Characterizing Tight-Link Bandwidth of Multi-Hop Paths Using Probing Response Curves

Seong-Ryong Kang
LTE Systems Lab, Samsung Electronics
Suwon, Gyeonggi 443-742, Korea
Email: sseong.kang@samsung.com

Dmitri Loguinov
Department of Computer Science, Texas A&M University
College Station, TX 77843, USA
Email: dmitri@cs.tamu.edu

***Abstract*—Bandwidth estimation plays an important role in characterizing Internet paths. Existing approaches can be classified into *measurement tools* [3], [6], [9], [10], [16], [17], [23], [29], which usually have extensive simulations, but no convergence analysis for general cross-traffic, and *theoretical models* [5], [14], [19], [20], [22], which usually have provable convergence, but no practical implementation. Another issue in related work is the unknown performance of certain proposed algorithms in real networks where delay measurements are not perfect due to various OS and hardware-related timing irregularities [26]. We address the former issue by developing a measurement tool PRC-MT that not only achieves asymptotic accuracy in multi-path networks with arbitrary cross-traffic, but also simultaneously measures the capacity and available bandwidth of the tight link. We address the latter issue by performing a comparison study of existing tools in Emulab and assessing their susceptibility to timing irregularities of end-hosts. Our results show that PRC-MT outperforms all existing tools in terms of accuracy, achieves similar convergence delay, and does not require any manual configuration. We also find that interrupt moderation may cause existing tools (such as Pathload [10], Pathchirp [27], and CapProbe [16]) to become quite inaccurate in certain network configurations and exhibit behavior completely different from that in `ns2` [28].**

I. Introduction

Bandwidth of Internet paths is an important metric for many applications. However, without a direct access to network resources, end-to-end bandwidth measurement under general conditions on cross-traffic is a rather complex process [5], [20]. Unfortunately, theoretically justified techniques [19], [20], [22] are commonly not available as practical tools that can be used in real networks and vice versa (i.e., existing implementations are often based on fluid models that exhibit bias in bursty networks [20] and/or rely on heuristics with unknown theoretical performance). In addition, many current techniques produce unreliable results in actual networks where packet dispersions cannot be sampled accurately due to hardware interrupt moderation and various OS-imposed overhead [26]. Our goal in this paper is to provide a useful implementation of a theoretical model and examine its performance in non-simulated networks in comparison with existing tools. We perform this task below.

A. Measuring the Tight Link

Existing techniques usually estimate either the *available bandwidth* [6], [11], [27], [29], or the *bottleneck bandwidth* [3], [16] of the path. The former term refers to the unused bandwidth A_t of the *tight* link (i.e., link with the smallest available bandwidth) and is closely related to the rate at which new applications can send into the path without congesting it. The latter metric is the capacity C_n of the *narrow* link (i.e., link with the lowest speed), which can be viewed as an upper bound on the sending rate that the path can support. Note that A_t can be measured in all network configurations, while this is not necessarily true for C_n.

Even though both A_t and C_n are useful metrics, certain applications require capacity C_t of the tight link instead of C_n, which allows them to compute the utilization of the tight link and possibly achieve better characterization of what causes bottlenecks in the path.[1] Only a few approaches can measure C_t [13], [14], [23], but they are either based on single-hop models that are inaccurate in multi-path networks, or rely on hop-by-hop probing, which we do not study in this paper. A recent theoretical development [20] shows that both A_t and C_t can be provably measured in any end-to-end path with *infinite* buffers by exploiting a certain piece-wise linear relationship between the sending rate r_I of probe packets and the corresponding arrival rate r_O at the receiver. Although this work opens a door for developing a new characterization technique for tight links, it remains to be seen if an automated implementation can achieve good performance in networks with *limited* buffer space and exhibit overhead comparable to that of existing tools.

Recall that [20] relies on correctly identifying the first two linear segments of the *probing response curve* (PRC), which is a functional relationship between r_I/r_O and r_I. Identifying and separating the linear segments in a stochastic PRC is a non-trivial task since the curve itself may fluctuate and/or deviate from the fluid piece-wise linear limit depending on path-specific characteristics as well as the number of probes per train and their size. In addition, building an entire PRC sometimes requires sending a large amount of traffic into the path and exhaustively probing a wide variety of sending rates (i.e., as done in [21]). Thus, the main challenge in PRC-based estimation is the development of automated algorithms for detecting linear segments in practice and selecting probing

[1] Also note that certain bandwidth estimation tools [6], [29] require C_t in order to measure A_t.

rates that result in quick convergence of the method.

In this paper, we tackle the above problems using an iterative probing technique that we call *PRC Measurement Tool* (PRC-MT), which is capable of estimating both A_t and C_t in arbitrary multi-hop paths. PRC-MT autonomously selects probing rates, train and packet size, and termination conditions so as to achieve any desired tradeoff between accuracy and overhead (i.e., better accuracy requires more probes and vice versa). We implement PRC-MT in Linux and evaluate its performance using Emulab [4]. We find that PRC-MT, limited to the execution delay of prior methods (i.e., $90-120$ seconds), estimates A_t and C_t with $90-99\%$ accuracy in a wide range of network configurations.

B. Timing Irregularities

Attempting to run existing tools in PlanetLab, we found that some of them frequently produced no estimate at all (e.g., Pathload [10], [11], [26]) while others returned results that did not seem reasonable (e.g., CapProbe [16]). It became immediately clear that one of the main factors that differentiates bandwidth estimation in real networks from that in `ns2` is *end-host timing irregularities*, which include hardware interrupt moderation [26] and OS scheduling delay jitter (which depends on the CPU utilization of the host). We sampled a number of hosts in PlanetLab and found that many of them used interrupt moderation, which could be enabled at the sender (i.e., packets did not leave the host immediately), at the receiver (i.e., arriving packets were delivered to the OS "bunched up"), or at both. In fact, modern gigabit NICs enable interrupt moderation by default, which means that bandwidth-measurement tools that are not robust to timing irregularities are unlikely to be successful in real networks.

To reduce the effect of interrupt moderation, techniques such as Pathchirp [27] and the current version of Pathload [26] incorporate mechanisms that aim to "weed out" packets affected by interrupt delays. Specifically, Pathchirp requires manual modification to force it to send more probing packets to obtain an accurate estimate. For Emulab experiments in this paper, we use 6 times more packets per probing train (i.e., chirp) than the default value in order to achieve reasonable accuracy. This modification reduces the effect of interrupt delay, but prolongs the measurement. On the other hand, Pathload attempts to filter out affected packets without increasing the number of probing packets, which unfortunately has a limited effect when interrupt delays become non-trivial. This makes Pathload's estimation much more susceptible to error, which happens fairly often in practice. Kang *et al*. [15] overcome this problem in a recent proposal called *Interrupt Moderation Resilient Pathload* (IMR-Pathload), which utilizes signal de-noising techniques such as wavelet decomposition and window-based averaging in detecting a delay trend exists in one-way delay samples of probe packets.

We assess the performance of PRC-MT in scenarios with non-negligible interrupt delays in comparison with Pathload [26], IMR-Pathload [15], Pathchirp [27], IGI/PTR [6] using metric A_t and Pathrate [3], CapProbe [16] using metric C_t when the narrow link coincides with the tight link. For available bandwidth A_t, our results show that PRC-MT exhibits no negative side-effects related to interrupt moderation, converges in $90-140$ seconds in all examined topologies, and outperforms the other studied methods in terms of accuracy ($1-5\%$ error). We also find that IMR-Pathload's estimates are generally within 7% of the correct value and its convergence delay is $80-100$ seconds. After manually tweaking Pathchirp's train size and running duration, we were able to reduce its error to about 15% and execution time to 200 seconds; however, its default version performs much worse. Even though we supply IGI/PTR with the correct tight-link capacity C_t, both methods exhibit $40-60\%$ error, but on the bright side converge within just $3-5$ seconds.

For tight-link capacity C_t, PRC-MT's error is below 7% in all studied cases, while that of Pathrate exceeds 15% and that of CapProbe is close to 60%. The measurement delay of prior methods is also significantly higher than that of PRC-MT – almost 2200 seconds in Pathrate and 500 seconds in CapProbe. Our Emulab results suggest that existing methods (in their unmodified form) may experience certain non-negligible performance issues in real networks, while techniques PRC-MT (introduced in this work) and IMR-Pathload are much more likely to remain robust in practical settings. In fact, PRC-MT not only provides automatic self-configuration that overcomes interrupt-moderation effects and achieves quick convergence, but also simultaneously estimates A_t and C_t and is asymptotically accurate.

II. Bandwidth Estimation Based on Probing Response Curve

In this section, we investigate practical issues and difficulties of using PRC in measuring available bandwidth A_t and capacity C_t of the tight link. We then develop empirical algorithms that overcome these problems and lead to a new measurement tool called PRC-MT, which can measure both bandwidth metrics of the tight link over multi-hop paths under arbitrary cross-traffic and routing patterns. We start by describing the basic idea of this approach.

A. Basic Idea

Define r_I to be a sending rate of packets in a probe-packet train at the sender and r_O to be their arrival rate at the receiver. Further define $F = r_I/r_O$ to be the ratio of r_I and r_O under fluid cross-traffic. Then, F can be expressed as [20]:

$$F = \frac{r_I}{r_O} = \begin{cases} 1 & r_I \le A_t \\ \dfrac{\lambda_t + r_I}{C_t} & A_t \le r_I \le B \end{cases}, \qquad (1)$$

where λ_t is the amount of cross-traffic that traverses the tight link, A_t and C_t are the respective available bandwidth and capacity of the tight link, and B represents a certain input rate that is greater than A_t but no less than the second smallest available bandwidth of the path. Note that B is dependent on the routing matrix of cross-traffic that traverses the path and thus it is not possible to compute its value without

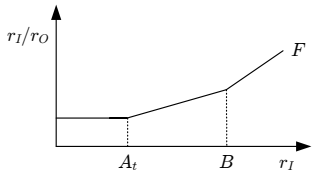
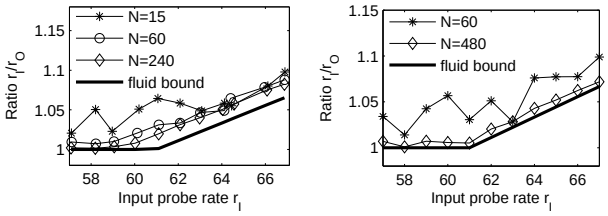


Fig. 1. Relationship between input rate r_I and output rate r_O.



(a) Emulab ($q = 1500$ bytes) (b) ns2 ($q = 200$ bytes)

Fig. 2. Probing response curves for a different probe-train length N in Emulab experiments and `ns2` simulations.

complete knowledge of cross-traffic routing patterns. Fig. 1 illustrates a hypothetical fluid response curve F, which shows the relationship between r_I/r_O and r_I.

Observe from the figure that F consists of piece-wise linear segments (at least two in multi-link paths), which breaks at particular input rates A_t and B. The first segment ends when input rate r_I equals the available bandwidth A_t of the tight link, while the second segment breaks down at rate B. Hence, to extract A_t, we need to identify the first break point where F starts to become larger than one. On the other hand, it requires to identify the second linear segment in F to compute the tight-link capacity C_t since $C_t = 1/\alpha$, where α is the slope of the second line segment.

Now the question we have is how to find the first break point for estimation of A_t and how to identify the second line segment and compute its slope α for capacity estimation C_t without even knowing the exact value of B in practice. We address these issues in the following subsections.

B. Difficulties

Define Z to be the real probing response curve of a path over which arbitrarily routed bursty cross-traffic flows traverse. Note that Z is different from the fluid curve F (as long as a probe-train length N and a probe-packet size q are finite) and this makes the task of identifying the first break point and the second linear segment in Z significantly more challenging than that in the fluid case. Note from [20] that Z is lower-bounded by fluid response curve F and asymptotically approaches F as $N \to \infty$ or $q \to \infty$. However, the difference between Z and F is non-zero (i.e., $Z - F > 0$) in real networks, where the size of packets is typically limited by the maximum transfer unit (MTU) of network elements and the packet-train length N cannot be arbitrarily large since router queue sizes are limited.

Before discussing implications of this deviation of Z from F, we explore how the response curve Z behaves with a different probe-train length N by conducting experiments in Emulab and `ns2` [24] using a single-hop topology of capacity $C_t = 90$ Mb/s. For this experiment, we keep link utilization at 32% (i.e., $A_t = 61$ Mb/s and $\lambda_t = 29$ Mb/s) and plot the response curve Z for several different values of N in Fig. 2.

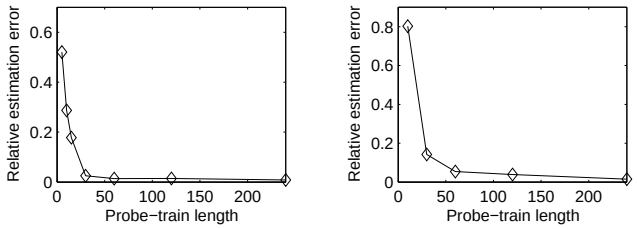
Notice in Fig. 2(a) that when N is small (e.g., 15), Z fluctuates substantially and exhibits large deviation from the fluid lower-bound F. However, as N increases, Z shows prominent two linear lines and its deviation from F becomes smaller. For example, with $N = 240$, r_O is within 1.5% of r_I until r_I reaches around 61 Mb/s, which is the available bandwidth A_t of the path in this setup. Note that the difference between Z and F would be zero as $N \to \infty$. We do not show this since we cannot use arbitrarily large N in Emulab without causing packet loss due to queue size limit. Instead, we conduct `ns2` simulations for different N with a smaller probe-packet size $q = 200$ bytes to better demonstrate behavior of Z for a large probe-train length N. As shown in Fig. 2(b), we observe that the behavior of Z is similar to the Emulab result and gets very close to F when $N = 480$.

Next, we discuss how the evolution of the real response curve Z on input probe rate r_I affects bandwidth estimation. Recall that to estimate the tight-link bandwidth A_t and C_t, we need to identify the end of the first line segment (for estimation of A_t) and stable second linear line (for extraction of C_t) from the response curve Z. For accurate discovery of the first break point in Z, it is required that variation in Z should be small for different r_I. More importantly, in order to extract accurate capacity estimates C_t, the second line segment in Z should be *parallel* to that in F even though they do not match (i.e., deviation $Z - F > 0$). Note that if the second line segments in Z and F are parallel, then we can use any two points on the line in Z to compute its slope α, which reflects the true capacity C_t of the tight link regardless of their locations as long as they are on the second line segment. Further note that under this condition, the amount of deviation $Z - F$ has no direct impact on estimation accuracy. However, if the second line segment in Z is not parallel to that of fluid counter part F, then estimation accuracy of C_t depends on which two points we select in computing the slope α, which makes the capacity estimation be more susceptible to measurement errors.

To confirm the above discussion and demonstrate the direct impact of the probe-train length N on estimation accuracy, we conduct experiments using the same single-hop setup. First define e_C and e_A to be the respective relative estimation errors of capacity and available bandwidth of the tight link L_t of a path:

$$e_C = \frac{|C_t - \tilde{C}_t|}{C_t}, \quad e_A = \frac{|A_t - \tilde{A}_t|}{A_t}, \tag{2}$$

where $\tilde{C}_t$ and $\tilde{A}_t$ are the respective estimates of the true capacity C_t and available bandwidth A_t of the tight link L_t. We then illustrate evolution of e_A and e_C for different N in Fig. 3. As Fig. 3(a) shows, e_A quickly drops from 28% (for $N = 10$) to a value that is less than 2% as N becomes 60. Similarly, estimation accuracy of C_t is significantly improved from $e_C = 80\%$ for $N = 10$ to $e_C = 3\%$ for $N = 120$ (see



(a) e_A (b) e_C

Fig. 3. Evolution of relative estimation errors e_A and e_C of PRC-MT for different N.

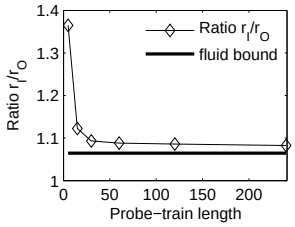


Fig. 4. Evolution of ratio r_I/r_O for different values of N.

Fig. 3(b)). These results indicate the importance of having sufficiently large N, which makes the line segments in the response curve Z straight without much fluctuation and allows accurate bandwidth estimation.

Note that even though a large N brings down fluctuation of line segments in Z, using an arbitrarily large value is not desirable since it increases measurement overhead and can also induce too much packet loss within a probe train. On the other hand, use of too small N results in high estimation errors. Hence, it is clear that there exists a trade-off between accuracy and overhead and thus proper selection of the probe-train length N is very important in developing PRC-MT. However, selection of appropriate value of N is non-trivial since fluctuation of line segments in the real response curve Z depends on many unknown factors (such as amount of cross-traffic) that are specific for a path under investigation.

Now, the problem we need to solve is how to select N in practice for a particular path such that the second line segment in Z becomes parallel to that in F. We investigate this next.

C. Parameter Selection

Recall that for a sufficiently large N, the slope of the second line segment in Z converges to a value that makes it parallel to that of fluid curve F in an input rate range $r_I \in [A_t, B]$. We can interpret this as that the ratio r_I/r_O saturates at a certain value when the probe-train length N becomes large. To confirm this, we examine the quantity of r_I/r_O for a different N using the setup discussed in the previous subsection. For this purpose, we send packets with rate $r_I = 68$ Mb/s (which is higher than the available bandwidth $A_t = 61$ Mb/s of the path in this setup) with a varying N and plot r_I/r_O in Fig. 4. As the figure shows, r_I/r_O quickly drops to a value that is slightly larger than the fluid-bound (i.e., $(\lambda_t + r_I)/C_t = 1.07$) as N increases. This leads us to investigating an empirical method, which iteratively probes for N that makes the ratio r_I/r_O saturate for a given input rate r_I.

In what follows below in this section, we discuss a simple selection procedure for N, which adjusts its value based on variation of r_I/r_O for a given sending rate r_I discussed above. Although there is no particular constraints on the input sending rate for this routine, it is preferable to use a rate that is not so smaller than the available bandwidth A_t of the path since variation of the ratio r_I/r_O for an input rate that is smaller than A_t diminishes rather fast with small increase in a probe-train length N [20]. Finding a rate that is no less than A_t would be sufficient for this purpose and Asymptotic Dispersion Rate (ADR) [2] of a path is a good candidate for the initial value of r_I since it is proven in [2] that $A_t < ADR$. Hence, PRC-MT probes for ADR by sending a single packet train and computes $ADR = q/E[y]$ at the receiver (where $E[y]$ is the average inter-packet dispersion of packets in the probe train). With the input sending rate r_I determined, the rest of the procedure for train-length probing is as follows.

Define γ to be the ratio of the current input and output rates: $\gamma = r_I/r_O$ and γ_{old} to be the previous value of γ. Further define N_{min} and N_{max} to be a respective minimum and maximum train length that can be adjusted by the user and σ to be a certain threshold that can vary between zero and one. The selection routine conducts binary search between N_{min} and N_{max} to find a value that saturates γ. This procedure tests if γ converges to a certain value for a given length N by sending to the receiver packet trains of length N with rate $r_I = ADR$. To assess saturation of γ, the selection procedure computes the following relative error metric ε:

$$\varepsilon = \frac{|\gamma - \gamma_{old}|}{\gamma}. \tag{3}$$

Then, the routine decreases N if $\varepsilon \leq \sigma$; increases otherwise.

Note that users can use any packet size $q_{min} \leq q \leq q_{max}$ (where we use $q_{min} = 200$ bytes and $q_{max} = 1500$ bytes), in which case N_{min} and N_{max} are scaled up or down by q_{max}/q. This routine ensures us to select a larger N when a user chooses a smaller packet q, sufficing the condition for stabilizing fluctuation of the line segments in Z as discussed in the previous subsection. For experiments in Sections III and IV, we use $N_{min} = 60$, $N_{max} = 3000$, $\sigma = 0.02$, and $q = 200$ bytes.

D. Bandwidth Probing

With the probe-train length N in hands, our next question is how to identify the first break point in the response curve Z, at which the input sending rate r_I starts to become larger than the arrival rate r_O (see Fig. 1). To efficiently search for this point, PRC-MT uses iterative probing-based search, which is similar to Pathload [11]. Note, however, that the two tools are different in a way that assesses whether an input rate r_I corresponds to A_t. For example, PRC-MT determines if the current rate $r_I > A_t$ by *directly* comparing r_I with r_O, while Pathload *infers* it by examining one-way delays of the probe packets.

For probing, PRC-MT sends a group of K packet-trains with a given rate r_I. Then, based on how much fraction η of them belongs to either $r_I > r_O$ or $r_I < r_O$, it adjusts its sending rate r_I for next K probe-trains. Specifically, PRC-MT decreases r_I if ηK probe-trains are asserted to be $r_I > r_O$; increases r_I if they are asserted to be $r_I < r_O$. Note that it is possible that neither of the above two cases happens (i.e., number of probe-trains that belong to either $r_I > r_O$ or $r_I < r_O$ are less than ηK). If this is the case, we treat it like a "grey region" in Pathload (see [11] for details).

Let $[W_L, W_H]$ be an available bandwidth range updated after each round of K probe-train measurements, where W_L represents the highest rate that has been identified to be less than A_t for a certain round and W_H represents the lowest rate that has been identified to be higher than A_t up to that round. After each round of probing, PRC-MT updates the bandwidth range and selects a new probing rate r_I for next round using the way Pathload does. This search process continues until the bandwidth range $[W_L, W_H]$ around A_t becomes smaller than a certain threshold ω that can be automatically selected (e.g., $\omega = 0.02ADR$) based on measured ADR or given by the user. PRC-MT returns $\tilde{A}_t = (W_L + W_H)/2$ as the available bandwidth estimate of the tight link when its internal algorithm terminates. We empirically set $\eta = 60\%$ and $K = 12$ as their respective default value in PRC-MT.

After finished probing the available bandwidth, PRC-MT starts a procedure for the tight-link capacity C_t estimation. The main focus of this routine is to select two points that will be used to extract C_t from the second line segment. Note that to facilitate estimation of C_t, PRC-MT records the sending rate r_I and its corresponding receiving rate r_O during available bandwidth probing whenever the current r_I reduces the upper bound W_H due to r_I being larger than r_O. These recorded points are the possible candidates for computing a capacity estimate $\tilde{C}_t$.

Note that we can select any two among the recorded points to extract C_t in ideal case (i.e., the second segment is a perfect straight line and there is no measurement noise). Unfortunately, however, there is no straightforward method that chooses optimal two points with certain measurement noise and an imperfect straight line, which leads us to exploring empirical method (which we explain below).

Assume that there are $m \geq 2$ recorded points (x_1, y_1), $\ldots, (x_m, y_m)$, where x_i and y_i $(i = 1, \ldots, m)$ are the respective sending and receiving rates used during the available bandwidth probing. PRC-MT first chooses (x_i, y_i), where x_i is the smallest among m points that satisfies $x_i \geq W_H$. We have two reasons for not using the point with $r_I = \tilde{A}_t$ as the first point. First, it is not very clear from the response curve Z where the second line segment starts around A_t (see Fig. 2(a)). The other reason is that the estimated value $\tilde{A}_t$ may be on the first line segment due to measurement error (e.g., $\tilde{A} < A_t$), in which case can result in high errors in capacity estimation.

Recall that to produce accurate capacity estimates $\tilde{C}_t$, the second point should be on the second line segment. Since PRC-MT picks the first point (x_i, y_i) whose sending rate x_i is closest to W_H among recorded points, we may consider the point that is close to the first one as the best candidate for the second point since it has higher chance to be on the second line segment. However, if two points are close to each other, then computing the linear slope of the two points is more susceptible to measurement noise. Hence, it is better to have the second point as far away as possible from the first one as long as they are on the second line segment. Thus, we select the farthest two points among the recorded points to extract C_t. Based on the above discussion, PRC-MT uses (x_j, y_j) as the second point, where x_j is the largest among the recorded points.

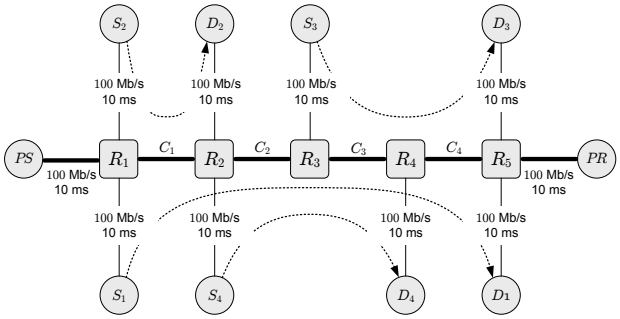


Fig. 5. Evaluation topology.

Having two points (x_i, y_i) and (x_j, y_j) selected, PRC-MT computes the tight-link capacity estimate $\tilde{C}_t$:

$$\tilde{C}_t = \frac{y_i y_j (x_i - x_j)}{x_i y_j - x_j y_i}, \tag{4}$$

which is inverse of the slope of the linear line segment between $(x_i, x_i/y_i)$ and $(x_j, x_j/y_j)$.

Before concluding this section, we should note that if the number of recorded points m is less than 2, PRC-MT requires to send additional packets with rates r_I that is larger than $\tilde{A}_t$ to obtain (r_I, r_O) pairs. Even though this case will rarely happen (only when a very large threshold ω is used to terminate the algorithm), we include this for the sake of completeness.

III. Performance of PRC-MT

To evaluate the performance of PRC-MT, we conduct experiments in Emulab[2] [4]. We examine estimation accuracy of PRC-MT and its convergence behavior and then compare these results with those in existing methods. For experiments in this section, we do not use interrupt moderation (i.e., interrupt delay $\delta = 0$) at the receiver and defer discussion of these tools under interrupt delay to Section IV. We start by describing the experimental setup.

A. Experimental Setup

For all experiments, we use a topology shown in Fig. 5, in which source PS sends probe data to the destination PR through five routers $R_1, \ldots, R_5$. Nodes S_i $(i = 1, 2, 3, 4)$ send

[2]In Emulab, users can change configuration of network interface cards.

TABLE I
EVALUATION SETUP

	Different link bandwidths (Mb/s)							
	C_1	A_1	C_2	A_2	C_3	A_3	C_4	A_4
Case-I	75	31.8	90	51.6	90	42.1	[60]	40.7
Case-II	75	41.3	90	70.7	90	46.7	[60]	26.4
Case-III	[60]	35.8	90	70.7	90	23.4	75	18.1
Case-IV	[60]	21.6	90	65.9	90	42.1	75	36.7
Case-V	[60]	50.2	90	61.1	90	41.9	75	50.8
Case-VI	75	28.9	90	37.8	90	13.8	[60]	31.2

TABLE II
PERFORMANCE OF PRC-MT ($\delta = 0$ μS)

Evaluation scenario	Relative estimation error		
	e_A	e_C	time
Case-I	3.49%	6.04%	89 sec
Case-II	2.35%	2.52%	115 sec
Case-III	0.88%	1.31%	96 sec
Case-IV	5.05%	3.51%	138 sec
Case-V	5.51%	6.38%	102 sec
Case-VI	9.74%	3.51%	125 sec

cross-traffic packets to destination nodes D_i ($i = 1, 2, 3, 4$) at an average rate λ_i. The speed of all access links is 100 Mb/s (delay 10 ms) and the remaining links L_i ($i = 1, 2, 3, 4$) between routers R_i and R_{i+1} have capacities C_i and propagation delay 40 ms.

To examine the estimation accuracy of PRC-MT, we use six different network settings shown in Table I, which lists the capacity and available bandwidth of each link for different experimental scenarios. Note that the table shows a fair amount of cross-traffic at each node, which is needed to ensure that each case represents some non-degenerate scenario. Without cross-traffic, most studied techniques are accurate and their comparison is not very insightful. The shaded values in each row represent the tight-link capacity C_t and available bandwidth A_t of the path for each case. The values in square brackets are the capacities C_n of the narrow link (i.e., bottleneck bandwidth) for each case. Notice from the table that the experimental settings cover all possible relationships between the location of the tight link and narrow link. For instance, in cases II and IV, the narrow link *coincides* with the tight link; in cases I and VI, the narrow link *follows* the tight link; while in cases III and V, the narrow link *precedes* the tight link.

In all experiments, we use TCP cross-traffic generated by the Iperf traffic generator [8] to load network paths. Although Iperf traffic does not exactly resemble Internet traffic, it is adequate for our purposes in this paper. We run 100 threads in each cross-traffic source S_i to generate TCP flows that are injected into routers R_1, R_2, and R_3 and keep the utilization of each router R_i according to the values shown in Table I. To maintain a fixed average utilization at each link in the experiment, we place an additional router (not shown in the figure) between node S_1 and router R_1, S_2 and R_1, S_3 and R_3, and S_4 and R_2 to limit the aggregate sending rate of TCP flows to the capacity of the additional router. The utilization of R_1, R_2, and R_3 is controlled by properly setting the capacity of the auxiliary router.

B. Estimation Accuracy of PRC-MT

We next investigate estimation accuracy and convergence behavior of PRC-MT. Experimental results of PRC-MT are summarized in Table II, which shows relative estimation errors e_A and e_C and convergence time of PRC-MT's internal algorithm. Note that like many other existing methods (e..g, Pathload [26], IGI/PTR [6]), PRC-MT's running time depends on round-trip delay (that includes propagation delay of each link and queuing delay of each intermediate routers) of the path under investigation.

As Table II shows, PRC-MT estimates available bandwidth of the tight link with over 90% accuracy for all cases studied in this paper. Its estimation accuracy of the tight-link capacity is as good as that of available bandwidth for all studied cases. In all experimentations, PRC-MT's algorithm converges within 140 seconds.

C. Performance Comparison

In this subsection, we compare PRC-MT with several existing available bandwidth estimators (Pathload [26], Pathchirp [27], and IGI/PTR [6]) and recent capacity estimation tools (Pathrate [2] and CapProbe [16]) with respect to estimation accuracy using the setup shown in Table I. For existing methods, we use user-level implementations[3] (which do not require super-user privilege to run the program) that are publicly available or obtained from the authors.

1) Available Bandwidth Comparison: We first compare PRC-MT with Pathload, Pathchirp, and IGI/PTR. We also have studied Spruce [29], but do not include its result here since it performs significantly worse than the other tools in all cases studied in this paper (see [18], [20] for details of Spruce and possible causes of its estimation inaccuracy in multi-hop paths).

Table III shows relative estimation errors e_A for different cases. For Pathload, we average the low and high values of the produced estimates after its internal algorithm terminates. For Pathchirp, we use "jumbo" option J that increases accuracy by sending more packets in each probe train (called chirp) than the default. We manually set this option $J = 6$ to send 6 times more packets than the default to produce accurate and reliable available bandwidth estimates. Selection of the value J is purely based on trial and error since Pathchirp does not offer any automatic selection mechanism for it. Different from other tools studied in this subsection, Pathchirp is an open loop system, which does not have an automatic convergence mechanism. It runs for a specified time t and stops when the running time reaches t, without knowing convergence of its estimate. We use $t = 200$ seconds to obtain results in this paper even though its default execution time is $t = 600$ seconds since its estimation accuracy has not been improved

[3]Note that kernel-level implementation can improve packet time-tamping at the measurement hosts, which in turn can lead to improved performance. However, we do not use this for fair comparison of existing estimation tools.

TABLE III
AVAILABLE BANDWIDTH ESTIMATION METHODS ($\delta = 0$ μS)

Evaluation scenario	Relative estimation error e_A							
	PRC-MT		Pathload		Pathchirp		IGI / PTR	
	e_A	time	e_A	time	e_A	time	e_A	time
Case-I	3.49%	89 sec	9.45%	69 sec	10.84%	200 sec	10.58/16.02%	3 sec
Case-II	2.35%	115 sec	8%	69 sec	8.53%	200 sec	4.21/9.93%	4 sec
Case-III	0.88%	96 sec	7.57%	70 sec	0.39%	200 sec	72.76/30.28%	5 sec
Case-IV	5.05%	138 sec	6.48%	69 sec	1.62%	200 sec	19.72/24.63%	6 sec
Case-V	5.51%	102 sec	16.58%	108 sec	19.81%	200 sec	13.38/5.31%	3 sec
Case-VI	9.74%	125 sec	15.01%	99 sec	18.04%	200 sec	98.56/59.24%	5 sec

TABLE IV
CAPACITY ESTIMATION METHODS ($\delta = 0$ μS)

Methods	Relative estimation error e_C			
	Case-II		Case-IV	
	e_C	time	e_C	time
PRC-MT	2.52%	115 (sec)	3.51%	138 (sec)
Pathrate	28.33%	2191 (sec)	21.67%	2191 (sec)
CapProbe	47.32%	500 (sec)	63.38%	500 (sec)

even we run it more than 200 seconds in our experimental setup. In IGI/PTR case, we use the estimates available at the end of its internal convergence algorithm. Note that we feed IGI/PTR the *exact* tight-link capacity C_t, while all other tools operate without this information.

As the table shows, Pathchirp produces estimates with less than 20% of error for all cases. Note that IGI/PTR produces estimates very quickly (40 times faster than Pathchirp), but its estimation error is significantly higher than that of Pathchirp (see [13] for details of possible causes of IGI's estimation inaccuracy even with more probe samples and longer measurement time in multi-hop paths). Pathload measures the paths with accuracy that is similar to Pathchirp. Notice in the table that PRC-MT produces bandwidth estimates with accuracy that is comparable to or better than those of Pathload and Pathchirp.

2) Bottleneck Bandwidth Comparison: Note that only in cases II and IV, the narrow link coincides with the tight link of the path. Hence, we compare PRC-MT only in these path configurations with recent bottleneck bandwidth estimators CapProbe and Pathrate. For CapProbe, we use 1800 packet-pairs for estimation since it often does not produce good estimates (on the studied paths) with 100 pairs recommended in the paper [16]. In Pathrate, the internal algorithm executes for over 2000 seconds (around 36 minutes) to get an estimate of the bottleneck capacity of the end-to-end path (note that Pathrate has quick termination mode that takes about 100 seconds, but we do not use this since its estimate is not accurate in the cases studied in this paper).

Table IV illustrates relative capacity estimation errors e_C of the different methods. As the table shows, PRC-MT produces capacity estimates $\tilde{C}_t$ of the tight link within 5% of its true values C_t in the studied cases, which is significantly better than those of Pathrate and CapProbe (see [13] for details of possible causes of CapProbe's random convergence and estimation inaccuracy in heavily congested paths).

IV. IMPACT OF INTERRUPT DELAY ON BANDWIDTH MEASUREMENT

As use of interrupt moderation (that delays generation of new interrupts) has become a common practice in modern network settings, host machines in real networks employ interrupt delays that vary widely in order to reduce CPU utilization and to increase network throughput. It is reported in [7] that Microsoft Windows-based operating systems perform best when Intel Gigabit NIC (GbE) controller interrupts with delays between 83 and 250 μs, while Linux-based systems perform best with interrupt delays between 125 and 1000 μs. Jin *et al.* [12] also report that a variety of systems equipped with Gigabit NICs require to delay generation of interrupts over 470 μs to achieve good throughput in receiving high-speed TCP streams and to substantially reduce CPU utilization.

To assess robustness of bandwidth estimation tools under the influence of interrupt moderation, we investigate how non-trivial interrupt delay affects the tools by comparing their estimation accuracy using the same setup in Table I. Among tools evaluated with no interrupt delay ($\delta = 0$) in Section III, use of interrupt moderation affects Pathload the most, while others exhibit estimation accuracy that is similar to that in the cases without using interrupt moderation. Recently, Kang *et al.* [15] proposed a measurement tool called IMR-Pathload that is resilient to various interrupt delays and significantly improves Pathload's estimation reliability under such conditions (see [15] for detailed behavior of Pathload and IMR-Pathload under a wide range of interrupt delays). We include this method in comparison of estimation tools under interrupt moderation in the following subsections.

A. Performance Comparison under Interrupt Moderation

We compare PRC-MT with IMR-Pathload as well as the same existing available bandwidth estimators and capacity estimation tools studied in Section III with non-trivial interrupt delay to understand their robustness to end-host interrupt moderation. For this purpose, we use the same path configurations discussed in Section III.

1) Available Bandwidth Estimation: Table V illustrates relative estimation errors e_A of all available bandwidth estimators studied in this paper for different cases under an interrupt delay $\delta = 500$ μs.

As the table shows, PRC-MT produces very accurate bandwidth estimates (less than 6% of error) and outperforms all

TABLE V
AVAILABLE BANDWIDTH ESTIMATION METHODS ($\delta = 500$ μS)

Evaluation scenario	Relative estimation error e_A									
	PRC-MT		IMR-Pathload		Pathload		Pathchirp		IGI / PTR	
	e_A	time	e_A	time	e_A	time	e_A	time	e_A	time
Case-I	3.71%	90 sec	5.12%	88 sec	−−	−−	7.22%	200 sec	62.34/22.8%	4 sec
Case-II	3.83%	89 sec	2.17%	89 sec	−−	−−	13.57%	200 sec	62.37/13.83%	4 sec
Case-III	1.55%	133 sec	6.78%	95 sec	−−	−−	5.14%	200 sec	44.14/44.81%	4 sec
Case-IV	0.19%	89 sec	3.24%	99 sec	−−	−−	13.01%	200 sec	59.03/21.25%	5 sec
Case-V	5.81%	92 sec	7.23%	79 sec	−−	−−	11.26%	200 sec	69.21/1.64%	3 sec
Case-VI	5.56%	96 sec	5.56%	80 sec	−−	−−	3.54%	200 sec	29.15/67.68%	5 sec

TABLE VI
CAPACITY ESTIMATION METHODS ($\delta = 500$ μS)

Methods	Relative estimation error e_C			
	Case-II		Case-IV	
	e_C	time	e_C	time
PRC-MT	1.72%	89 (sec)	7.65%	86 (sec)
Pathrate	17.5%	2191 (sec)	18.33%	2191 (sec)
CapProbe	57.65%	500 (sec)	81.77%	500 (sec)

other existing methods studied. Notice that IMR-Pathload's estimation accuracy is as good as PRC-MT, but Pathload is unable to produce estimates for any of the cases as shown in the table as empty cells. This suggests that Pathload's algorithm is susceptible to non-trivial interrupt delays (see [15] for details). Observe that Pathchirp exhibits estimation accuracy that is slightly worse than PRC-MT and IMR-Pathload in some cases (e.g., cases II, IV, and V), but comparable to them in other cases. Note that Pathchirp's estimation accuracy is similar to that observed without the influence of interrupt moderation (i.e., $\delta = 0$) (see Table III), which implies that its "jumbo" option (if selected properly) that sends substantially more probe packets makes it resilient to interrupt delays. Estimation accuracy of IGI/PTR is not much different from those cases with no interrupt delay shown in Table III, but is a lot worse than those of PRC-MT, IMR-Pathload, and Pathchirp.

2) Capacity Estimation: We compare PRC-MT with CapProbe and Pathrate in cases II and IV, in which tight link and narrow link coincide. We show relative estimation errors e_C of the above methods in Table VI. As the table shows, PRC-MT produces capacity estimates with over 90% accuracy, which significantly outperforms Pathrate and CapProbe in the studied cases.

B. Measurement Overhead

We next briefly discuss the amount of probe data used in bandwidth sampling for different methods. To allow Pathload's algorithm to terminate normally and produce some bandwidth estimate, we use a small interrupt delay $\delta = 100$ μs (other values of δ produce similar results and are omitted for brevity). For the existing methods, we use the same packet size and number of trains or packet pairs recommended in the original paper.

Table VII shows the number of packet samples and corresponding total amount of data used to get bandwidth estimates for cases II and IV. When tools use varying sizes of packets during probing, we show their average in the table. As the table shows, IGI/PTR and CapProbe do not use many samples while PRC-MT, IMR-Pathload, Pathload, Pachchirp, and Pathrate require significantly more probe packets for their measurement. Note that PRC-MT requires even more samples in case II than Pathrate to examine the path. However, since it uses smaller packet size on average, the amount of data used is significantly less than that of Pathrate. For instance, Pathrate sends 26000 samples in both cases, which amounts to 37 MB of data, while PRC-MT uses $4 - 8$ MB of probe packets. Also note that both IMR-Pathload and Pathload incur almost 4 times less overhead than Pathchirp.

Recall that in this paper we use "jumbo" option to increase Pathchirp's estimation accuracy since without using that option, its accuracy is significantly worse than that of Pathload. This result is somewhat different from that reported in [28], which conducted simulations using `ns2` with low link utilization (at most 53%). In our experimental setup used in this work where link utilization reaches up to 80%, Pathchirp were not able to produce accurate estimates without using the jumbo option due to higher cross-traffic interference, timestamping inaccuracy, and interrupt moderation. Higher measurement overhead of Pathchirp in this paper (different from that in [28]) accounts for the use of the jumbo option.

V. RELATED WORK

Bandwidth estimation has been extensively studied in the past 10 to 15 years and many techniques have been proposed in the literature. One direction of this prior work is to measure capacity of the tight link, which utilizes inter-packet spacings sampled at the receiver to identify the capacity mode presented in their histogram (bprobe [1], PBM [25], Nettimer [17], and Pathrate [2]) or uses minimally delayed packet pairs to produce its estimate (CapProbe [16]). Another dimension of related work focuses on measuring available bandwidth of the tight link (Pathload [11], IMR-Pathload [15], IGI/PTR [6], and TOPP [23]). These techniques adjust sending rates of probe packets to infer the available bandwidth using relationship between the sending rate and corresponding receiving rate. Note that unlike PRC-MT introduced in this paper, none of these methods can measure both bandwidth metrics in real network environments with arbitrary cross-traffic and routing patterns.

Recently, there have been a growing demand to evaluate existing tools under real network settings, where timestamping

TABLE VII
BANDWIDTH SAMPLING OVERHEAD ($\delta = 100\ \mu$S)

Methods	Case-II			Case-IV		
	Number of packets	Packet size (bytes)	Total probe data (MB)	Number of packets	Packet size (bytes)	Total probe data (MB)
PRC-MT	32,162	262	8.4	14,086	292	4.1
IMR-Pathload	8,400	239	2	12,000	225	2.7
Pathload	10,800	260	2.8	12,000	267	3.2
Pathchirp	7,560	1,000	7.5	7,560	1,000	7.5
IGI/PTR	600	1,000	0.6	800	1,000	0.8
Pathrate	26,000	1,454	37.8	26,000	1,454	37.8
CapProbe	1,800	1,000	1.8	1,800	1,000	1.8

inaccuracy, interrupt delays, and high link utilization of network paths can significantly affects estimation accuracy and reliability of existing tools. Although studies (such as [28]) based on `ns2` use traffic reproduced from real networks, their evaluation setup provides perfect timestamping and no interrupt delays and arguably does not capture the real behavior of these tools in the Internet.

VI. CONCLUSION

This paper implemented a new bandwidth measurement tool called PRC-MT that can extract both bandwidth metrics of the tight link over multi-hop paths under arbitrary cross-traffic and routing patterns. We evaluated PRC-MT in Emulab and showed that PRC-MT produces available bandwidth and capacity estimates with very high accuracy even under the influence of a large interrupt delay. We also evaluated existing bandwidth estimation tools under various network settings and found that Pathload is susceptible to timing irregularities caused by interrupt moderation while IMR-Pathload and Pathchirp showed resilience to such conditions.

REFERENCES

[1] R. L. Carter and M. E. Crovella, "Measuring Bottleneck Link Speed in Packet Switched Networks," *Performance Evaluation*, vol. 27–28, pp. 297–318, Oct. 1996.

[2] C. Dovrolis, P. Ramanathan, and D. Moore, "Packet-Dispersion Techniques and a Capacity-Estimation Methodology," *IEEE/ACM Trans. Netw.*, vol. 12, no. 6, pp. 963–977, Dec. 2004.

[3] C. Dovrolis, P. Ramanathan, and D. Moore, "What Do Packet Dispersion Techniques Measure?" in *Proc. IEEE INFOCOM*, Apr. 2001, pp. 905–914.

[4] Emulab. [Online]. Available: http://www.emulab.net/.

[5] P. Haga, K. Diriczi, G. Vattay, and I. Csabai, "Granular Model of Packet Pair Separation in Poissonian Traffic," *Computer Networks*, vol. 51, no. 3, pp. 683–698, Feb. 2007.

[6] N. Hu and P. Steenkiste, "Evaluation and Characterization of Available Bandwidth Probing Techniques," *IEEE J. Sel. Areas Commun.*, vol. 21, no. 6, pp. 879–974, Aug. 2003.

[7] Interrupt Moderation Using Intel GbE Controllers. [Online]. Available: http://download.intel.com/design/network/applnots/ap450.pdf.

[8] Iperf – The TCP/UDP Bandwidth Measurement Tool. [Online]. Available: http://dast.nlanr.net/Projects/Iperf/.

[9] V. Jacobson, "pathchar – A Tool to Infer Characteristics of Internet Paths." [Online]. Available: ftp://ftp.ee.lbl.gov/pathchar/.

[10] M. Jain and C. Dovrolis, "End-to-End Available Bandwidth: Measurement Methodology, Dynamics, and Relation with TCP Throughput," in *Proc. ACM SIGCOMM*, Aug. 2002, pp. 295–308.

[11] M. Jain and C. Dovrolis, "Pathload: A Measurement Tool for End-to-End Available Bandwidth," in *Proc. Passive and Active Measurement Workshop*, Mar. 2002.

[12] G. Jin and B. L. Tierney, "System Capability Effects on Algorithms for Network Bandwidth Measurement," in *Proc. ACM IMC*, Oct. 2003, pp. 27–38.

[13] S. Kang, X. Liu, A. Bhati, and D. Loguinov, "On Estimating Tight-Link Bandwidth Characteristics over Multi-Hop Paths," in *Proc. IEEE ICDCS*, Jul. 2006.

[14] S. Kang, X. Liu, M. Dai, and D. Loguinov, "Packet-Pair Bandwidth Estimation: Stochastic Analysis of a Single Congested Node," in *Proc. IEEE ICNP*, Oct. 2004, pp. 316–325.

[15] S. Kang and D. Loguinov, "IMR-Pathload: Robust Available Bandwidth Estimation under End-Host Interrupt Delay," in *Proc. PAM*, Apr. 2008, pp. 172–181.

[16] R. Kapoor, L. Chen, L. Lao, M. Gerla, and M. Sanadidi, "CapProbe: A Simple and Accurate Capacity Estimation Technique," in *Proc. ACM SIGCOMM*, Aug. 2004, pp. 67–78.

[17] K. Lai and M. Baker, "Measuring Bandwidth," in *Proc. IEEE INFOCOM*, Mar. 1999, pp. 235–245.

[18] L. Lao, C. Dovrolis, and M. Y. Sanadidi, "The Probe Gap Model can Underestimate the Available Bandwidth of Multihop Paths," *ACM SIGCOMM Comput. Commun. Review*, vol. 36, no. 5, pp. 29–34, Oct. 2006.

[19] X. Liu, K. Ravindran, B. Liu, and D. Loguinov, "Single-Hop Probing Asymptotics in Available Bandwidth Estimation: Sample-Path Analysis," *ACM IMC*, pp. 300–313, Oct. 2004.

[20] X. Liu, K. Ravindran, and D. Loguinov, "Multi-Hop Probing Asymptotics in Available Bandwidth Estimation: Stochastic Analysis," in *Proc. ACM IMC*, Oct. 2005, pp. 173–186.

[21] X. Liu, K. Ravindran, and D. Loguinov, "Measuring Probing Response Curves over the RON Testbed," in *Proc. Passive and Active Measurement Workshop*, Mar. 2006.

[22] S. Machiraju, D. Veitch, F. Baccelli, and J. Bolot, "Adding Definition to Active Probing," *ACM SIGCOMM Comp. Comm. Rev.*, vol. 37, no. 2, pp. 17–28, Apr. 2007.

[23] B. Melander, M. Björkman, and P. Gunningberg, "A New End-to-End Probing and Analysis Method for Estimating Bandwidth Bottlenecks," in *Proc. IEEE GLOBECOM*, Nov. 2000, pp. 415–420.

[24] ns2, "Network Simulator." [Online]. Available: http://www.isi.edu/nsnam/ns/.

[25] V. Paxson, "Measurements and Analysis of End-to-End Internet Dynamics," *Ph.D. Dissertation, Computer Science Department, University of California at Berkeley*, 1997.

[26] R. Prasad, M. Jain, and C. Dovrolis, "Effects of Interrupt Coalescence on Network Measurements," in *Proc. Passive and Active Measurement Workshop*, Apr. 2004.

[27] V. Ribeiro, R. Riedi, R. Baraniuk, J. Navratil, and L. Cottrell, "pathChirp: Efficient Available Bandwidth Estimation for Network Paths," in *Proc. Passive and Active Measurement Workshop*, Apr. 2003.

[28] A. Shriram and J. Kaur, "Empirical Evaluation of Techniques for Measuring Available Bandwidth," in *Proc. IEEE INFOCOM*, May 2007.

[29] J. Strauss, D. Katabi, and F. Kaashoek, "A Measurement Study of Available Bandwidth Estimation Tools," in *Proc. ACM IMC*, Oct. 2003, pp. 39–44.