composition: a reality check", in *Eighth International Conference on Web Information Systems Engineering*, Nancy, 2007, pp. 523-532.