A Novel Hybrid Probing Technique for End-to-End Available Bandwidth Estimation

Hui Lin∗†  Min Liu∗  Anfu Zhou∗  Huasha Liu∗†  ZhongCheng Li∗

∗Institute of Computing Technology, Chinese Academy of Sciences, Beijing, 100190, P.R.China
† Graduate School of Chinese Academy of Sciences, Beijing, 100190, P.R.China
Email: {linhui, liumin, zhouanfu, lihuasha, zcl}@ict.ac.cn

Abstract—The information of available bandwidth on an end-to-end path is important for various network applications, and several probing methods have been proposed to estimate it in recent years. However, previous methods are either based on fluid model or are only partially suitable for bursty real internet cross traffic; and the accuracy of their estimation degrades at different extents in multi-hop situations. Moreover, all previous PGM (Probing Gap Model) based methods require the knowledge of bottleneck link capacity, which may not be available in practice. In this paper, we extend the analysis of queuing behavior of probing packets from single-hop scenarios to multi-hop scenarios and propose a novel hybrid probing technique, called PATHCOS++, which integrates the advantages of both PRM (Probing Rate Model) and PGM based methods, to estimate the end-to-end available bandwidth. Unlike previous works, PATHCOS++ does not make fluid cross traffic assumption and does not require the information about bottleneck link capacity. Simulation results show that PATHCOS++ is quite efficient and provides end-to-end available bandwidth estimation that is significantly more accurate than current state-of-the-art techniques do. The accuracy of PATHCOS++ is nearly unaffected when there are multiple congestible links.

Index Terms—Active probing, bandwidth estimation, available bandwidth, measurement tools

I. INTRODUCTION

The information of available bandwidth of an end-to-end network path is useful for various applications, such as traffic engineering, admission control[1], TCP start up performance improvement[2], adaptive streaming[29], multi-path transmission[30], optimal routing in overlay networks[31]. Obtaining this information directly from routers is usually impossible due to the decentralized nature of internet and privacy issues. Thus, it is important to estimate the available bandwidth via some measurement techniques. These measurement techniques can be classified into two categories: passive measurement and active probing. Although passive measurement could potentially be very efficient, it relies on user data traffic and hence limits its application scope. On the other hand, active probing methods are more flexible and becomes the main way of available bandwidth estimation. Several active probing methods, such as C-probe[4], TOPP[8], Delphi[15], pathload[9], pathChirp[10], IGI/PTR[11], Spruce[12], and Bfind[14], have been proposed in recent years.

Recall that the end-to-end available bandwidth is defined as the minimal residual capacities of the links along the path within a certain interval. Consider an end-to-end path consisting of N links, denoted as \( L_1, L_2, \ldots, L_n \), and each of them has the capacity \( C_1, C_2, \ldots, C_n \), respectively. Let \( U_i(t) \) be a on/off function of time. E.g. \( U_i(t) = 1 \) if link \( L_i \) is busy in transmitting packets at time \( t \) and \( U_i(t) = 0 \) if link \( L_i \) is idle at time \( t \). The residual capacity of Link \( L_i \) during time interval \( [t, t+\tau] \) is given by

\[
R_i(t, t+\tau) = C_i \left[ 1 - \frac{\int_t^{t+\tau} U_i(t) \, dt}{\tau} \right] \quad (1)
\]

and the end-to-end available bandwidth during the time interval \( [t, t+\tau] \) is defined as

\[
R(t, t+\tau) = \min_{i=1,\ldots,n} R_i(t, t+\tau) \quad (2)
\]

\( R(t, t+\tau) \) often varies over different time \( t \) as well as over different observation interval \( \tau \). Note that the available bandwidth defined here does not equal to the achievable throughput of an application, because applications usually take transmission protocols such as TCP, which may be affected by other factors like packet loss, delay and reordering, they are unable to fully utilize the available bandwidth.

The challenges in inferring the available bandwidth are to take the estimation as accurately, efficiently and robustly as possible. By efficient we mean that estimation techniques should place minimal overhead on network and take measurement quickly, and by robust we mean that estimation techniques should remain usable in a variety of network situations, such as when the capacity of bottleneck link\(^1\) is unknown, when the cross traffic is bursty, or when there are multiple congestible links\(^2\). These challenges are also the focus of our work.

As described in Section II, previous techniques can be generalized into two categories: Probing Rate Model (PRM) and Probing Gap Model (PGM) methods. PRM methods are based on constant-rate fluid cross traffic model and iteratively

\(^1\)Bottleneck link is the link with the minimal residual capacity along the path. We use terms “bottleneck link” and “tight link” interchangeably.

\(^2\)Because we often take active probing techniques that may send probes at a rate higher than the residual capacity of some links, we call these links “congestible link.”
send probing packets at different rates and make efforts to find the turning point at which the output rate mismatches the input one. Unfortunately, the unique turning point of probing response curve does not exist when cross traffic is bursty. While PGM methods, such as IGI[11], may be partially suitable for bursty cross traffic, they all require the knowledge of capacity of bottleneck link, which may not be available in practice. Moreover, all of these methods become inaccurate when there are multiple congestible links. Motivated by the limitations of previous work, we developed PATHCOS++, a novel hybrid probing technique for available bandwidth estimation. By hybrid we mean that like PRM methods, PATHCOS++ takes the concept of self-induced congestion, and like PGM methods, PATHCOS++ make use of the relationship between the packet gaps to estimate the available bandwidth. However, the probing pattern and the algorithm PATHCOS++ uses in calculating the available bandwidth is different from any one of previous techniques. PATHCOS++ sends probing packets with rates controlled by a cos function, and find big bumps in the probing response curves to conduct the available bandwidth estimation without the information about the bottleneck link capacity. Unlike turning points, the big bumps can be easily identified even if the cross traffic is bursty. Experimental results show that the technique PATHCOS++ takes is quite promising.

Our main contributions are highlighted below:

- We extend the analysis of queuing behavior of probing packets from single hop scenarios to multi-hop scenarios and derive a deterministic model for available bandwidth estimation. Unlike previous works, this model does not make assumptions on the characteristics of cross traffic.
- Based on the model, we propose a novel efficient probing technique called PATHCOS++ that integrates the advantages of previous available estimation techniques. It is more accurate and robust than existing probing techniques.
- We point out that previous evaluation method can induce non-negligible bias if the process of cross traffic is not time invariant and stationary, and we propose a more rigid approach to evaluate the accuracy of the inference logic of available bandwidth estimation techniques. By using this approach we make a thorough experimental study on the performance of PATHCOS++ under various scenarios and compare PATHCOS++ with prominent current state-of-the-art techniques.

The rest of this paper is organized as follows. In Section II we survey previous works on available bandwidth estimation and show their limitations. In Section III, we analyze the queuing and delay behavior of probing packets in routers and derive a deterministic model for available bandwidth estimation. Based on this model, we propose PATHCOS++ in Section IV. In Section V we take NS[32] simulations to evaluate the performance of PATHCOS++ and compare it with existing available bandwidth estimation techniques. Section VI concludes this paper.
accuracy of these methods suffers. As pointed by X. Liu [20], [21], the unique turning point of response curve can only be recovered under some idealized conditions, e.g. by sending probing trains that have infinite length, which is impractical. Moreover, previous works [24], [22] also suggest that these methods tend to be inaccurate in multi-hop scenarios.

In the last a few years some probing techniques based on statistical methods are proposed, such as BART [27] and Traceband [26]. BART takes Kalman filters and Traceband takes Hidden Markov Models to estimate the available bandwidth. However, their basic sampling methods are similar as those described above. Besides, the validation of these methods in multi-hop multi-congestible link scenarios is still to be evaluated.

There are also some other measurement techniques related to available bandwidth such as Delphi [15] and the works described in [16]. These techniques assume specific cross traffic process and try to reconstruct its parameter. However, the probing part of these techniques are similar as Spruce. An increasing effort in recent study has been put on improving the theoretic understanding of available bandwidth estimation, such as [17], [18] and [19]. These works do not propose new probing methods and the topics are beyond the scope of this work.

III. NETWORK SCENARIOS AND ANALYSIS

Like most of previous works, we assume that:
1) the network consists of a series of store-and-forward nodes, each of them is equipped with a FIFO queue and has a constant service rate;
2) the packet delay results from propagation delay, service time and variable queuing delay;
3) the network is stable, e.g. the rate of cross traffic in each hop does not exceed the capacity of that hop;

Although some factors such as packet reordering and packet loss may also be an issue, we do not consider them here due to the page limit. We analyze the queuing behavior of the probing packet in single hop and multiple hop scenarios as follows.

A. Single Hop Scenario

Consider that there is only one link between the sender and the receiver. C is the capacity of the link and d is the propagation delay. A train of N probing packets 1, 2, ..., N, each with length s bits, are sent from the sender at time t1, t2, ..., tn and are received at time t′1, t′2, ..., t′n, respectively. Then, for the kth probing packet, we have

\[ t′_k = t_k + d + q_k + \frac{s}{C} \]  

where \( q_k \) is the queuing time of the kth probing packet.

For the probing packet \( k + 1 \), there are two possibilities:
1) The router keeps on working between the arrival time of probing packet \( k \) and \( k + 1 \), which means probing packet \( k \) and \( k + 1 \) are within the same busy period;
2) there is at least one idle period between the arrival time of probing packet \( k \) and \( k + 1 \) so that the two packets belong to different busy periods.

It should be noted that some previous PGM methods implicitly assume the two probing packets are within the same busy period. If this assumption does not hold, these methods would over estimate the cross traffic and underestimate the available bandwidth. Spruce sends packet pairs at the rate \( s/C \) to ensure that the packet pairs are within the same busy period.

We take the quadruple \( (t_k, t_{k+1}, t′_{k+1}, t'_k) \) as one sample of the path. If the probing packet \( t_k \) and \( t_{k+1} \) are within the same busy period, we call the sample “clean”; otherwise, we call it “contaminated”. We will try best to utilize clean samples and avoid contaminated ones to estimate the available bandwidth. Note that if \( q_k + \frac{s}{C} > t_{k+1} - t_k \), the sample would always be clean.

Let \( V(t_k, t_{k+1}) \) be the number of cross traffic packets that arrive at router between the arrival time of probing packet \( k \) and \( k + 1 \), \( cs_i \) be the size of the ith cross traffic packet, and \( W(t_k, t_{k+1}) = \sum_{i=1}^{V(t_k, t_{k+1})} \left( \frac{cs_i}{t_{k+1} - t_k} \right) \) be the service time spent on these cross traffic. If the sample is clean, we will have:

\[ q_{k+1} = q_k + \frac{s}{C} + W(t_k, t_{k+1}) - (t_{k+1} - t_k). \]  

From (3) and (4), the available bandwidth during interval \([t_k, t_{k+1}]\) can be estimated as

\[ R(t_k, t_{k+1}) = C \left( 1 - \frac{q_{k+1} - q_k - \frac{s}{C} + t_{k+1} - t_k}{t_{k+1} - t_k} \right) \]  

\[ = C \left( 1 - \frac{t'_{k+1} - t'_k - \frac{s}{C}}{t_{k+1} - t_k} \right). \]  

This is just what previous PGM methods use for available bandwidth estimation. Previous PGM methods all assume that the bottleneck capacity \( C \) is known. We will discuss how to estimate the available bandwidth without the knowledge of \( C \) later. We now firstly extend the analysis to multi-hop scenarios.

B. Multiple Hop Scenario

Assume that the path consists of \( N \) links, denoted as \( L_1, L_2, \ldots, L_j, \ldots, L_n \), where \( L_j \) is the bottleneck link. The capacity of the ith link \( L_i \) is \( C_i \). Probing packets 1, 2, ..., \( M \) are sent at time \( t_1, t_2, \ldots, t_M \) and are received at time \( t_{1}', t_{2}', \ldots, t_{M}' \). Let \( q_k \) be the queuing delay of packet \( k \) at the ith link, and \( d_i \) be the propagation delay of the ith link. We have:

\[ t_{i}' = t_k + \sum_{i=1, i \neq j}^{N} \left( q'_i + \frac{s}{C_i} + d_i \right) + q'_k + \frac{s}{C_j} + d_j. \]  

Let \( T_k \) be the time when the kth probing packet arrives at the bottleneck link, \( T_k = \sum_{i=1}^{j-1} \left( \frac{s}{C_i} + q'_k + d_i \right) \), and \( W(T_k, T_{k+1}) = \sum_{i=1}^{V(T_k, T_{k+1})} \left( \frac{cs_i}{t_{k+1} - t_k} \right) \) be the service time spent on cross traffic during time interval \([T_k, T_{k+1}]\) at link \( L_j \), if
probing packet \( k \) and \( k + 1 \) are within the same busy period, we would get

\[
q_{k+1}^i = q_k^i + \frac{s}{C_j} + W(T_k, T_{k+1}) - (T_{k+1} - T_k). \tag{7}
\]

Let \( Q_k^i = \sum_{i=1}^{N} (q_{k+1}^i - q_k^i) \), and \( owd_k \) be the One Way Delay of the \( k \)th probe, from (6) and (7) we have:

\[
owd_{k+1} - owd_k = \frac{s}{C_j} + W(T_k, T_{k+1}) + Q_k^i - (t_{k+1} - t_k). \tag{8}
\]

Previous PGM methods all assume a single tight link, and do not take account for the influence of cross traffic of other congestible links on the spacing of probing packets. If the traffic load in non-bottleneck link is low, this would not be an issue. However, if the influence of cross traffic in non-bottleneck link is comparable with that of cross traffic in bottleneck link, the performance of these methods degrades drastically. We now take the equations developed above to derive the following deterministic model which can be used in estimating available bandwidth.

**C. A Deterministic Model**

The single hop scenarios can be viewed as a special case of multi-hop scenarios, and we only consider multi-hop cases here. Assume that we get \( M \) of multi-hop scenarios, and we only consider multi-hop cases.

\[
\langle owd, t, q \rangle = \left\{ \begin{array}{ll}
\text{owd}_1, & \text{if } (t, q) \text{ are within the same busy period}, \\
\text{owd}_1 + \frac{(M-1)s}{C_j}, & \text{otherwise}.
\end{array} \right.
\]

Generally it is nearly impossible to develop a probing technique that can rigidly satisfy the conditions mentioned above. But we find an approach that can approximate the goal very closely. We describe this approach in the following section in detail.

**IV. PATHCOS++**

PATHCOS++ estimates the available bandwidth along a path by sending a train of time stamped probing packets from sender to receiver. The receiver monitors the changes in One Way Delay (OWD) of the probing packets and conduct analysis based on them. Like previous PRM methods, PATHCOS++ also takes the concept of self-induced congestion. However, previous PRM methods only use information about whether delay is increasing or not, while PATHCOS++ can fully exploit the information contained in packet delay signatures.

**A. Sending probing packets**

Consider a train consisting of \( N \) probing packets, each with size \( s \) bits. Assume that the transmission time of the \( k \)th packet is \( t_k \). The instantaneous rate of packet \( k \) is \( r_k = \frac{s}{t_k} \). PATHCOS++ uses a cos function to control the interspacing time:

\[
\Delta_k = \frac{s}{B + Acos(\frac{k\pi}{N})}. \tag{12}
\]

Accordingly, \([B - A, B + A]\) is the range of available bandwidth PATHCOS++ can estimate. The left part of Fig. 1 is an example which shows one probing train sent by PATHCOS++. Usually a complete estimation of available bandwidth takes several such probing trains. The probing rate in each train is decreasing. The higher probing rate is used for self-induced congestions (let the probing packet fill up the bottleneck queues) and the lower rate is used to let the queues relax. PATHCOS++ takes the feedback of receiver and adaptively adjust \( A \) and \( B \). \( B \) should be set lower than the available bandwidth so that the probes have enough time to relax from congestion state. In current implementation...
PATHCOS++ sets $B$ as $0.8 \times \text{estimated available bandwidth}$ and $A = 0.95 \times B$.

B. Estimating the Available Bandwidth

Fig. 1. illustrates a probing train and its response curve. This example is typical and is taken from our experiments (topology shown in Fig. 7). The response curves of probing trains may be smooth or jagged due to different burstiness of cross traffics, but the general shapes are similar: the One Way Delay (OWD) would first show an increasing trend and then a decreasing trend. This is caused by the probing pattern of PATHCOS++. When the arrival rate of probing packet and the cross traffic is higher than the service rate of bottleneck link, the queue would first show an increasing trend and then a decreasing trend. When the probing rate goes lower than the available bandwidth, the queue starts to “relax”, leading to a decreasing trend of OWD. We now focus on the big bumps (shown in Fig. 1) of probing response curves and ignore the small oscillations. Note that when the OWD of probing packets is higher than average, it means that the packets have encountered congestion, and the bottleneck link would be very likely to be busy working.

For single hop scenarios, because
$$\sum_{i=1}^{j-1} (q^i_M - q^i_1) = \sum_{i=j+1}^{N} (q^i_M - q^i_1) = 0,$$
we can select any segment of the big bump of response curve as long as the first $\text{owd}$ equals to the last one. Usually probing packets belonging to the upper part of the big bump is more favorable, because compared with those at the bottom of the big bump, these packets would be more likely to belong to the same busy period. But it is also not good to only use top part of big bumps because this would waste a lot of probing samples. The algorithm described later can attain the desirable trade-off between accuracy and efficiency.

When it comes to multi-hop scenarios, the situation becomes complex, especially when there are multiple congestible links. Supposing that we choose a set of samples belonging to the big bump, e.g. $(t_1, t'_1, t_2, t'_2), (t_2, t'_2, t_3, t'_3), \ldots, (t^i_M, t^i_{M+1}, t^i_M, t^i_M)$, then even if $\text{owd}_1 = \text{owd}_M$, the differences between the queuing time of the first and the $M$th probing packet in other links may be non-negligible, e.g.
$$\sum_{i=1}^{j-1} (q^i_M - q^i_1) = \delta_1 \text{ and } \sum_{i=j+1}^{N} (q^i_M - q^i_1) = \delta_2,$$
where $\delta_1$ and $\delta_2$ are two positive numbers that are not small enough and can induce noticeable bias if (11) is used to estimate the available bandwidth.

From long experiments we find that the “starting point” and “ending point” of “big bumps” (shown in Fig. 1) would usually correspond to similar summation of queuing time in intermediate links, we approximately get
$$\sum_{i=1}^{j-1} (q^i_M - q^i_1) \approx 0 \text{ and } \sum_{i=j+1}^{N} (q^i_M - q^i_1) \approx 0.$$ The pseudo code of algorithm which finds the “big bump” is shown in Fig. 3 (from line 1 to line 15).

The “big bump”, however, still cannot be directly used for bandwidth estimation, because there are some special cases that must be handled with care.

1) $A + B$ is too low. This may be due to inappropriate initialization of $A$ and $B$ or a sudden change in the available bandwidth. In this case the probing train is insufficient to induce congestion in the bottleneck link and the OWDs of probing packets would only be a set of random samples of queuing delay (shown in the left part of Fig. 2). To deal with this problem we take a threshold $\text{len}_{\text{tsh}}$. Only when the the length of “big bump” exceeds this threshold will it be used for calculation.

2) $A + B$ is too high. In such cases the tail of the probing train cannot witness a relaxed queue (shown in the right part of Fig. 2). To deal with this problem we use threshold $\text{height}_{\text{tsh}}$. Only when “big bumps” tall enough will they be considered as valid (algorithm pseudo code: Fig. 3, lines 16-18).

The algorithm PATHCOS++ takes to estimate the available bandwidth is shown in Fig. 3. In current implementation of PATHCOS++ the threshold $\text{len}_{\text{tsh}}$ is set as $\frac{N}{2}$ and the threshold $\text{height}_{\text{tsh}}$ is set as 3. The threshold values are choosen because they work very well in experiments.

Fig. 1. A Probing Train and its Response Curve

Fig. 2. Examples of Response Curves with Inappropriate A and B

408
Algorithm PATHCOS++

Input: $owd_0, ..., owd_{N-1}, t_0, ..., t_{N-1}, len_tsh, begin_tsh$
\{ $owd_i$ one way delay of the $i$th probe, $t_i$ sent time of the $i$th probing packet. \}
Output: available bandwidth, or -1 if $A + B$ is too high and -2 if $A + B$ is too low.
1: $maxlen ← 0$; $sp ← 0$; $ep ← 0$
2: for $i = 0$ to $N − maxlen$ do
3: $n ← 0$; $j ← i$
4: while $j + 1 ≤ N − 1$ and $owd_{j+1} > owd_i$ do
5: $j ← j + 1$; $n ← n + 1$
6: end while
7: if $j + 1 ≤ b$ and $owd_{j+1} ≤ owd_i$ then
8: if $|owd_i − owd_{j+1}| < |owd_i − owd_j|$ then
9: $n ← n + 1$; $j ← j + 1$
10: end if
11: if $(i − sp) = 3$ and $(i − sp < ep − j − 1)$ \{discard packets likely to be in different busy periods \} or $(n > maxlen)$ \{bigger bump \} then
12: $maxlen ← n$; $sp ← i$; $ep ← j$
13: end if
14: end if
15: end for
16: if $owd_p ≠ owd_0$ and \[
\frac{|\max(owd_0, ..., owd_{N-1}) − owd_p|}{owd_p − owd_0} < \frac{height_tsh}{len_tsh}
\] then
17: return -1; \{ $A + B$ too high \}
18: end if
19: if $ep − sp > len_tsh$ then
20: return $s × (ep − sp)/(t_{ep} − t_{sp})$;
21: else
22: return -2; \{ $A + B$ too low \}
23: end if

Fig. 3. The PATHCOS++ algorithm

V. EXPERIMENTS AND RESULTS

We take NS2[32] simulations in order to evaluate PATHCOS++ comprehensively in a controllable and reproducible environment. We compare PATHCOS++ with Spruce and pathChirp, which are prominent representatives of state-of-art PGM and PRM techniques.

It should be noted that most of previous works evaluate the accuracy of available bandwidth estimation tools by comparing the estimated values with the minimal averaged residual bandwidth of each link during some same time period, and ignore the time shift caused by packet queuing or transmission delay in non-bottleneck links. Such methods are intuitive and would not cause any problem when the cross traffic is a time invariant and stationary process. However, if the cross traffic is bursty, the evaluation results can be affected by the measurement timescale, sampling intensity, delay in each hop and RTT, and such methods cannot correctly judge the accuracy of the inference logic of available bandwidth estimation tools. We have observed from experimentation that the bias induced by such evaluation may be even larger than the estimation error of available bandwidth estimation techniques.\(^4\)

In order the address this problem, we take a different approach for evaluation. Consider the following example. Assume that a path consisting of $N$ links $L_1, L_2, ..., L_N$ is probed, and a train/pair of probing packets with sequence number $a, a + 1, ..., a + m$ is used to estimate the available bandwidth. Probing packets $a$ and $a + m$ arrives at link $L_i$ at time $t^i_a$ and $t^i_{a+m}$, respectively. We then calculate the residual bandwidth of link $L_i$ during interval $[t^i_a, t^i_{a+m}]$ (denoted as $R_i(t^i_a, t^i_{a+m})$) by exactly counting the amount of cross traffic that arrives at $L_i$ during this time interval. We write a program with ANSI C language to analyze the output traces generated by NS2 and do this work. The value of available bandwidth used for evaluation during this probing period is calculated by

$$AB^a_{a+m} = \min_{i=1, ..., N} \left( R_i(t^i_a, t^i_{a+m}) \right)$$ \hspace{1cm} (13)

Because all probing techniques can only estimate the minimal residual bandwidth when the probing packets traverses the intermediate links, (13) can be used to evaluate the inference logic of these techniques without bias. Also note that by using this approach the evaluation result is not affected by sampling rates as well as the process of cross traffic.

A. Single Congestible Link Scenario

We first evaluate PATHCOS++ using the topology shown in Fig. 4. The cross traffic between nodes N2 and N3 is set as trace reproduced(shown in TABLE I)., Poisson and CBR(average rate 500Mbps). The capacity of $L_1$ is set as 1Gbps. We use the default configuration of pathChirp and Spruce. For PATHCOS++, $N$ is set to 100. The size of probing packets are set as 1K bytes. All of the tools take the same average probing rate. For each tool we run the simulation

\(^4\)Another example is that the experimental results of [24] show that Spruce has more estimation errors than pathload in single-hop scenarios, but in fact according to the theoretical analysis of Xiliang Liu[21]. Spruce is accurate in single-hop scenarios.

\(^5\)These traces have also been used in [24].
for 100 seconds and calculate the Relative Estimation Error (REE), which is defined as

$$REE = \frac{|Estimated\ Value - Real\ Value|}{Bottleneck\ Link\ Capacity}. \quad (14)$$

The result is shown in TABLE II. Each group of values in the table shows the average, the 5- and 95-percentiles of REE, from left to right, respectively. We observe that Spruce is the most accurate, which is consistent with our analysis; the accuracy of pathChirp is determined by the characteristic of cross traffic, and the relative error of PATHCOS++ is very low for all types of cross traffic.

### B. Multiple Congestible Links

1) **Different Traffic Load on Non-bottleneck Links:** Most state-of-art probing techniques implicitly assume there is only one tight link in the path. Previous works has suggested that available bandwidth estimation tools tend to underestimate the available bandwidth if the path consists of multiple congestible links[23], [24]. We now evaluate the performance of PATHCOS++ with different traffic load on non-bottleneck links. We choose poisson cross traffic which follows “per-hop persistent” pattern. The topology is shown in Fig. 5. The link $L_3$ is the bottleneck link. The capacities of all links are set as 10Mbps. The rate of cross traffic $CT_3$ in $L_3$ is set as 6Mbps, and the rate of cross traffic on other links($CT_1, CT_2, CT_3$, and $CT_5$) increases from 0.1Mbps to 6Mbps. The sizes of cross traffic packet are randomly distributed among 40 B, 550 B and 1500 B. For each situation we run each tool for 300 seconds and take the average estimation value. The results are shown in the right part of Fig. 6. We can observe that the both Spruce and pathChirp tend to underestimate the available bandwidth as the traffic load on non-bottleneck links increases, while the accuracy of PATHCOS++ is nearly unaffected.

It is interesting to notice that when the rate of non-bottleneck cross traffic goes near 6Mbps, PATHCOS++ also gives a slightly conservative estimation. However, this is not caused by the inference logic of PATHCOS++. In fact, when we take (13) to calculate the available bandwidth during each probing period, we find that PATHCOS++ is still accurate. This is because poisson traffic is dynamic, and the residual bandwidth of non-bottleneck link may become lower than 4Mbps when their traffic load goes near 6Mbps. Because the probing technique only gives the minimal per-link residual bandwidth when it traverses each link, the possibility that a probing train encounters at least one link with residual bandwidth lower than 4Mbps increases as the traffic load of non-bottleneck goes near 6Mbps, thus the averaged estimation value will be lower than 4Mbps.

2) **Different Traffic Load on Bottleneck Link:** Previous studies suggest that different path tightness in bottleneck link may also influence the accuracy of bandwidth estimation tools. We evaluate the performance of PATHCOS++ with bottleneck link utilization ranging from 10% to 90%. We increase the cross traffic $CT_3$ from 1Mbps to 9Mbps and the cross traffic rate in other links are set as $\frac{CT_3}{2}$. The results are shown in the left part of Fig. 6. We can observe that PATHCOS++ is not affected by different utilization in bottleneck links, and is more accurate than pathChirp and Spruce.

3) **A More General Case:** We now evaluate the performance of PATHCOS++ under a more general scenario. The topology used is shown in Fig. 7. We reproduce those real Internet traces to generate the cross traffic. Trace UNC05 is used for $CT_1$, IPLS-CLEV for $CT_2$, IPLS-KSCY for $CT_3$, UNC28 for $CT_5$ and Ibiblio for $CT_3$. We run the simulation for 60 seconds and the results are shown in Fig. 8. The average, 5- and 95-percentiles of REE are shown in TABLE III.

Again, we found that PATHCOS++ is the most accurate.
pathChirp gives conservative estimates and Spruce is the most inaccurate.

VI. CONCLUSION

In this paper we presented PATHCOS++, a novel active probing technique that can estimate the available bandwidth along an end-to-end path dynamically. PATHCOS++ integrates the advantages of PRM and PGM probing techniques and fully exploits the information contained in the probing delay signatures. Our simulation results show that PATHCOS++ outperforms existing probing techniques in terms of accuracy, especially in multi-hop scenarios, while does not induce more overhead on network.

We are planning to refine the model to further improve the accuracy and reduce the overhead of PATHCOS++ while at the same time taking into account of the other factors such as packet loss and reordering, and test PATHCOS++ in real network environments. A recent study has suggested that all current probing based techniques only estimate the fair share of link capacity, rather than the available bandwidth when they are applied to CSMA/CA based wireless links[28]. We are planning to adapt PATHCOS++ to such scenarios.

REFERENCES