

Towards unified self-congestion probing for bandwidth measurement

Xiaojun Hei[†], Shan Chen[‡], Brahim Bensaou[‡] and Danny H.K. Tsang[†]

[†]Department of Electronic and Computer Engineering

[‡]Department of Computer Science and Engineering

Hong Kong University of Science and Technology

Clear Water Bay, Kowloon, Hong Kong

heixj@ece.ust.hk, chenshan@cse.ust.hk, brahim@cse.ust.hk and eetsang@ece.ust.hk

Abstract

The self-congestion probing, which estimates bandwidth by controlling a temporal congestion of a probing stream, is the most popular approach in bandwidth measurement. Self-congestion tools are easy to implement, fast to converge and are robust to network dynamics with reasonably good accuracy; however, the current tools only exploit parts of the congestion signals, though the probing stream experiences a rich spectrum of congestion signals. TCP protocols follow the same self-congestion principle on inferring available bandwidth. We propose a unified self-congestion probing framework by bridging the self-congestion probing for available bandwidth measurement and TCP congestion control. The recent progress of TCP congestion control in both theory and experimentation provides new avenues in improving the current self-congestion tools. Based on this unified framework, we design and evaluate a simple available bandwidth probing scheme to utilize the Explicit Congestion Notification signal, namely, ECNProbe. We conduct a measurement study on a Linux-based testbed and evaluate the performance of several available bandwidth measurement tools. We demonstrate that the proposed ECNProbe significantly improves the measurement accuracy with small convergence time and low overhead probing.

I. INTRODUCTION

P2P applications have become the largest contributor of the Internet traffic in the past years. These applications often run over the congestion-responsive TCP transport or even the congestion-unresponsive UDP transport for delivery. Due to these huge traffic volumes, other Internet applications often suffer

from service degradation. ISPs have been deploying traffic throttling techniques to limit the rate of P2P traffic [1]. To release the tension between ISPs and P2P applications, we argue that P2P applications should only utilize the available bandwidth conservatively, instead of eating up the network capacity aggressively, when they dimension their overlays and conduct overlay routing. Although there is no apparent incentive for P2P applications to restrain themselves from only acquiring available bandwidth, we are proposing a new design philosophy in which P2P applications understand the traffic stress of ISPs and help to minimize their impact on the network at the application layer. To this end, each peer should be able to probe for bandwidth of overlay links. In addition, P2P traffic should respond more quickly to network congestion than normal TCP transports. In this way, ISPs would embrace P2P applications because these applications provide strong incentives for Internet users to adopt the latest broadband access technologies to enjoy multimedia applications while other Internet applications can still maintain a satisfactory service performance.

A number of end-to-end available bandwidth estimation schemes have been proposed in the past decade. These schemes can be classified based on different criteria [2]. Based on the probing load, we classify these schemes into two categories: queueing-model based [3], [4] and self-congestion based [5], [6]. In both approaches, a probing packet stream is actively sent out into the network, and this probing stream interacts with cross traffic at each hop along the end-end path. The available bandwidth is estimated based on the measurement of the performance metrics of the probing stream. Nevertheless, the measurement tools are operated at dramatically different traffic loads in these two approaches.

In the queueing-model based approach, the probing stream should control its probing rate to avoid congestion along the end-to-end path; therefore, the performance metrics of the probing stream, i.e, loss and delay, can be characterized by modeling the corresponding loss process or the delay process. The available bandwidth is one of the parameters of the queueing models. By inverting the loss model or the delay model, available bandwidth can be estimated. The model-based approach usually leads to truly non-intrusive available bandwidth estimation techniques. However, an accurate characterization of the loss/delay process of the probing stream often requires some prior assumptions about the end-to-end path, including the arrival process of cross traffic. When these assumptions do not meet the real network environment, measurement errors may occur.

In the self-congestion based approach, the probing stream has to temporarily congest the end-to-end path together with cross traffic. The available bandwidth is estimated by determining an appropriate

congestion turning point based on the congestion signals experienced by the probing stream. At this temporal congestion state, packets are accumulated in the bottleneck queue. The aggregate traffic can be treated as fluid-flow and the stochastic behaviors of cross traffic no longer exhibit significant impact on the accuracy of the estimation of the available bandwidth. Those self-congestion based tools are usually simple to develop, fast to converge and are robust to network dynamics with sufficient accuracy. The major challenge is how to control the temporal congestion introduced by the probing stream in such a way that the interference is minimized upon TCP flows along the path [7]. Note that at the congestion state, the probing stream experiences a rich spectrum of congestion signals, i.e., increased delay, decreased throughput, incurring packet loss, etc.; however, the current self-congestion tools only exploit parts of these congestion signals to determine the onset of the congestion. There has been significant progress in TCP congestion control recently in both theory and experimentation. A large body of TCP protocols follow the same self-congestion principle in inferring available bandwidth. The rich research results on TCP provide new avenues in improving the current self-congestion tools.

In this paper, we propose a unified self-congestion probing framework for bandwidth measurement. This framework is able to summarize a large body of self-congestion based available bandwidth measurement tools. We reveal the relationship and difference between two closely-related bandwidth metrics, available bandwidth versus TCP fair share rate. We claim that this generalized probing framework provides insights for developing new available bandwidth measurement tools with better accuracy, fast convergence speed and less probing overhead. We conduct a case study on estimating available bandwidth measurement by determining the onset of the congestion appropriately at the bottleneck link based on the Explicit Congestion Notification (ECN) signal [8] provided by routers. We design and evaluate a simple available bandwidth probing scheme to utilize this ECN signal, namely, ECNProbe. We construct a testbed with a single bottleneck to evaluate four popular available bandwidth measurement tools, IGI, Pathload, pathChirp, Spruce, DietTOPP and ECNProbe. Our evaluation results indicate that ECNProbe achieves accurate estimation with small probing time and low probing overhead.

The rest of the paper is organized as follows. We present a survey on the state-of-the-art progress in available bandwidth measurement and highlight the contribution of this paper in Section II. In Section III we present the unified self-congestion probing framework and propose a simple probing scheme, ECNProbe. In Section IV, we evaluate and compare IGI, Pathload, pathChirp, Spruce, DietTOPP and ECNProbe. Finally, the concluding remarks are made in Section V.

II. STATE-OF-THE-ART AVAILABLE BANDWIDTH MEASUREMENT

There are a few related bandwidth metrics, i.e., capacity, available bandwidth and TCP fair share rate, etc.. A good review can be found in [2]. Capacity is the transmission bit rate of a link. Available bandwidth is the maximum throughput that a link can provide at a certain time period for given cross traffic. TCP fair share rate is the received bit rate of a flow when this flow competes bandwidth “fairly” with other TCP flows on the same link.

Due to a wide range of applications, available bandwidth estimation has become an active research area in the past decade. We do not intend to present a complete list of the available measurement tools here; instead, we provide a summary of major research efforts in the area. We classify these schemes into two categories: queueing-model based and self-congestion based. When a probing stream runs in a self-congestion mode, we may also construct a model to capture the behavior of the system. The major difference between the above two approaches is due to the aggregate traffic intensity (ρ) at the bottleneck link. When $\rho < 1$, we are able to construct a queueing model to capture the interaction between the probing stream and cross traffic. Nevertheless, the self-congestion tools create temporal congestion such that $\rho \geq 1$.

In [9], Sharma and Mazumdar estimated the traffic intensity of cross traffic by studying the delay process of the Poisson probing stream using $M_1 + M_2/GI/1$ and $G + M/GI/1$. Nevertheless, the proposed schemes require that the network nodes are cooperative and provide local queueing information, such as the mean queue length and the waiting time of the probing stream. Alouf et al., proposed to simultaneously estimate the cross traffic intensity and the buffer size of the bottleneck link by investigating the loss process of the probing stream using $M_1 + M_2/M/1/K$ and $M_1 + M_2/D/1/K$ in [3]. Following the same approach in [10], Salamatian et al., inferred the cross traffic intensity based on the loss process analysis of $MMPP/M/1/N$. In [4], Hei et al., demonstrated that packet loss is not a good measurable metric for available bandwidth measurement. In addition, packets may be dropped due to active queue management (AQM) instead of buffer overflow. They inferred the cross traffic intensity by analyzing the delay process of the probing stream using $GI_1 + GI_2/GI_i/1$. Two special cases were studied: $M_1 + M_2/GI_i/1$ for Poisson probing and Poisson cross traffic, and $D + M/GI_i/1$ for Periodic probing and Poisson cross traffic. Note that in the above schemes, available bandwidth is not directly measured; instead, cross traffic is estimated. With the assumptions that there is only single bottleneck link along the end-to-end path and the bottleneck capacity is known, available bandwidth is the residual capacity between the link capacity and cross traffic.

In the self-congestion approach, Pathload [5] and Initial Increasing Gap (IGI)/Packet Transmission Rate (PTR) [6] are two of the earliest tools. By trying different probing rates using a binary search, a reasonable available bandwidth estimate can be found in Pathload by determining a delay increasing trend [5]. In [6] Hu and Steenkiste showed that a turning point exists at which the average input delay gap of the probing stream equals the average output delay gap. At this turning point, IGI utilizes a fluid-flow based formula to estimate the available bandwidth and PTR estimates the available bandwidth on the bottleneck link as the average rate of the packet train. IGI and PTR work at the turning point of the delay gap when the path is self-congested by the additional probing load. In [11], TOPP shared the same self-congestion principle as Pathload and IGI/PTR. The difference is that TOPP utilized the throughput measurement to identify a throughput transition point when $\rho = 1$. In [12], Strauss et al., improved IGI/PTR by setting the inter-gap time between two probe pairs to be exponentially distributed. Their measurement tool, Spruce, shows better accuracy than IGI. In [13], Ribeiro et al., proposed to create a self-induced congestion using an active probing stream with an exponential flight pattern, called “pathChirp”. PathChirp uses the delay of its probing packets to compute probing rate, and a turning point is identified when the probe sending rates and receiving rates start to match. A number of consecutive research have been conducted to improve the self-congestion probing from various aspects. In [14], Liu et al., took a sample-path argument to construct a generalized probing-response-curve to understand the packet-pair/packet-train output dispersion at the congestion turning point.

In this paper, we take the congestion-control engineering approach in understanding the self-congestion probing for available bandwidth measurement. TCP congestion control has been heavily investigated in the literature. Many TCP flavors have been proposed and used by Internet applications. Available bandwidth estimation is an indispensable component in TCP. Most TCP flavors are equipped with the available bandwidth estimation component implicitly. A few TCP flavors utilize the available bandwidth measurement explicitly, such as TCP Westwood [15]. Nevertheless, TCP probing can also be treated as a self-congestion probing approach. In the proposed unified self-congestion probing framework, we illustrate the relationship and difference between available bandwidth and TCP fair share rate, which is the targeted bandwidth share achieved by various TCP protocols [16].

Researchers have been conducting extensive evaluation and comparison studies over the existing available bandwidth measurement tools through simulations, controlled laboratory network and real network experiments [17]–[19]. We also conduct a measurement study on a Linux-based testbed and evaluate

the performance of a few popular self-congestion tools. Within the above unified self-congestion probing framework, we intend to illustrate insight into how these self-congestion probing schemes react to different congestion signals. These understandings may be helpful in developing new self-congestion based available bandwidth measurement tools with better accuracy, fast convergence speed and less probing overhead.

III. A UNIFIED SELF-CONGESTION BASED PROBING FRAMEWORK

A. Motivation

In the end-to-end active probing for available bandwidth measurement, a probing packet stream is sent out from the sending node to the receiving node. This packet stream consists of multiple packet pairs or packet trains. The probing stream interacts with cross traffic on each hop along its end-to-end path; the available bandwidth is estimated based on such interactions. In self-congestion based probing, this interaction effect is often maximized at the temporal congestion and can be easily measured at the end points, such as increased queueing delay, increased delay variation, decreased throughput, incurred packet loss, etc.. If this temporal congestion is too aggressive, those responsive TCP flows at the bottleneck detect the congestion and back-off the transmission rate; then, the eventual available bandwidth may be over-estimated due to the back-offs of those TCP flows. The challenge in self-congestion probing for available bandwidth measurement is to identify an appropriate congestion turning point and adjust the probing rate quickly to push the aggregate system load to this congestion point for a temporal period of time.

Conventional TCP flows essentially employ a self-congestion principle to share network bandwidth with other TCP flows. A TCP flow controls its congestion window and limit its sending rate at which it sends traffic into its connection as a function of perceived network congestion [20]. The fundamental difficulty in TCP probing is the complication between its efficiency control and fairness control [21]. TCP flows aim to achieve a high throughput by configuring their congestion window size following the delay-bandwidth product. At the same time, TCP flows are required to compete fairly for bandwidth with other TCP flows. This requirement constrains the TCP flows to probing the available bandwidth conservatively; hence, the convergence time may be quite long. Unlike TCP probing, self-congestion probing for available bandwidth relaxes the fairness requirement, as long as the probing traffic does not visibly impact the TCP throughput.

To detect congestion along the path, two major approaches have been taken in designing TCP protocols: loss-based versus delay-based. The loss-based TCP protocols, i.e., TCP Reno [22] and NewReno [23],

only take packet losses as the indication of congestion. If a Reno/NewReno sender perceives no packet losses on the path between itself and the destination, the sending rate is increased continuously to probe the available bandwidth; if the sender perceives that there are packet losses, the sender backs off the sending rate by half. Thus the sending rate fluctuates around an equilibrium congestion point. Note that the additive-increase multiplicative-decrease (AIMD) algorithm behind TCP congestion control for adjusting the congestion window size ensures that the bottleneck link bandwidth is fully utilized while certain fairness among competing flows is achieved in the long run. The bandwidth probing behaviors in loss-based TCP is intrusive, as these flows attempt to grab their fair share rate by pushing other flows back even though the bottleneck capacity has already been fully occupied.

The other category of TCP protocols deploy delay-based congestion detection. The increasing packet transmission delay is often served as an indication of incipient congestion and these TCP senders respond correspondingly to relieve congestion before congestion losses occur. TCP Vegas [24] is one of the representative delay-based protocols with impact. TCP Vegas computes the difference between the *actual input rate* ($cwnd/RTT$) and the *expected rate* ($cwnd/RTT_{min}$), where RTT is the estimation of the round trip time and RTT_{min} is the minimum round trip time measured, to infer network congestion. This difference reflects the number of packets queued in the buffers along the connection path [25]. TCP Vegas attempts to keep the number of queued packets small and constant to achieve efficiency and avoid severe congestion at the intermediate routers. In responding to early congestion signals, TCP Vegas is less intrusive than loss-based TCP. Nevertheless, TCP Vegas is not able to achieve its own TCP fair share rate when competing with loss-based TCP flows.

The basic difference between the loss-based and delay-based TCP protocols is their equilibrium congestion point. Delay-based TCP aims to evolve around the small buffer point at the bottleneck, while loss-based TCP attempts to drive the bottleneck buffer overflow to have the maximum possible buffer occupancy for each round. Nevertheless, the self-congestion probing for available bandwidth is to detect the congestion point when the bottleneck capacity is fully utilized and the queue is about to build up. From this perspective, we can have a unified view of the self-congestion probing for available bandwidth and TCP probing for fair share rate.

B. Overview

As shown in Fig. 1, we outline the structure of this unified probing framework for uni-direction end-to-end bandwidth measurement. The sender sends out a probing stream; this stream traverses the end-to-end

path and competes bandwidth with other flows along the path. Probing packets experience a rich spectrum of congestion signals $p(t)$; these signals can be offered by the network or be estimated from end-hosts. These probing packets carry congestion signals and finally arrive at the receiver; thereafter, the receiver estimates the end-to-end bandwidth based on these congestion signals. The aggregate probing traffic is controlled by a probing window of $W(t)$ packets and the probing traffic intensity is controlled by packet spacing to achieve the probing rate $x(t)$.

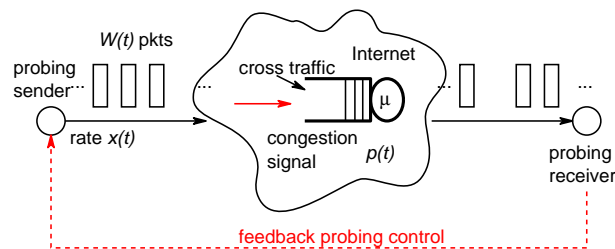


Fig. 1. A Unified Self-congestion based Probing Framework

The probing process is conducted in a round-by-round fashion. At the beginning of the probing process, the sender has no knowledge about the bandwidth and starts with an initial probing window $W(0)$ and an initial probing rate $x(0)$. The first blind probing is unlikely to create a targeted congestion turning point and hence lead to a poor bandwidth estimate at the receiver. Based on the congestion signals $p(0)$ carried by the initial probing stream, the receiver controls the probing behaviors ($W(t)$ and $x(t)$) at the sender so that the probing stream experiences the targeted congestion turning point. With a different setting of this targeted turning point, the probing stream estimates different bandwidth metrics, i.e., available bandwidth versus TCP fair share rate. Note that to control the intrusiveness of the measurement stream, the probing window should be controlled carefully. Since the probing rate $x(t)$ eventually converges to the targeted bandwidth metric, the measurement accuracy essentially depends on the accuracy of the probing rate control. For a reference in Table I, we summarize the congestion signal selection and the turning point detection algorithms used in some popular self-congestion available bandwidth measurement tools and TCP protocols. A promising avenue to improve the self-congestion tools is to design the probing tool by utilizing the rich spectrum of congestion signals and deploying hybrid window control and rate control to achieve accurate probing for bandwidth measurement with fast convergence and small overhead.

TABLE I
CONGESTION SIGNAL SELECTION AND TURNING POINT DETECTION ALGORITHMS

Tool	Congestion signal	Turning point detection
IGI [6]	Delay gap	Binary/linear search
PTR [6]	Throughput	Binary/linear search
Pathload [5]	Delay trend	Binary search
pathChirp [13]	Delay gap	Average
TOPP [11]	Throughput	Regression
Spruce [12]	Delay gap	Algorithmic computation
TCP Reno [22]	Loss	AIMD iteration
TCP new Reno [23]	Loss	AIMD iteration
TCP Vegas [24]	Delay	AIAD iteration

C. ECNProbing

TCP congestion control has been investigated extensively. S. Floyd proposed to add Explicit Congestion Notification (ECN) [8] in Internet routers to improve the performance of TCP flows. Although the ECN usage on routers is still limited [26], most Internet routers are equipped with this ECN feature and deployment attempts are on going. In this section, we design a simple self-congestion probing scheme to exploit ECN to determine the congestion turning point. In Section IV, in a measurement study we investigate whether the proposed ECNProbe is able to determine an appropriate congestion turning point and estimate available bandwidth accordingly.

The current ECNProbe implementation is coded using C in Linux. In ECNProbe, the sender has two communication channels with the receiver: probing channel and control channel. A probing sequence consists of multiple packet trains. Each packet train consists of W probing packets. In the current version of ECNProbe, W is a constant and the probing packet size L is also a constant. The probing packets are injected into the network using the raw socket. The control channel is used by the receiver to feedback the probing control messages; the probing rate $x(t)$ is adjusted at the sender. This control connection is implemented using the TCP socket to ensure a reliable delivery of control signaling messages. The probing process of ECNProbe is conducted on a round-by-round fashion. In each round, the sender injects W probing packets at the rate $x(t)$ and then wait for the control message from the receiver. Only after the sender receives a control message from the receiver, the next packet train is sent out. The receiver receives the probing packets, examines the number of ECN-marked packets and then computes the ECN marking ratio r_{ECN} . The congestion turning point is determined by a pre-specified ECN marking ratio γ . Similar to Pathload, the receiver runs a binary search algorithm to control $x(t)$. When $r_{ECN} > \gamma$, congestion occurs and $x(t)$ reduces; when $r_{ECN} < \gamma$, congestion does not occur and $x(t)$ increases. When $|r_{ECN} - \gamma| < \epsilon$,

the binary search terminates and the final probing rate $x(t)$ is the estimated available bandwidth.

IV. PERFORMANCE EVALUATION

In this section, we present a measurement study on the performance of five popular available bandwidth measurement tools and the proposed ECNProbe on a Linux-based testbed with a single-bottleneck. Table II summarizes the measurement tools evaluated in the experiments. Five software tools are used to generate the cross traffic. The parameters of the tools use the recommended values for the tools if not explicitly specified. These default parameters are tabulated in Appendix A.

TABLE II
SOFTWARE SUMMARY FOR THE AVAILABLE BANDWIDTH MEASUREMENT TOOLS

Cross traffic generator	Iperf v1.70 [27] Poisson Traffic Generator v1.2 [28] D-ITG v2.4 [29] Traffic Generator (TG) v2.0 [30] The Multi-Generator Toolset (MGEN) v4.0 [31]
Available bandwidth measurement tools	Pathload v1.2 [5] IGI root version v1.0 [6] Spruce v0.2 [12] pathChirp v2.3.7 [13] DietTOPP v0.1 [11], [32]

To characterize the performance of each available bandwidth tool, we use the following three metrics for comparison:

- **Accuracy:** each measurement run of one tool should yield a good estimate of the available bandwidth. We plot each measurement result against the ideal available bandwidth value in a time diagram. In the controlled environment, we can exactly control the cross traffic intensity.
- **Convergence:** a timely measurement is important for available bandwidth results to be useful for P2P applications. We quantify the convergence time of a tool as the total time taken by this tool to return an estimate.
- **Overhead:** an active probing stream injects additional traffic into the network. For a large-scale deployment of a tool, it is important that this tool generates low amount of probing traffic. The probing overhead can be quantified as the total amount of probing traffic and the average probing bit rate used by the tool to provide an estimate.

In summary, the design goal of an available bandwidth tool is to achieve accurate estimation with small convergence time and low overhead probing.

A. Testbed Configuration

As shown in Fig. 2, the testbed has a dumb-bell topology and consists of 5 computers. One computer serves as a Linux router and enables the features of Random Early Detection (RED) and Early Congestion Notification (ECN). This router is equipped with two network interfaces, located in two network domains, 192.168.0.* and 192.168.1.*. The top two computers are used as the cross traffic sender and the receiver, respectively. The bottom two computers serve as the probing sender and the receiver. The cross traffic sender, the probing sender and the left interface of the router are connected via a 100Mbps Ethernet switch. The cross traffic receiver, the probing receiver and the right interface of the router are connected via a 10Mbps Ethernet Hub. In Fig. 2, the cross traffic and the probing traffic traverse the switch, the router, the hub and finally arrive at their respective receiver. We can observe that this testbed has one bottleneck link at the 10Mbps hub. The cross traffic stream and the probing stream compete for the 10Mbps bottleneck capacity.

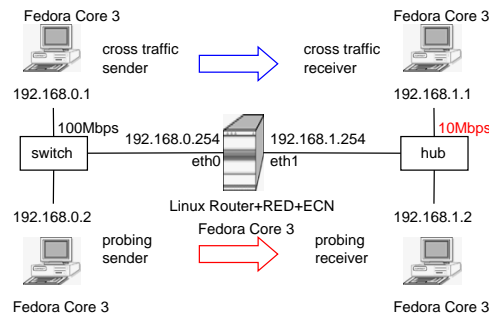


Fig. 2. The testbed with a single-bottleneck

We conduct repeated experiments for each tool. After the cross traffic are injected into the network, we start a tool for probing available bandwidth. After the tool returns an estimate of the available bandwidth, this result is also recorded for later analysis. Then we terminate the probing process. The starting time and the ending time of the tool are recorded to compute the convergence time of the tool. We restart the probing process from the fresh state. As the two types of cross traffic in this measurement study are unresponsive UDP traffic, the immediate restart of the tool does not perturb the network conditions. Note that if the cross traffic consist of elastic TCP traffic, the duration between two measurement runs should be sufficiently long to ensure that the network conditions return to the original steady state.

B. Performance of Cross Traffic Generation

An accurate generation of cross traffic is important in our measurement study. We launched two types of cross traffic, constant-bit-rate (CBR) and Poisson. In our experiments, we examine the packet arrival process of the cross traffic generators. The packet size of cross traffic is of 500 bytes. Figure 3 depicts the complementary cumulative distribution function (CCDF) of the packet inter-arrival time of Iperf and D-ITG. For CBR traffic, the ideal CCDF curve should be a vertical line. For example, when the bit rate is 2Mbps, the interval packet arrival time is $\frac{500 \times 8}{2 \times 10^6} = 0.002$ second. When the cross traffic is 2Mbps, the D-ITG failed to generate a constant bit rate. In all the five cases of cross traffic load, Iperf generates good CBR packet streams.

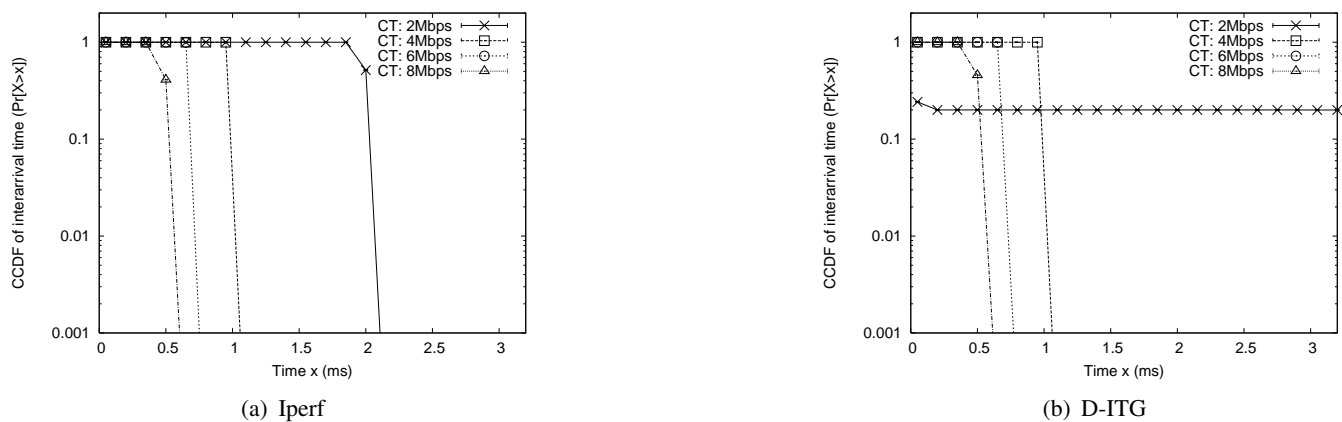


Fig. 3. Performance of the CBR cross traffic generators

Figure 4 depicts the CCDF of the packet inter-arrival time of Poisson traffic generator, TG, D-ITG and MGEN. The ideal CCDF curves of the inter-arrival time of the Poisson traffic should linear curves rooted at point (0, 1). We find that only the Poisson traffic generator is able to generate a good Poisson traffic stream as shown in Figure 4(a). In the remaining experiments, we will use Iperf to generate the CBR cross traffic and use the Poisson traffic generator to generate the Poisson traffic.

C. Performance of Probing Sequence Generation

End-to-end available bandwidth measurement tools send out probing packet sequences following specific patterns. An accurate implementation of these probing sequences are prerequisite in estimating available bandwidth accurately. In this section, we examine the performance of the probing sequences generation in different tools. During the measurement sessions, we capture the traffic using tcpdump [33] and filter out the probing sequences order to study the probing process of different tools.

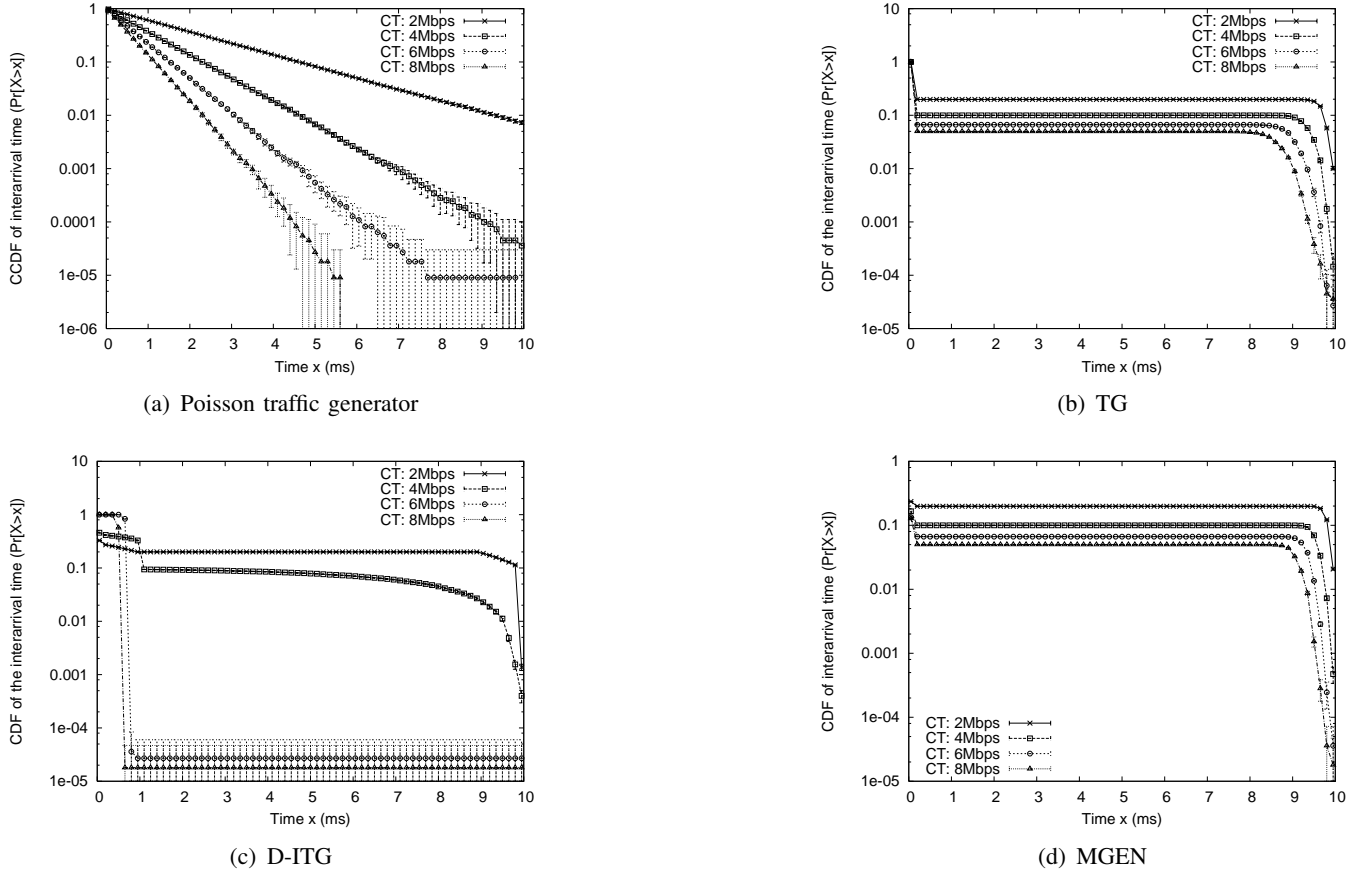


Fig. 4. Performance of the Poisson cross traffic generators

These probing sequences can be classified into two categories: gap-based and rate-based. In gap-based probing sequences, the inter-arrival time between probing packets is controlled carefully following some criteria. In IGI, the gap between two probing packets in the same packet train is maintained as a constant; in pathChirp, the packet gaps in the same packet chirp are decreased exponentially. As shown in Fig. 5(a), one probing session of IGI using the binary search is depicted. In this IGI session, 6 probing packet trains are sent out. In each train, packets are supposed to be spaced with a constant gap in order to generate a packet train at a particular probing rate. We observe that this packet gaps are clustered at a certain range. Nevertheless, the accuracy of this IGI probing sequence is acceptable in that the converged probing rate, $500 \times 8 \text{ bit} / 0.77 \text{ msec} = 5.2 \text{ Mbps}$, is approaching the actual available bandwidth 5Mbps. Fig. 5(b) shows the inter-arrival time of two packet chirps in pathChirp. This exponentially decreasing packet gaps in a chirp are generated accurately.

In rate-based probing sequences, the rate of each packet train is maintained at a constant. Pathload and DietTOPP generate rate-based probing sequences. As shown in Fig.6(a), this Pathload probing sequence

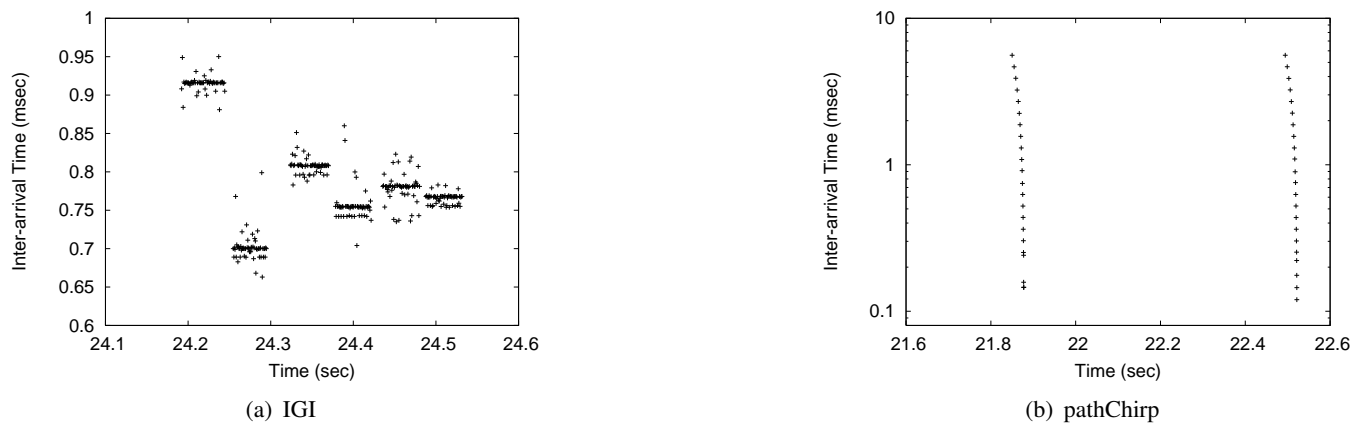


Fig. 5. Packet inter-arrival process of the gap-based probing sequences: IGI versus pathChirp

consists of 6 packet trains during the measurement session. The packet gaps in each packet train are maintained at a constant; therefore, the probing rates of these packet trains are controlled very accurately. Similar observations are found in Fig. 6(b), in which the probing rates of packet trains in DietTOPP are also controlled very well.

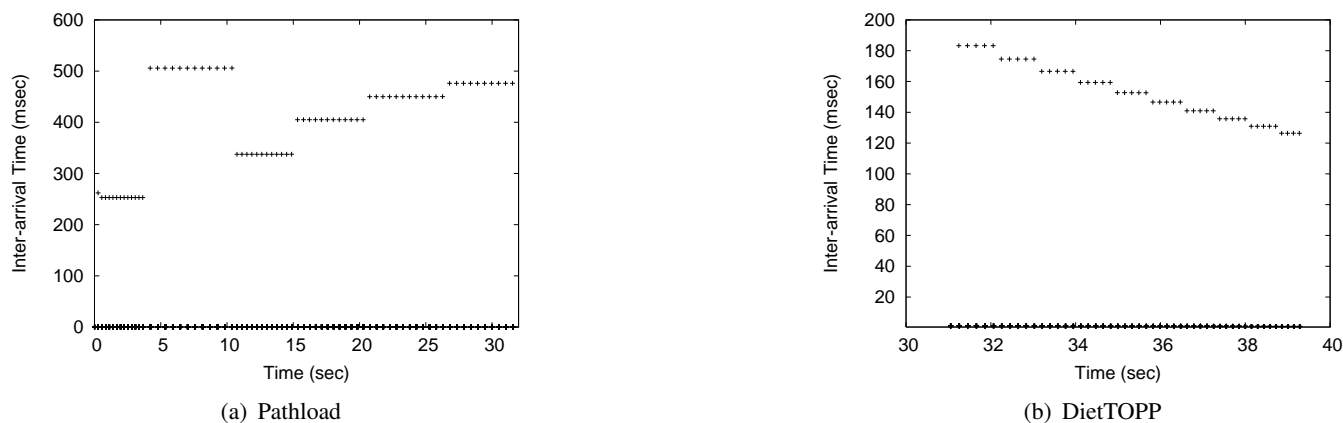


Fig. 6. Packet inter-arrival process of the rate-based probing sequences: Pathload versus DietTOPP

In summary, both the gap-based and rate-based probing sequences can be generated with reasonably good accuracy in practice. Nevertheless, it is easier to generate rate-based probing sequences than gap-based sequences because gap-based probing sequences require to schedule the transmission of each probing packets at some time gap while the rate-based probing sequence only requires the scheduling operations of the timer at much more coarse time granularity.

D. Performance of IGI, Pathload, pathChirp, Spruce and DietTOPP

Among the measurement tools under our investigation in this section, IGI and Pathload terminate automatically when the convergence criteria are satisfied. pathChirp, Spruce and DietTOPP are equipped with an open-loop control. Without an explicit termination, the measurement session continues. We apply the recommended termination criteria for Spruce and DietTOPP. However, there are no suggested termination criteria for pathChirp. As shown in Fig. 7, we examine the probing sessions of pathChirp under the CBR and Poisson cross traffic. We observe that it is safe to terminate a pathChirp session over 25 seconds to obtain one robust measurement result. In the remaining pathChirp measurements, we control the duration of the measurement sessions to be 25 seconds.

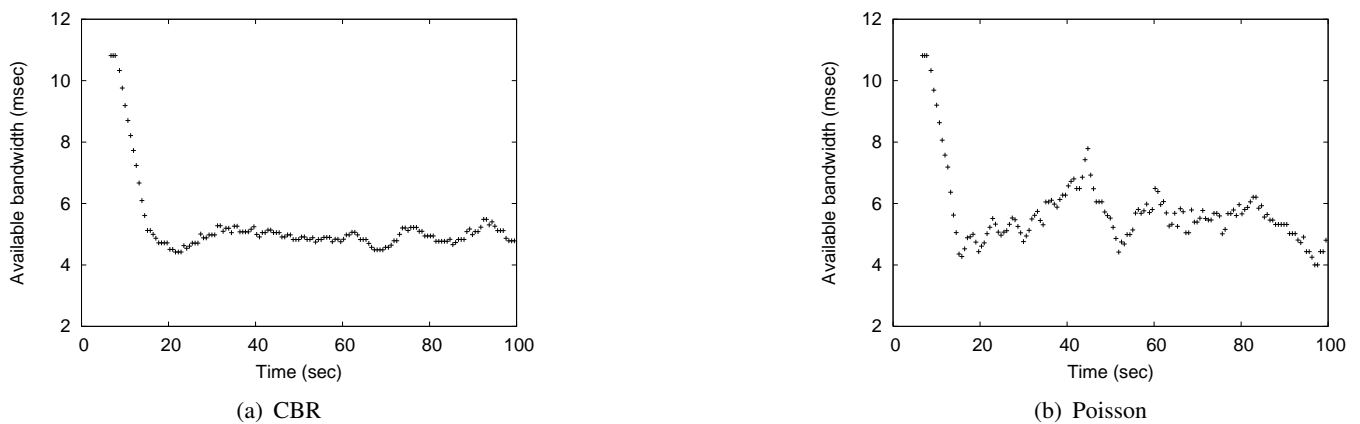


Fig. 7. Probing session of pathChirp under different cross traffic

We show the performance of IGI, Pathload, pathChirp, Spruce and DietTOPP with 5Mbps CBR cross traffic in the box plots in Fig. 8. The accuracy, convergence and overhead results of these tools are plotted in Fig. 9 with the Poisson cross traffic of 5Mbps average bit rate. In both cases, the ideal value of the available bandwidth is 5Mbps. With CBR cross traffic, IGI shows the most accurate results (4.6 Mbps) with small convergence time and low probing overhead. Pathload returns a narrow range of available bandwidth (4.0 Mbps). The available bandwidth ranges of Pathload are also close to the true available bandwidth value 5Mbps; however, Pathload requires a long time (~ 70 seconds) to converge and injects a large amount of probing traffic (~ 3 MB) in each measurement. PathChirp shows more significant variable measurement results than IGI and Pathload in spite that we apply the measurement duration of pathChirp up to 25 seconds. Such a long measurement duration also leads to significant measurement overhead up to 1MB. The results of Spruce scatter in the range of 5Mbps. This is an indication that the robustness of Spruce measurements is not very good; nevertheless, its convergence time is ~ 10 seconds

and the probing overhead is also low. DietTOPP shows reasonable accurate available bandwidth estimates (~ 4.4 Mbps) with medium converge time ~ 14 seconds and high measurement overhead (~ 2 MB). When the cross traffic is Poisson, all the five tool under investigation much more variable estimates of the available bandwidth. In particular, most of the pathChirp measurements over estimate the available bandwidth. This is not good in practice since such an over estimation may encourage applications eat more available bandwidth and aggravate network congestion.

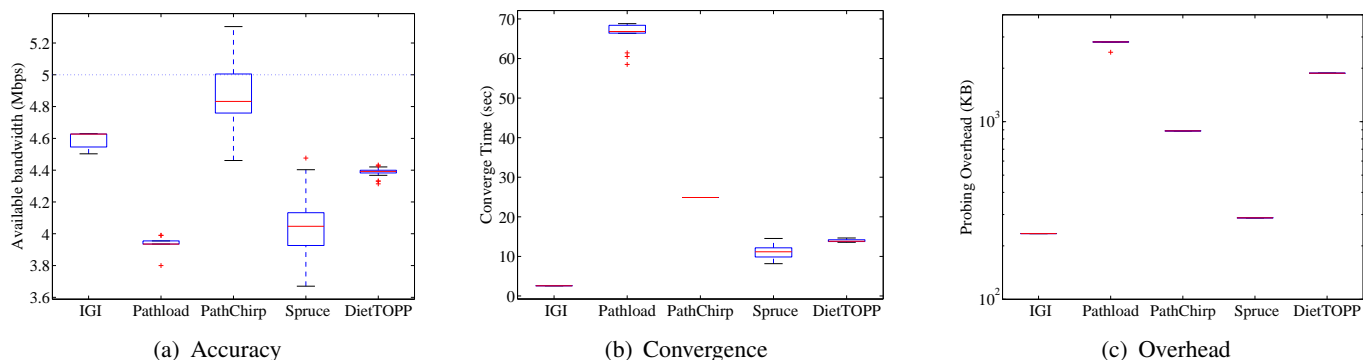


Fig. 8. Performance of IGI, Pathload, pathChirp, Spruce and DietTOPP in the single-bottleneck testbed with CBR cross traffic (5Mbps)

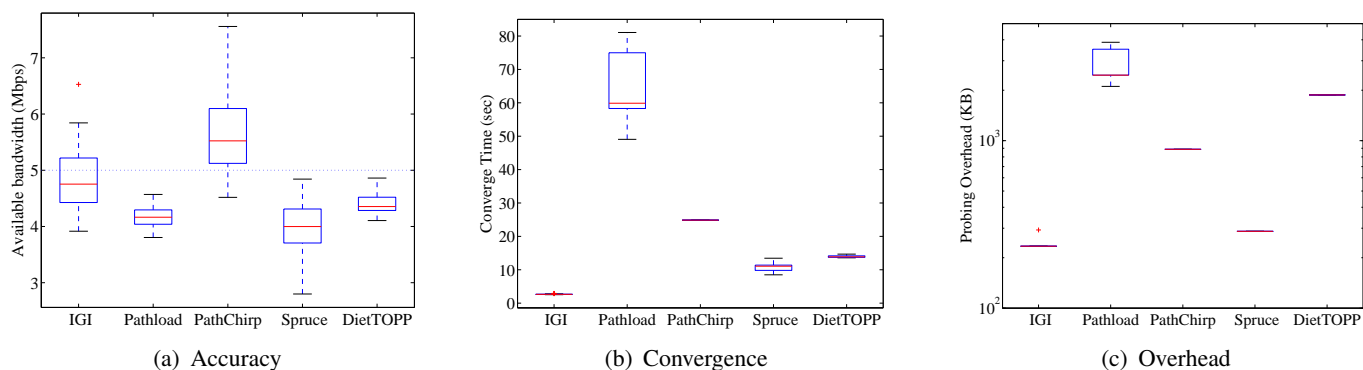


Fig. 9. Performance of IGI, Pathload, pathChirp, Spruce and DietTOPP in the single-bottleneck testbed with Poisson cross traffic (5Mbps)

E. Performance of ECNProbe

In Fig. 10, we show the performance of the proposed ECNProbe with CBR cross traffic under different loads in the box plots. With the Poisson cross traffic, the performance of the ECNProbe is plotted in Fig. 11. We consider 5 cases with cross traffic loads of 0, 2Mbps, 4Mbps, 6Mbps and 8Mbps. With both CBR and Poisson cross traffic, ECNProbe returns available bandwidth estimates consistently in the close proximity of the ideal values. This is a good indication that ECNProbe is robust to the network dynamics. In Fig. 10(b) and Fig. 11(b), most ECNProbe runs terminate in less than 10 seconds. Those ECNProbe

estimates (> 10 seconds) occur when there is no cross traffic. One explanation is that this ECNProbe stream has difficulties to saturate the link and create congestion; hence, the tool requires longer time to converge to the ideal value 10Mbps. In Fig. 10(c) and Fig. 11(c), the probing overhead of ECNProbe is a little higher than IGI and pathChirp with the same order of the traffic volume for each estimate. ECNProbe has significant lower overhead than Pathload. Note that this set of experiments of ECNProbe take the default parameter settings $W = 60$ packets, $L = 500$ bytes and $\gamma = 0.05$. There might be room for improving the performance by setting the optimal parameters for the probing. ECNProbe shows more variable measurement results with the Poisson cross traffic than the CBR cross traffic in our experiments. We are now examining the evolution of the ECN marking probability for a given load with CBR and Poisson cross traffic so that we can understand the behaviors of ECNProbe more clearly. The future Internet routers may be equipped with various AQM schemes; it is also interesting how ECNProbe interacts with these AQM plus ECN schemes.

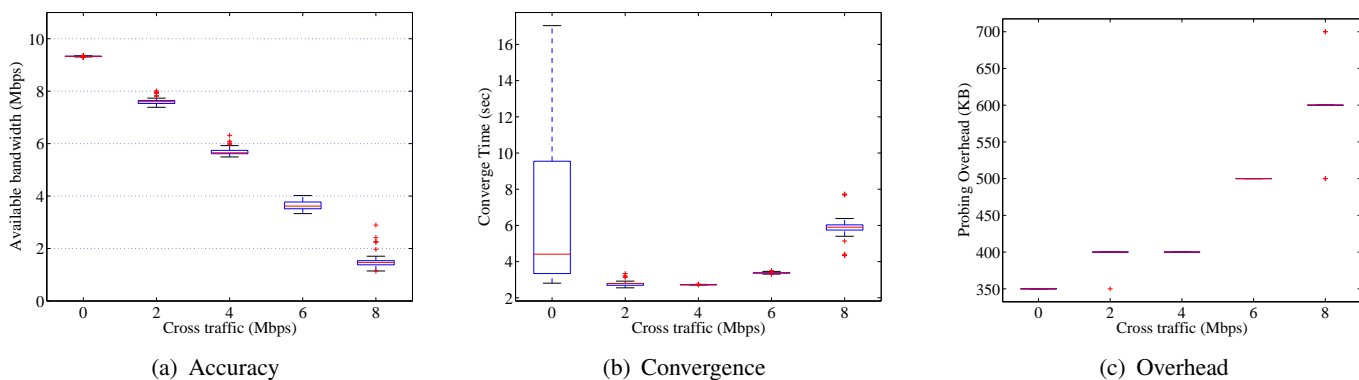


Fig. 10. Performance of ECNProbe in the single-bottleneck testbed with CBR cross traffic

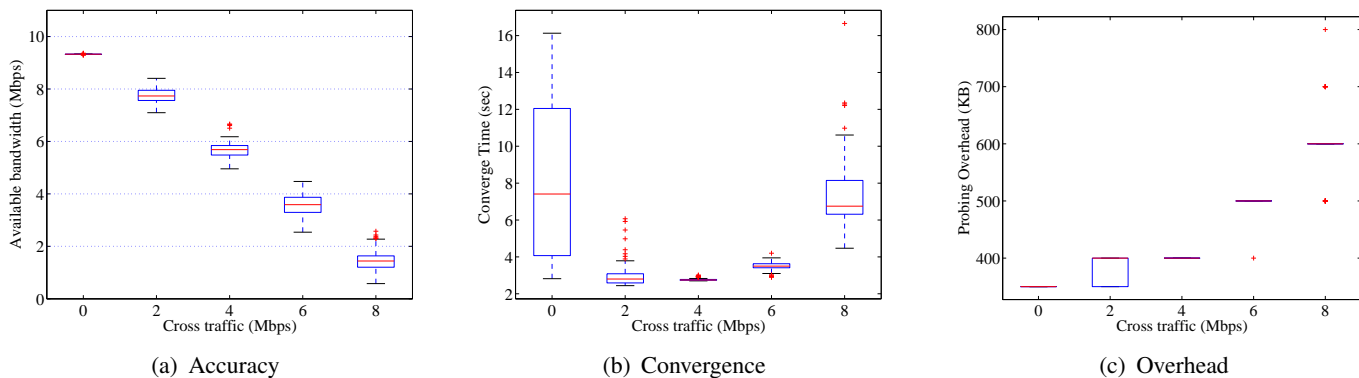


Fig. 11. Performance of ECNProbe in the single-bottleneck testbed with Poisson cross traffic

V. CONCLUSION

In this paper, we propose a unified self-congestion probing framework by bridging the self-congestion probing for available bandwidth and TCP congestion control. We reveal the difference and the relationship between the corresponding two bandwidth metrics: available bandwidth and TCP fair share rate. The rich research results on TCP may provide new avenues in improving the current self-congestion based bandwidth measurement tools. Based on this unified framework, we design and evaluate a simple available bandwidth probing scheme, ECNProbe. The preliminary results demonstrate that the proposed ECNProbe significantly improves the measurement accuracy with a similar convergence time and the overhead of IGI and pathChirp.

We conducted a measurement study on a Linux-based testbed with a single bottleneck to evaluate the performance of IGI, Pathload, pathChirp, Spruce, DietTOPP and ECNProbe. Although our experiments show negative results of the first four tools on their accuracy with Poisson cross traffic, it is worth further investigation because the parameter settings of the tools may have significant impact on their accuracy. This evaluation is still quite preliminary with simple cross traffic characteristics and a single bottleneck; however, it serves as a good starting point. Note that these four tools utilize different schemes to determine the congestion turning point for estimating available bandwidth. We are now constructing a common simulation framework to examine the impact of different congestion signals in determining an appropriate congestion turning point.

ACKNOWLEDGMENT

The authors would like to thank Miao Ma of HKUST for her constructive comments for improving the paper and we thank Donald Tse for his Linux implementation of ECNProbe in his final year project in the Broadband Network Laboratory of HKUST. We also thank anonymous reviewers for their constructive comments.

REFERENCES

- [1] "Comcast throttles bittorrent traffic, seeding impossible," 2007, <http://torrentfreak.com/comcast-throttles-bittorrent-traffic-seeding-impossible>.
- [2] 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.-Dec. 2003.
- [3] S. Alouf, P. Nain, and D. Towsley, "Inferring network characteristics via moment-based estimators," in *Proc. IEEE Conference on Computer Communications (INFOCOM)*, vol. 2, Piscataway, NJ, USA, 2001, pp. 1045–54.

- [4] X. Hei, B. Bensaou, and D. H. K. Tsang, "Model-based end-to-end available bandwidth inference using queueing analysis," *Elsevier Computer Networks Journal*, vol. 50, no. 12, pp. 1916–1937, Aug. 2006.
- [5] M. Jain and C. Dovrolis, "End-to-end available bandwidth: measurement methodology, dynamics, and relation with TCP throughput," *IEEE/ACM Trans. Netw.*, vol. 11, no. 4, pp. 537–549, 2003.
- [6] 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. 879–893, Aug. 2003.
- [7] A. Johnsson and M. Björkman, "Measuring the impact of active probing on TCP," in *Proc. International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS)*, July 2006.
- [8] S. Floyd, "TCP and explicit congestion notification," *SIGCOMM Comput. Commun. Rev.*, vol. 24, no. 5, pp. 8–23, 1994.
- [9] V. Sharma and R. Mazumdar, "Estimating traffic parameters in queueing systems with local information," *Perform. Eval.*, vol. 32, no. 3, pp. 217–230, 1998.
- [10] K. Salamatian, T. Bugnazet, and B. Baynat, "Cross traffic estimation by loss process analysis," in *Proc. ITC Specialist Seminar on Internet Traffic Engineering and Traffic Management*, July 2002.
- [11] B. Melander, M. Bjorkman, and P. Gunningberg, "A new end-to-end probing and analysis method for estimating bandwidth bottlenecks," in *Proc. IEEE Global Internet Symposium*, vol. 1, San Francisco, Nov. 2000, pp. 415 – 420.
- [12] J. Strauss, D. Katabi, and F. Kaashoek, "A measurement study of available bandwidth estimation tools," in *Proc. ACM Internet Measurement Conference (IMC)*, 2003, pp. 39–44.
- [13] V. J. Ribeiro, R. H. Riedi, R. G. Baraniuk, J. Navratil, and L. Cottrell, "pathChirp: Efficient available bandwidth estimation for network path," in *Proc. Passive and Active Measurement Workshop (PAM)*, 2003.
- [14] X. Liu, K. Ravindran, and D. Loguinov, "A queueing-theoretic foundation of available bandwidth estimation: single-hop analysis," *IEEE/ACM Trans. Netw.*, vol. 15, no. 4, pp. 918–931, Aug. 2007.
- [15] C. Casetti, M. Gerla, S. Mascolo, M. Y. Sanadidi, and R. Wang, "TCP Westwood: end-to-end congestion control for wired/wireless networks," *Wirel. Netw.*, vol. 8, no. 5, pp. 467–479, 2002.
- [16] S.-C. Tsao, Y.-C. Lai, and Y.-D. Lin, "Taxonomy and evaluation of TCP-friendly congestion-control schemes on fairness, aggressiveness, and responsiveness," *IEEE Network*, vol. 21, no. 6, Nov.-Dec. 2007.
- [17] Y. Labit, P. Owezarski, and N. Larrieu, "Evaluation of active measurement tools for bandwidth estimation in real environment," in *Proc. 3rd IEEE/IFIP Workshop on End-to-End Monitoring Techniques and Services (E2EMON)*, May 2005, pp. 71 – 85.
- [18] C. D. Guerrero and M. A. Labrador, "Experimental and analytical evaluation of available bandwidth estimation tools," in *Proc. IEEE LCN Workshop on Network Measurements*, Nov. 2006.
- [19] A. Shriram and J. Kaur, "Emperical evaluation of techniques for measuring available bandwidth," in *Proc. IEEE Conference on Computer Communications (INFOCOM)*, Anchorage, Alaska, May 2007, pp. 2162–2170.
- [20] J. F. Kurose and K. W. Ross, *Computer networking: a top-down approach*, 4th ed. Addison-Wesley, 2007.
- [21] D. Katabi, M. Handley, and C. Rohrs, "Congestion control for high bandwidth-delay product networks," in *Proc. ACM conference on applications, technologies, architectures, and protocols for computer communications (SIGCOMM)*, 2002, pp. 89–102.
- [22] V. Jacobson, "Congestion avoidance and control," in *Proc. ACM conference on applications, technologies, architectures, and protocols for computer communications (SIGCOMM)*, 1988.
- [23] S. Floyd and T. Henderson, "The NewReno modification to TCP's fast recovery algorithm," RFC 2582, Apr. 1999.
- [24] L. S. Brakmo, S. W. O'Malley, and L. L. Peterson, "TCP Vegas: new techniques for congestion detection and avoidance," in *Proc. ACM conference on Communications architectures, protocols and applications (SIGCOMM)*, 1994, pp. 24–35.
- [25] S. Chen, B. Bensaou, and J. Zhu, "On estimating the bandwidth share of TCP connections in presence of reverse traffic," in *Proc.*

- IEEE International Conference on High Performance Switching and Routing (HPSR)*, New York, USA, May 30 - June 1 2007, pp. 1–6.
- [26] A. Medina, M. Allman, and S. Floyd, “Measuring the evolution of transport protocols in the Internet,” *SIGCOMM Comput. Commun. Rev.*, vol. 35, no. 2, pp. 37–52, 2005.
- [27] “Iperf,” the National Laboratory for Applied Network Research (NLANR), <http://dast.nlanr.net/Projects/Iperf/>.
- [28] “Poisson traffic generator,” Rice University, http://www.spin.rice.edu/Software/poisson_gen.
- [29] “Distributed internet traffic generator (D-ITG),” universita’ di Napoli “Federico II”, <http://www.grid.unina.it/software/ITG/index.php>.
- [30] “Traffic generator tool (TG),” the University of Southern California, <http://www.postel.org/tg/tg.htm>.
- [31] “The multi-generator toolset (MGEN),” naval Research Laboratory (NRL) PROTOcol Engineering Advanced Networking (PROTEAN) Research Group, <http://manimac.itd.nrl.navy.mil/MGEN>.
- [32] A. Johnsson, B. Melander, and M. Björkman, “DiefTopp: A first implementation and evaluation of a simplified bandwidth measurement method,” in *Second Swedish National Computer Networking Workshop*, Karlstad, November 2004, p. 5. [Online]. Available: <http://www.mdh.se/index.php?choice=publications&id=0795>
- [33] “tcpdump,” <http://www.tcpdump.org/>.

APPENDIX A

DEFAULT PARAMETERS FOR THE MEASUREMENT TOOLS

TABLE III
DEFAULT PARAMETERS OF IGI

Packet number	60 packets
Packet size	500 bytes
Mode	binary

TABLE IV
DEFAULT PARAMETERS OF PATHLOAD

Fleet length	12 streams
Stream length	100 packets
Bandwidth resolution	$\omega = 200$ kbps
Grey region resolution	$\chi = 200$ kbps
PCT threshold	0.55
PDT threshold	0.4
Aggregate threshold	0.6

TABLE V
DEFAULT PARAMETERS OF PATHCHIRP

Packet size	1000 bytes
Spread factor	1.2
Decrease factor	1.5
Busy period threshold	5
Convergence bound	25 seconds

TABLE VI
DEFAULT PARAMETERS OF SPRUCE

Packet size	1500 bytes
# of packet pairs	100

TABLE VII
DEFAULT PARAMETERS OF DIETTOPP

Packet size	1500 bytes
Probe for proportional share	
Packet number	48
Train number	10
Step	1
Probe for bandwidth estimate	
Packet number	16
Train number	5
Step	10

APPENDIX B

CONFIGURATION PROCEDURE FOR THE TESTBED

A. Hardware Settings

As shown in Figure 2, the hardware settings are tabulated in Table VIII.

TABLE VIII
HARDWARE SETTINGS IN THE TESTBED

device	IP address	hardware
cross traffic sender	192.168.0.1	Dell Optiplex GX150
probe sender	192.168.0.2	Dell Optiplex GX270
cross traffic receiver	192.168.1.1	Dell Optiplex GX150
probe receiver	192.168.1.2	Dell Optiplex GX150
switch (100Mbps)		Xchanger SOHO 2080FP LAN Switch
hub (10Mbps)		Surecom EtherPerfect 508T

B. Router

- Configure two network cards

```
ifconfig eth0 192.168.0.254 up
```

```
ifconfig eth1 192.168.1.254 up
```

- Enable the routing function

```
echo "1" > /proc/sys/net/ipv4/ip_forward
```

- Configure the fire wall

```
iptables -F
```

C. End-hosts

There are 4 end-hosts in the single-bottleneck testbed. The IP addresses of these hosts are also configured in Table VIII.

- Configure the network card

```
ifconfig eth0 192.168.0.1 up
```

- Configure the routing table

```
router add -net 0.0.0.0/0 gw 192.168.0.254
```

- Configure the fire wall

```
iptables -F
```

- Enable ECN:

```
echo "1" > /proc/sys/net/ipv4/tcp_ecn
```

- Configure RED:

```
tc qdisc add dev eth0 root red limit 256000 min 12000 max 32000  
avpkt 1000 burst 20 probability 0.02 bandwidth 10000 ecn
```