Estimating Available Bandwidth Using Packet Pair Probing

Ningning Hu, Peter Steenkiste

September 9, 2002
CMU-CS-02-166

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

The packet pair mechanism has been shown to be a reliable method to measure the bottleneck link bandwidth of a network path. However, the use of packet pairs to measure available bandwidth has had more mixed results. In this paper we study how packet pairs and packet trains can be used to estimate the available bandwidth on a network path. As a starting point for our study, we construct the gap model, a simple model that captures the relationship between the competing traffic and the input and output packet pair gap for a single hop network. We validate the model using measurements on a testbed. The gap model shows that the initial probing gap is a critical parameter when using packet pairs to estimate available bandwidth. Based on this insight, we propose a new technique to measure the available bandwidth – the Initial Gap Increasing (*IGI*) algorithm, which experimentally determines the best initial gap for measuring available bandwidth. Our experiments show that measurements that take 4-6 round trip times allow us to estimate the available bandwidth to within about 10%.

This research was sponsored in part by the Defense Advanced Research Project Agency under contracts F30602-99-1-0518, monitored by AFRL/IFGA, Rome NY 13441-4505, and F30602-96-1-0287, monitored by Rome Laboratory, Air Force Materiel Command, USAF.

Keywords: Network measurements, available bandwidth, packet pair, probing

1 Introduction

Characterizing the end-to-end network throughput that users can expect is a problem that is both intellectually intriguing and of practical importance. However, the scale of the Internet, traffic volume, and diversity of technologies make this a very challenging task. Furthermore, regular Internet users do not have access to network internals, adding to the complexity of understanding, characterizing, and modeling the performance of the Internet. While the problems of characterizing end-to-end latency and bottleneck link capacity have received a lot of attention [14] [8] [9] [11] [6] [15], the path property that is probably of most interest to applications is the throughput an application can expect on a path through the network. In this paper, we focus on measuring the *available bandwidth*, which we define as the difference between the link capacity and the competing traffic volume on the link that limits end-to-end throughput.

Techniques for estimating available bandwidth can be classified into two categories: passive measurement [20] [21] and active probing [14] [8] [9] [11] [6] [15]. Passive measurement tools use the trace history of existing data transfers. While potentially very efficient and accurate, their scope is limited to network paths that have recently carried user traffic. Active probing, on the other hand, can explore the entire network. Among the active probing techniques, packet pair techniques have proven to be the most successful. The basic idea is that the sender injects a pair of packets into the network to the destination of interest; the destination host then sends an echo for each packet. By measuring the changes in the spacing between the two packets, the sender can estimate bandwidth properties of the network path. While the packet pair mechanism has been shown to be a reliable method to measure the bottleneck link capacity on a path through the network [6] [8] [14], the use of packet pairs to measure the available bandwidth has had more mixed results.

Let us first define the terms *available bandwidth*, *bottleneck link*, and *tight link* more precisely. Assume an end-to-end path includes n links $L_1, L_2, ..., L_n$, their capacities are $B_1, B_2, ..., B_n$, and the traffic loads on these links are $C_1, C_2, ..., C_n$. We define the *bottleneck link* as $L_b(1 \leq b \leq n)$, where $B_b = min(B_1, B_2, ..., B_n)$, and the *tight link* as $L_t(1 \leq t \leq n)$, where $B_t - C_t = min(B_1 - C_1, B_2 - C_2, ..., B_n - C_n)$. The unused bandwidth on the tight link, $B_t - C_t$, is also the *available bandwidth* of the path. In the first part of the paper we assume that the tight link is a bottleneck link; we consider the case where the two are different in Section 6. Note that applications may not be able to fully utilize the available path bandwidth since other factors (the application, TCP (mis)configuration, ...) can limit throughput.

This paper has two contributions. First, we develop a *gap model* that captures the relationship between the competing traffic on the bottleneck link and the change in the gap of a packet pair probe in a single-hop network. We use this gap model to identify under what conditions changes in the packet pair gap can be used to accurately characterize the competing traffic on the bottleneck link. Next, we develop a new packet pair technique, called the *Initial Gap Increasing (IGI)* algorithm, to characterize the available bandwidth on a path through the network. This is done by experimentally determining the input packet pair gap that gives a high correlation between changes in the packet gap and the competing traffic on the bottleneck link. We evaluate the *IGI* algorithm using both simulations and experiments.

This paper is organized as follows. In Sections 3 we introduce the gap model for single hop networks. We use measurements on a controlled local testbed to validate our analysis. The *IGI* algorithm is introduced in Section 4 and evaluated in Section 5. In Section 6 we study how non-bottleneck links can affect the gap. We first discuss related work.

2 Related Work

The problem of characterizing the bottleneck link bandwidth using active probing is reasonably well understood. The work in [7] classifies the tools into single packet methods and packet pair methods, according to how probing packets are sent. Single packet methods measure link capacity by measuring the time difference between the round trip time from one end of the link to the other. This requires a large numbers of probing packets to filter out the effect of other factors that affect delay such as queueing delay. Tools using this technique include pathchar [11], clink [9] and pchar [15].

The idea behind packet pair techniques is to send groups of back-to-back packets, i.e., packet pairs, to a server which echos them back to the sender. As pointed out in earlier research on TCP dynamics [10], the spacing between the packet pairs is determined by the bottleneck link and is preserved by higher-bandwidth links. Example tools include NetDyn probes [4], packet pairs [13], bprobe [5] [6], and nettimer [14]. Most of these tools use statistical methods to estimate the bandwidth, based on the assumption that the most common value for the packet pair gap is the one that captures the bottleneck link transmission delay. In practice, the interactions between a packet pair and the network traffic on different links can be very complex, and it is not always appropriate to use the most common packet pair gap value [19]. Recent work on Pathrate [8] addresses these problems by explicitly analyzing the multi-model nature of the packet gap distribution.

Characterizing the available bandwidth is more difficult since it is a dynamic property. This means that the available bandwidth must be averaged over a reasonable time interval, so packet pair techniques often use packet trains, i.e., longer sequences of packets. A typical example of an active measurement tool for available bandwidth is PBM (Packet Bunch Mode) [19]. It extends the packet pair technique by using different-sized groups of back-to-back packets. If routers in the network implement fair queueing, the bandwidth indicated by the back-to-back packet probes is an accurate estimate of the “fair share” of the bottleneck link’s bandwidth [13]. Another example, cprobe [6], sends a short sequence of echo packets between two hosts. By assuming that “almost-fair” queueing occurs during the short packet sequence, cprobe provides an estimate for the available bandwidth along the path between the hosts. Treno [16] uses flow control and congestion control algorithms similar to those used by TCP to estimate available bandwidth. The work in [8] mentions a technique for estimating the available bandwidth based on the Asymptotic Dispersion Rate (ADR). Part of our work is related to ADR and we share the view that the ADR is the effect of all the competing sources along the transmission path. However, we also identify the initial input probing packet gap as a critical parameter that need dynamically determined during probing.

The Pathload [12] tool proposes to characterize the relationship between probing speed and available bandwidth by measuring the one way delay of probing packets. By trying different probing speeds, a reasonable estimate for the available bandwidth can be found. The work closest to ours is the TOPP method [18]. This method provides a theoretical model for the relationship between available bandwidth and probing packet spacing at both end points. Simulation results are used to validate the method. Both of these methods analyze the relationship between probing trains and available bandwidth, but their analysis does not capture the fine-grain interactions between probes and competing bandwidth, which is useful to, for example, understand the limitations of the techniques.

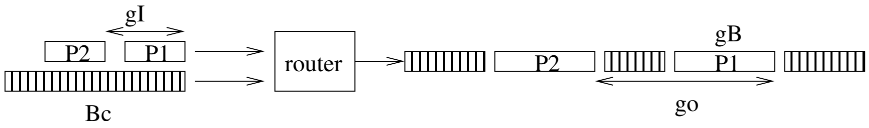

Figure 1: **Interleaving of competing traffic and probing packets.**. *g_I is the input gap, g_B is the probing packet length on the output link, g_O is the gap after interleaving with the competing traffic, B_C is the competing traffic throughput.*

3 Single Hop Gap Model

The idea behind using packet pairs to measure available bandwidth is to have the probing host send a pair of packets in quick succession and to measure how the gap between the two packets changes as a result of traversing the network (Figure 1). As the packets travel through the network, packets belonging to competing traffic will be inserted between them, thus increase the gap. As a result, the gap value at the destination host will be a function of the competing traffic rate, making it possible to estimate the amount of competing traffic. Unfortunately, how competing traffic affects the gap of a packet pair is much more complex than this description suggests. In the remainder of this section, we describe and evaluate a simple model that captures the relationship between the gap value and the competing traffic throughput for a single-hop network.

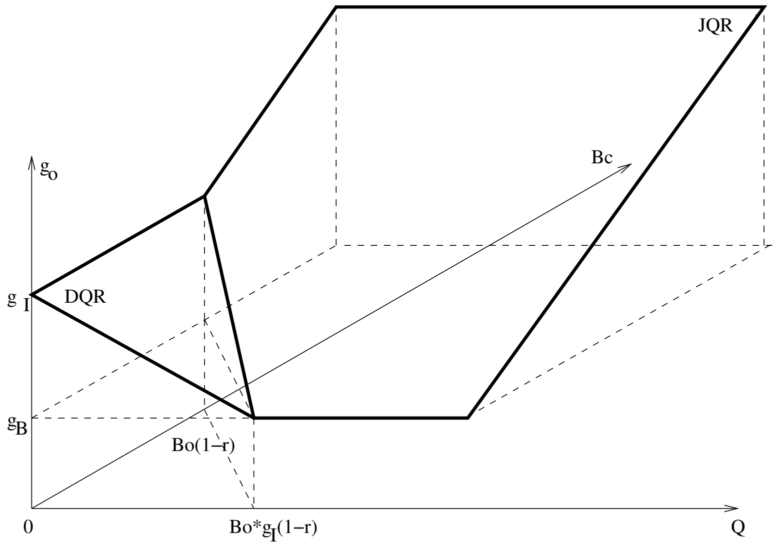

Figure 2: **Single hop gap model**. *The output gap g_O changes in two different ways: in the DQR region, g_O is not affected by B_C, while in the JQR region, g_O is proportional to B_C.*

3.1 Single Hop Gap Model

The gap model models the value of the output gap g_O, i.e. the time between when the first bit of P1 and the first bit of P2 leave the router. The 3-D figure in Figure 2 shows the value of g_O as a function of Q, the output queue size when packet P1 arrives at the router, and B_C, the competing traffic for the time interval between the arrival of packets P1 and P2. g_I is the gap value when P1 and P2 enter the router, and we call it *input gap*. Note that g_I includes P1's transmission delay [1] on the input link. g_B is the transmission delay of the probing packet on the output link. Since g_B is also the gap value of two back-to-back probing packets on the bottleneck link, we call g_B the *bottleneck gap*. Note that when the link capacity B_O, the probing packet length, and g_I are fixed, g_B and r (defined as $r = g_B/g_I$) in Figure 2 are constant.

The model in Figure 2 assumes that we use FIFO queueing and that all probing packets have the same size. It also assumes that the competing traffic is constant between the arrival of packets P1 and P2; given that this input value is of the order of 1 msec, this is a reasonable assumption. We observe that the model has two regions. As we show below, the regions correspond to cases when the two packets P1 and P2 do or do not fall in the same queueing period. Queueing period is defined to be a time segment during which the queue is not empty; different queue periods are separated by a period when queue is empty. For this reason, we call the regions the *Disjoint Queuing Region (DQR)* and the *Joint Queuing Region (JQR)*.

In the *DQR* region, P2 will find an empty queue; this is directly result of the fact that P1 and P2 are in separate queueing periods and of our assumption that the competing traffic is constant for the P1-P2 period. In this situation, the output gap will be the input gap minus the queueing delay experienced by P1, i.e.

$$g_O = g_I - Q/B_O$$

Under what condition will the queue be empty when P2 arrives? For that to happen, the router needs to complete three tasks before P2 arrives: process the queue Q (Q/B_O), process P1 (g_B), and process the competing traffic that arrives between the probing packets ($B_C \cdot g_I/B_O$). The router has g_I time to complete these three operations, so the condition for operating in region *DQR* is $Q/B_O + B_C \cdot g_I/B_O + g_B < g_I$, which is the triangular region labeled *DQR* in Figure 2. In this region, the output gap g_O does not depend on the competing traffic throughput B_C.

Under all the other conditions, when P2 arrives at the router the queue will not be empty, i.e. P1 and P2 are in the same queueing period. This corresponds to the *JQR* region and the output gap consists of two time segments: the time to process P1 (g_B) and the time to process the competing traffic that arrives between the two probing packets ($B_C \cdot g_I/B_O$). So in this region, the output gap will be

$$g_O = g_B + B_C \cdot g_I/B_O$$

That is, in this region, the output gap g_O is a function of the competing traffic throughput B_C.

This model directly points at a first problem in using packet pairs for estimating the competing traffic rates. If the packet pair happens to operate in the *DQR* region of the bottleneck router, the output will bear no relationship with the competing traffic and a user using the *JQR* equation (since the user does not know what region applies) will yield an arbitrary result. Furthermore, the estimate of the competing traffic obtained using a single packet pair will provide the average

[1] Transmission delay is defined as the time for a packet to be placed on a link by a sender.

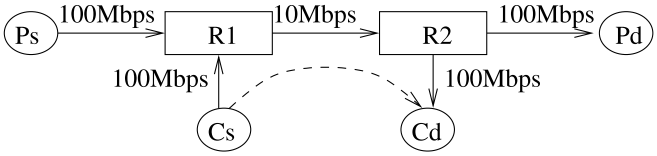

Figure 3: **Testbed configuration for single hop network experiment.**

competing traffic over g_I, which is a very short period. Since the competing traffic is likely to fluctuate, one will want to average the results of multiple samples, corresponding to independent packet pairs, but this increases the chance that some of the samples will fall in the *DQR* region.

Note that these conclusions do not apply to packet train methods [8] [19], since the "pairs" that make up the a packet train are not independent. We discuss this further in Section 4.1.

3.2 Testbed Experiment

To illustrate the gap model, we ran experiments on an isolated testbed. The testbed topology is shown in Figure 3. Ps and Pd are the probing source and destination, while Cs and Cd are used to generate competing TCP traffic. R1 and R2 are FreeBSD-based routers that run *tcpdump* on all relevant interfaces to record the packet timestamp information. We send out a probing train consisting of a series of evenly spaced 100B packets; each consecutive pair of packets in the train can serve as a probing pair. The competing traffic is generated using *Iperf* [22], which allows us to simulate typical user traffic such as FTP traffic. We can also control the competing traffic throughput by adjusting the TCP window size, which is useful in illustrating the features of the gap model.

3.2.1 Effect of *JQR*: capturing competing traffic

In this experiment, we use 1024 probing packets [2] and the input gap is set to $0.31ms$. We use a competing load of 7.2Mbps. A typical set of experimental results is shown in Figure 4; the top graph shows the output gap measured on R1 and the bottom graph shows the corresponding output gaps measured on R2. The increase in gap value on router R1 is caused by competing traffic on the bottleneck link.

The increased gap values in the top graph of Figure 4 fall into three clusters: $1.2ms$ (the time to transmit a 1500B TCP competing packet on a 10Mbps link), $2.3ms$, and $3.8ms$. This is because these probing pairs are interleaved by exactly 1, 2, and 3 competing packets. The fact that at most 3 packets are inserted in a probing gap should not be a surprise since the input gap is $0.31ms$ (note that the input links are 10 times faster than the bottleneck link). Besides the increased gap values, we also have some decreased gaps. Most of them are $0.08ms$, which is the transmission delay of the probing packet (100B) on the 10Mbps bottleneck link. The bottom graph shows that the increased gap values are maintained through router R2, because R2 has a higher output rate than input rate.

[2] We use such a large number in order to get a large enough probing period, but we make sure it does not cause packet drops so as not to affect the competing flow's throughput.

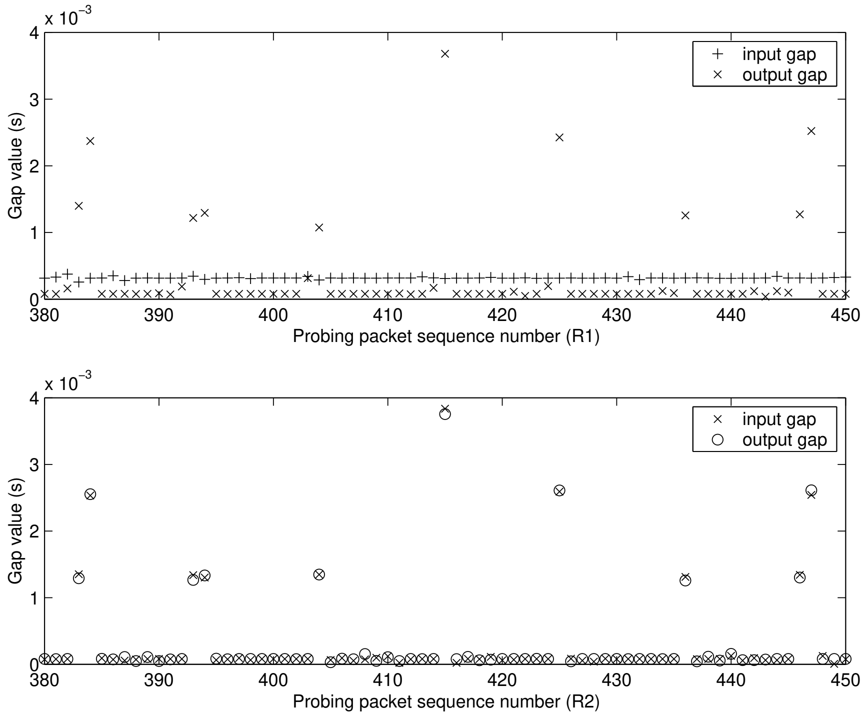

Figure 4: **Effect of** *JQR*. *Input and output gap for routers R1 (top) and R2 (bottom).*

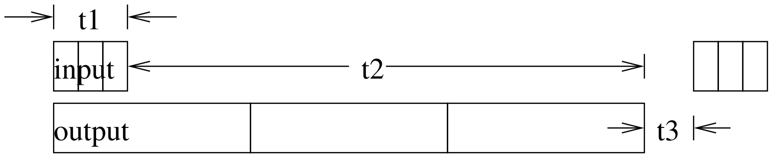

Figure 5: **Competing packets at the input and output interface of R1**. t_1 *is the transmission delay of the 3 competing packets on R1's input link,* $t_2 + t_1$ *is their transmission delay on the output link,* t_3 *is the interval between the time when the first 3 packets finish transmission and the time when the second 3 packets arrive.*

The observed changes in the gap values are a result of the burstiness of the competing traffic. In some cases, as is shown in Figure 5, the source Cs sends out three 1500B packets back-to-back (t_1 period in Figure 5). This builds up the queue in router R1 and the queue will drain during the period t_2. After period t_3, more competing traffic arrives. A packet pair that overlaps with period t_1 will see an increased gap, corresponding to 1, 2, or 3 competing packets. A packet pair that falls in period t_2 will see its gap reduced to 0.08ms. Packet pairs can also straddle the t_2 and t_3 periods, in which case their gap values reduce to values between g_B(0.08ms) and g_I (0.31ms). This effect corresponds to the *DQR* region, and in this experiment it is not very common.

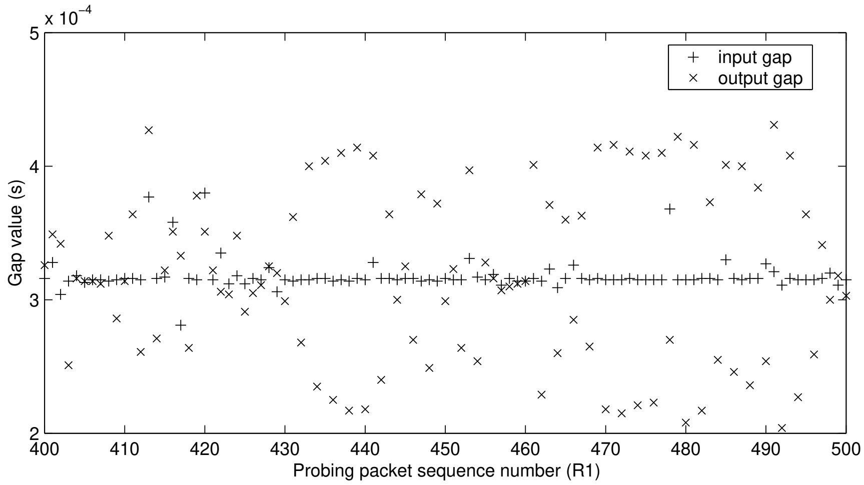

Figure 6: **Effect of** *DQR*. *The output gaps measured on router R1.*

3.2.2 Effect of *DQR*: losing competing traffic

We now change the competing traffic by setting the source TCP window size to 512B and the destination TCP window size to 128B. This forces the competing source to send one 128B packet each round trip time, i.e., the competing traffic is a low-bandwidth CBR load.

The results in Figure 6 show that the increased gap values no longer cluster around a small set of discrete values. Instead, the output gap values take on a set of values fairly evenly distributed between about 2 and 4.3 msec. To explain this, we show a detailed snapshot of the starting and ending time of 2 competing packets (A and B) and 6 probing packets (1-6) in Figure 7 for both the input and output interfaces of router R1. The lines show the transmission delays of the packets [3]. Given the nature of the competing traffic, probing packets will always encounter an empty or very short queue. In some cases we see a gap increase because P2 is delayed (e.g. the gap for pair (2, 3) increases to $0.414ms$). In other cases P1 is delayed, and we end up in region *DQR* (e.g. pair (3, 4)). The delays experience by P1 or P2 correspond to the transmission time of a partial packet so they can take on any value.

3.3 Discussion

The single hop gap model and our experiments show the challenges associated with using packet pairs to estimate competing traffic on the bottleneck link. To what degree the measured gap on the destination reflects the competing traffic load depends on what region the bottleneck router is operating in. The good news is that when we are operating in the right region, there is a proportional relationship between the output gap and the competing traffic. In the next section we will use this relationship to develop a new method for measuring the competing traffic. The bad news is that the

[3]Because the time stamp recorded by tcpdump is the time the last bit of a packet passes through the network interface, for the segments in Figure 7, only the right end points are raw trace data. The left end points are calculated using packet length and the corresponding interface's transmission rate. The small overlap between some of the packets, such as "3" and "A", is not possible and is probably due to the timing error of tcpdump.

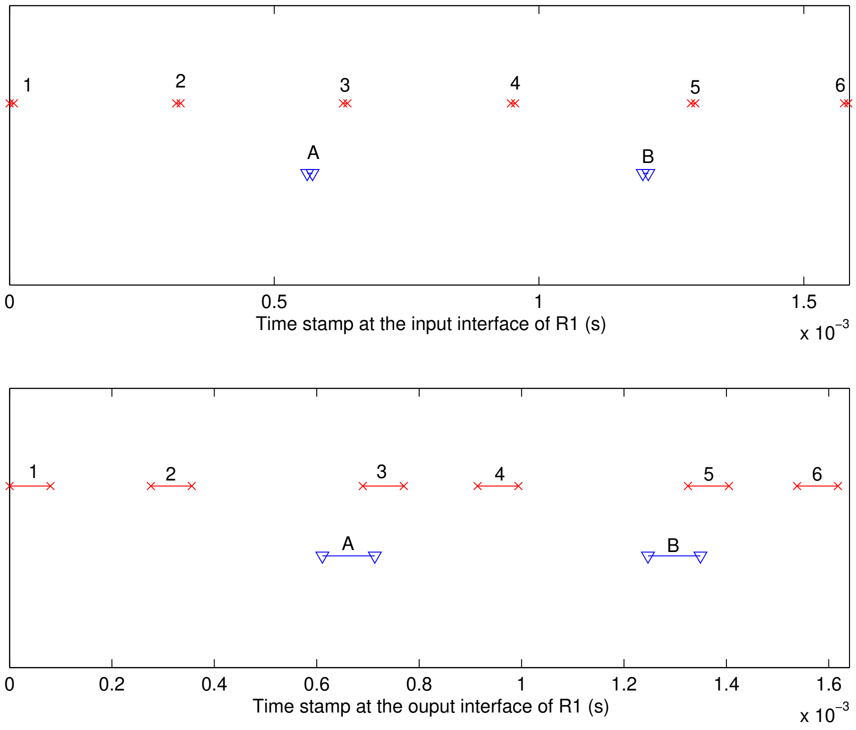

Figure 7: **Snapshot of the interleaving between two competing packets and six probing packets**. *The top figure is the trace from the R1's input interfaces, and the bottom figure is from R1's output interface.*

DQR region can introduce errors, so it is important to design the bandwidth probing experiments to avoid the *DQR* region.

4 Estimating Available Bandwidth

We introduce a new method for estimating the competing traffic on the bottleneck link.

4.1 The IGI formula

The experiments in Section 3.2.1 suggest a way of estimating competing throughput using packet trains instead of individual packet pairs. Note that the *DQR* effect is very small in this experiment, so there is a correlation between the increased gap values and the amount of competing traffic.

We compute the competing traffic throughput as follows. Assume M probing gaps are increased, the values are $G^+ = \{g_i^+ | i = 1, ..., M\}$; K gaps are unchanged and they are represented as $G^= = \{g_i^= | i = 1, ..., K\}$; and N gaps decrease and they are denoted as $G^- = \{g_i^- | i = 1, ..., N\}$. We use B_O to denote the link capacity of <R1, R2>, i.e., the bottleneck link. Then $B_O \sum_{i=1}^{M} (g_i^+ - g_B)$ is the amount of competing traffic that arrive at R1 during the probing period.

As a result, we can estimate the competing bandwidth on the bottleneck link as

$$\frac{B_O \sum_{i=1}^{M} (g_i^+ - g_B)}{\sum_{i=1}^{M} g_i^+ + \sum_{i=1}^{K} g_i^= + \sum_{i=1}^{N} g_i^-} \quad (1)$$

This method of computing competing traffic load will be used by the *IGI* algorithm in Section 4.3, and we call it *IGI* formula. In this formula, the denominator is in general the length of the entire packet train. This is indeed the case when there are no out of order packets. When there are out of order packet, the gap involving the out of order packet are not valid and are excluded, thus reducing the denominator.

For the experiments in Section 3.2.1, the *IGI* formula estimates a competing traffic throughput of 7.3Mbps. This compares well with the throughput of 7.2Mbps reported by *Iperf*. But *IGI* formula only works if we are in region *JQR*. When we apply the *IGI* formula to the experiments of Section 3.2.2, we obtain a throughput of 3.8Mbps, which is much higher than the real throughput of 1.4 Mbps.

4.2 Methodology

A second part of our method is that we must control the measurements so that we use the *JQR* region. There are three parameters that we can control:

1. *Probing packet size*. Experiments with small probing packets are very sensitive to interference. The work in [8] also points out there can be significant post-bottleneck effects for small probing packet size. This argues for sending fairly large probing packets.

2. *The number of probing packets*. It is well known that the Internet traffic is bursty, so a short snapshot is not enough to tell the average traffic load. That argues for sending a fairly large number of probing packets. Note however that sending too many packets can cause queue overflow and packet loss and also increases the network load.

3. *Initial probing gap*. The single hop gap model in Figure 2 shows that increasing the input gap will increase the *DQR* area, where the output gap is independent of the competing traffic. In fact, if $g_I \leq g_B$ the *DQR* area does not even exist. This argues for using small input gaps. However, when $g_I \leq g_B$, we are flooding the bottleneck link, which may cause packet loss and disrupt traffic.

Our experiments show that our results are not very sensitive to the probing packet size and packet number and there is a fairly large range of good values for those two parameters. For example, a 700B packet size and 60 packets per train work well on the Internet.

How to determine a good value for the initial probing gap is much more difficult. To clarify the trade-offs, imagine an experiment in which we send a sequence of packet trains with increasing initial probing gaps. For small initial probing gaps (smaller than g_B, the transmission time on the bottleneck link), we are clearly flooding the network, and measurements will not provide any information on the available bandwidth. When the initial probing gap reaches g_B, the *DQR* region appears in the gap model. Note that, unless the network is idle, we are still flooding the bottleneck link. So far, the average final output gap at the destination will be larger than the initial input gap.

When we continue to increase the initial probing gap, at some point, the average probing rate of the packet train will be equal to the available bandwidth on the bottleneck link. At that point, the final output gap will be equal to the initial input gap. This will continue to be the case as we continue to increase the initial probing gap, since all packets on average will experience the same delay.

We believe that the initial input gap value for which the average final output gap is equal to the initial input gap is the right value to use for estimating the available bandwidth. That is the smallest initial input gap value for which we are not flooding the bottleneck link and hence we can expect to get the most accurate measurements. The *IGI* algorithm is based on this observation.

4.3 *IGI* Algorithm

The IGI algorithm sends a sequence of packet trains with increasing initial packet gap. We monitor the difference between the average output gap and the input gap for each train and use the first train for which the two are equal. This point is called *turning point*. At this point, we use the *IGI* formula to compute the competing bandwidth. The available bandwidth is obtained by subtracting the estimated competing traffic bandwidth from an estimate of the bottleneck link bandwidth. The pseudo code of this algorithm is listed in Figure 8, and we call it *Initial Gap Increasing (IGI)* algorithm. In this algorithm, the bottleneck bandwidth can be measured using, for example, bprobe [6], nettimer [14], or pathrate [8]. Note that any errors in the bottleneck link capacity measurement will also affect the accuracy of the available bandwidth estimate. We will see an example in the next section.

A key step in the *IGI* algorithm is the automatic selection of the turning point. We use the procedure GAP_EQUAL() to do that. It tests whether the source gap "equals" the destination gap, where equality is defined as

$$\frac{|avr\_src\_gap - avr\_dst\_gap|}{\max(avr\_src\_gap, avr\_dst\_gap)} < \delta$$

In our experiments, δ is set to 0.1.

5 Experiment and Analysis

We implemented two versions of the *IGI* tool to accommodate the different network hosts we have access to: a kernel version for hosts on which we can change the operating system and a user-level version that can be run on any host. The kernel version sends out raw TCP packets as probing packets, and the timestamp is measured in the kernel with the help of libpcap [1] [17]. The user-level version uses UDP packets as probing packets, and the timestamp is the time when the probing application on the destination host receives the UDP packets. The kernel version has higher accuracy than the user-level version since it can get more accurate timestamps. Unless specifically mentioned, the measurements are carried out using the kernel version.

5.1 Testbed Experiment

The experiments in this section are used to study the behavior of the *IGI* algorithm. The testbed topology used is the same as in Figure 3.

Algorithm *IGI*:

```
{
    init_gap = SMALLGAP;
    probe_num = PROBENUM;
    packet_size = PACKETSIZE;
    src_gap_sum = probe_num * init_gap;
    dst_gap_sum = 0;
    while (!GAP_EQUAL(dst_gap_sum, src_gap_sum)) {
        init_gap+ = gap_step;
        src_gap_sum = probe_num * init_gap;
        SEND_PROBING_PACKETS(probe_num,
          packet_size, init_gap);
        dst_gap_sum = GET_DST_GAPS();
    }
    inc_gap_sum = GET_INCREASED_GAPS();
    c_bw = b_bw * inc_gap_sum/dst_gap_sum;
    a_bw = b_bw − c_bw;
}
```

Figure 8: *IGI* **algorithm**. SEND_PROBING_PACKETS() sends out $probe\_num$ probing packets of size $packet\_size$ with an input gap of $init\_gap$; GET_DST_GAPS() gets the destination gap values and adds them; GET_INCREASED_GAPS() returns the summary of the gaps that are larger than The bottleneck gap; c_bw, b_bw and a_bw denote the competing traffic bandwidth, the bottleneck link capacity, and the available bandwidth.

Figure 9 shows the probing results using an *Iperf* UDP competing flow of 3.6Mbps. It shows the average gap difference (averaged destination gap minus the averaged source gap) and the competing traffic throughput computed by the *IGI* formula. At the turning point, the initial input gap value is 0.84ms. The estimated competing traffic throughput is 3.2Mbps, which matches with the real value quite well.

In Figure 10 the competing traffic consists of WWW traffic generated by *Surge* [3], which is configured for 4 client processes and 250 user-equivalent threads. The input gap at the turning point is 0.96ms. The estimated competing traffic throughput using the *IGI* formula is 5.6Mbps, which is close to the measured value (5.1Mbps).

For both of the scenarios in Figures 9 and Figure 10, we repeated the experiments by changing the configuration for the competing traffic generation [4]. The results are shown in Figure 11. With *Iperf* CBR UDP competing traffic, *IGI* works very well: in most cases, the measurement error and the standard deviation are very small (less than 5%). The only exception appears for a competing traffic load of 8Mbps, where the error is partly due to probing packet loss. The measurement accuracy with the *Surge* competing traffic is not that good. We believe that the reason is that the *Surge* competing traffic is very bursty, so the instantaneous competing traffic throughput measured

[4]For *Surge*, the different competing traffic loads are created by changing the number of the client threads. In Figure 11(b), the thread counts for experiments 1 through 5 are 10, 20, 60, 80, and 100.

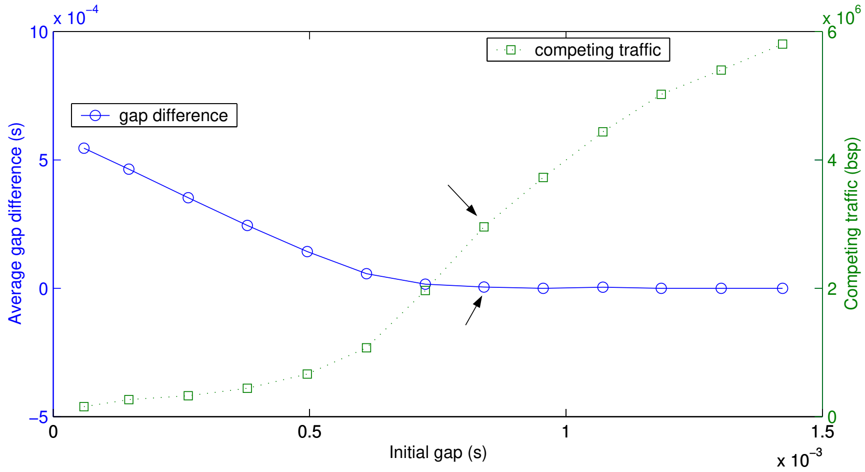

Figure 9: *IGI* **testbed experiment using** *Iperf. Probing number: 256, probing packet: 750B, competing traffic throughput: 3.6Mbps. The arrows point out the measurements at the turning point.*

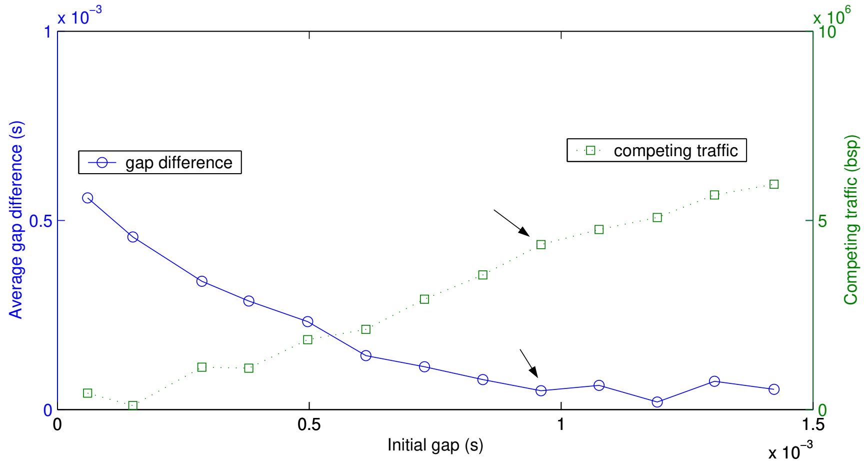

Figure 10: *IGI* **testbed experiment using** *Surge. Probing number: 256, probing packet: 750B, competing traffic throughput: 5.1Mbps. The arrows point out the measurements at the turning point.*

by *IGI* does not always match the long term average throughput reported by *Surge*.

5.2 Internet Experiments

In this section, we study the performance of *IGI* for Internet traffic. For one Internet path, we give a detailed analysis of *IGI*'s performance. Then we present the results of a day-long experiment for three Internet paths with different characteristics.

The first set of Internet experiments is carried out between a Linux machine at CMU which is

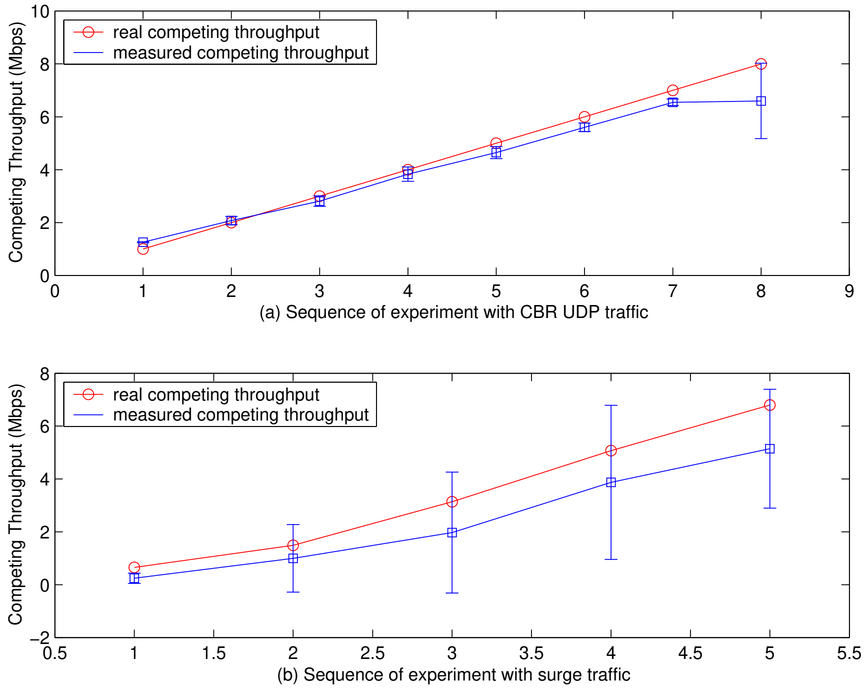

Figure 11: *IGI* **measurement accuracy on testbed.** *The IGI measurements for different competing traffic loads. The competing traffic in the top figure is CBR UDP traffic, generated by Iperf, and the bottom figure shows the results with competing traffic generated by Surge.*

connected to a 100 Mbps Ethernet, and a home machine in Columbus, Ohio, which uses a cable modem (our test shows that its capacity is 10Mbps). The two networks are connected through the regular Internet. For about 27 hours in January 2002, we continuously sent probing trains with 10 different initial input gaps (average values: 0.015ms, 0.158ms, 0.3ms, 0.443ms, 0.585ms, 0.728ms, 0.870ms, 1.012ms, 1.155ms, and 1.299ms). For each input gap, we use four different probing packet sizes (100B, 500B, 750B, 1400B) and each probing train includes 60 probing packets. The experiment sleeps for 30 seconds between consecutive probing trains.

Figure 12 shows a typical set of results. The top graph shows the average gap difference for the different packet sizes. For the 500B and 750B probing packets, there are clear turning points (marked with arrows) at 0.443ms and 0.585ms. The corresponding computed competing traffic throughput (bottom graph) are similar (1.2Mbps and 0.7Mbps), which give us available bandwidths of 8.8Mbps and 9.3Mbps.

For probing packets of 100B, the differences between the input and output gaps are always small. The estimated competing load using the *IGI* method is higher than that using 500B and 750B probes. This problem is probably caused by measurement errors: with small probing packet sizes the turning point gap is very small and hard to measure accurately. For 1400B probing packets, the turning point is very high (1.15ms) and the estimated competing traffic (0.63Mbps) is lower than that obtained using the 500B and 750B probing packets. We think this is a result of the fact that the larger probing packet size corresponds to a more aggressive flow that grabs more of the network bandwidth.

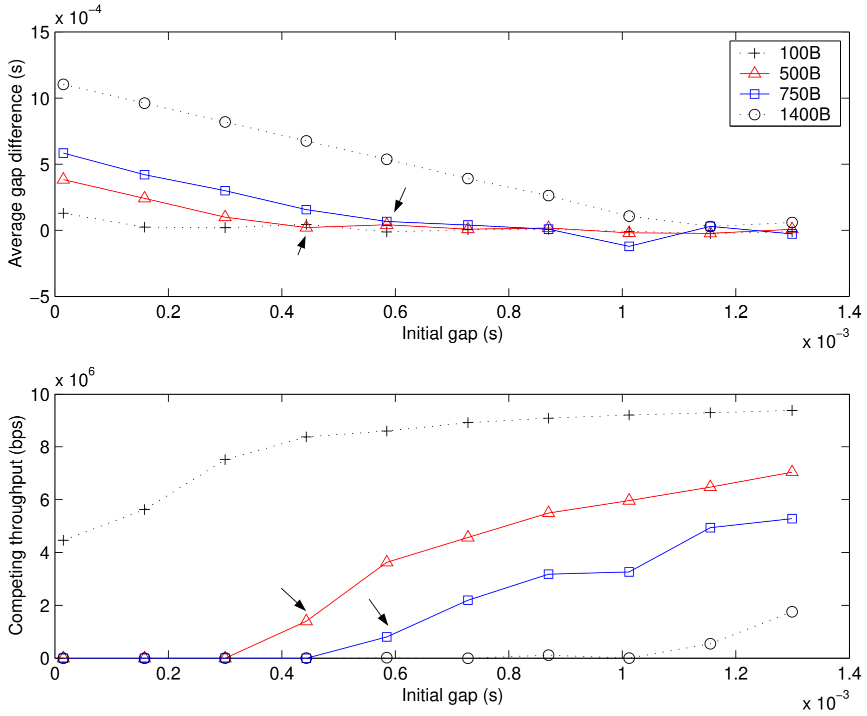

Figure 12: **Internet experiment between CMU and OHIO.** *The experiments use 10 input gap values (x axis) and each probing train includes 60 probing packets.*

SRC	*DST*	*Path capacity*	*RTT*
NWU	CMU	95Mbps	18ms
NCTU	CMU	88Mbps	250ms
SLC	CMU	10Mbps	24ms

Table 1: **Internet paths**

Next we present Internet measurements for three paths with very different characteristics; the path information is shown in Table 1. We used the user-level implementation of *IGI* since we did not have root privileges on remote test machines. The accuracy of the user-level is very sensitive to host load, so the experiments were carried out when these machines are idle. Also, we always use CMU side as the destination since the output gaps are measured on the destination host. The probing packet size is fixed to 500B.

Figure 13 shows a 24-hour trace for the competing traffic throughput as measured using *IGI* from NWU (Northwest University) to CMU. It shows a clear correlation between the time of day and the amount of competing traffic. It agrees with the well-known day-time network traffic: day time traffic is higher than night traffic, while the lowest traffic appears in the early morning. We also occasionally measured the available bandwidth directly using a 5 seconds 100Mbps UDP flow from NWU to CMU (day time). We measured a throughput of 63Mbps, which agrees with our measurement. We did not use UDP flow to measure the available bandwidth continuously

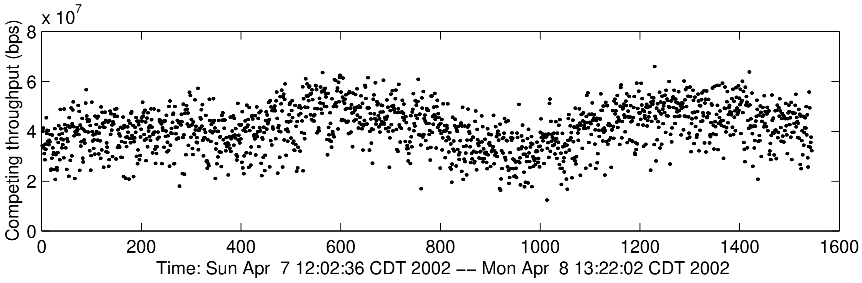

Figure 13: **NWU to CMU**. *Day-long trace of competing throughput measurements.*

because of the high load it places on the network.

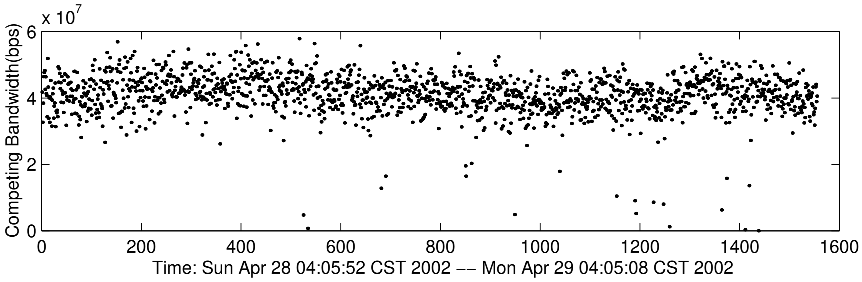

Figure 14: **NCTU to CMU.** *Day-long trace for competing throughput measurements.*

Figure 14 is the *IGI* estimates for the competing traffic on the bottleneck link between NCTU (National Chiao Tung University, Taiwan) and CMU. Measurements show that the bottleneck link capacity on this path is 88Mbps, and sampling using a UDP stream showed that the available bandwidth is 55Mbps, which validate our results. Unlike the data from NWU to CMU, the competing traffic does not show clear changes that match the day time versus night time pattern. We think it is due to the time difference (12 hours) between Taiwan and US, which makes it hard to say what time should be considered to be day time.

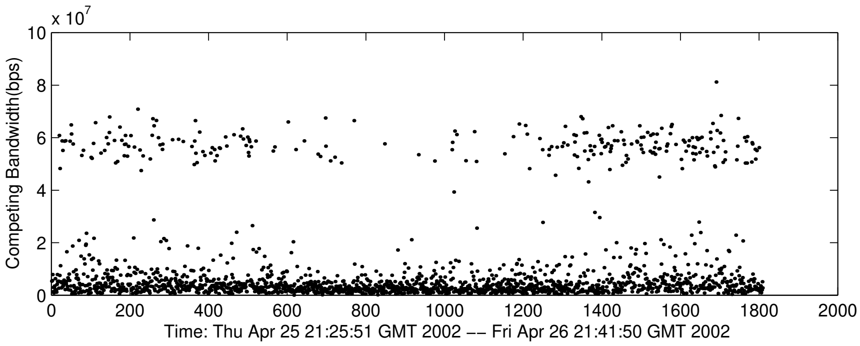

Figure 15: **SLC to CMU.** *Day-long trace for competing throughput measurement.*

Figure 15 shows similar results for experiments between the SLC host (Salt Lake City, UT) and CMU. While most of the measurements are accurate, we see some outliers around 60 Mbps, which is of course impossible with a 10Mbps bottleneck link capacity. Our analysis of these error cases shows that they are caused by bad estimates of the bottleneck link capacity.

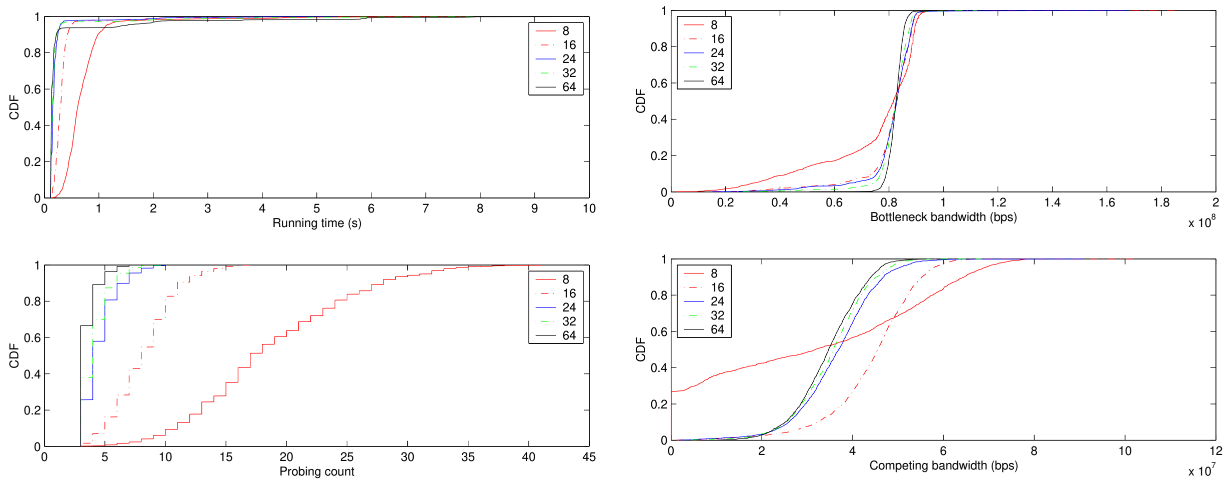

Figure 16: **Effect of probing packet number for NWU to CMU path.**

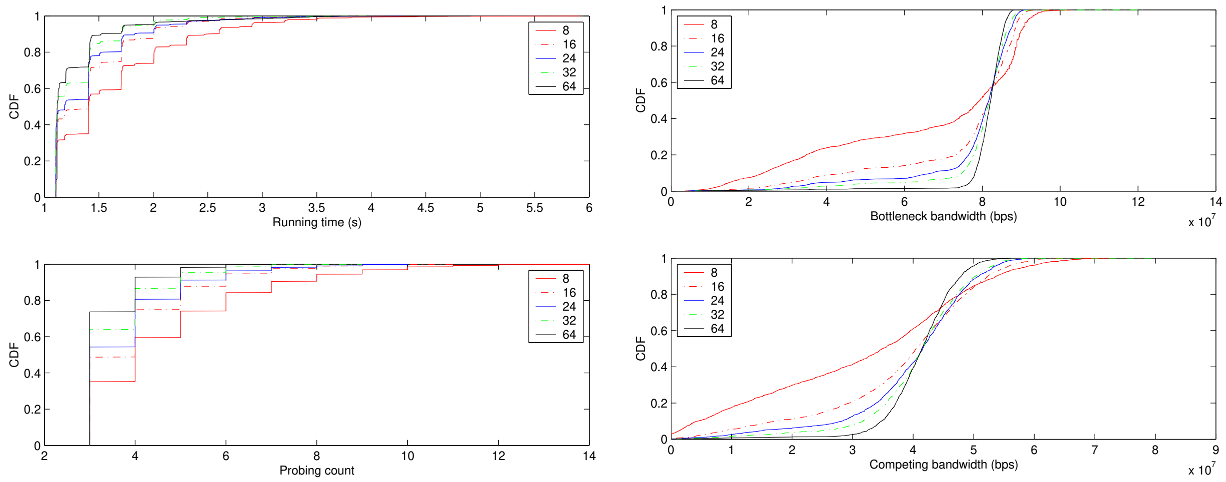

Figure 17: **Effect of probing packet number for NCTU to CMU path.**

5.3 Probing Time and Number of Probing Packets

In this section, we answer a question that is very relevant to application: How long does it take to get a reasonable bandwidth estimate? Also, how much "extra" network overhead does probing

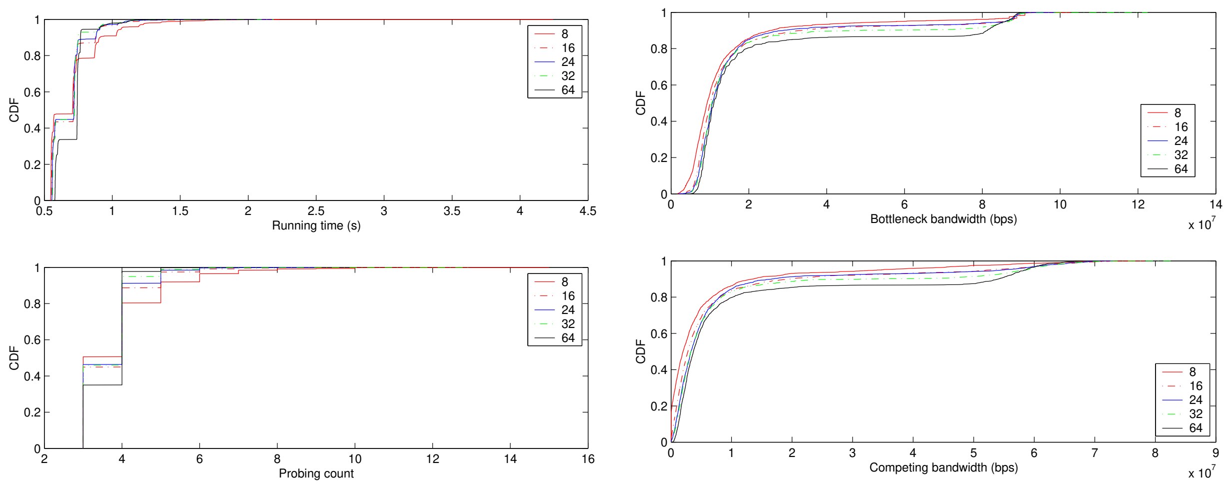

Figure 18: **Effect of probing packet number for SLC to CMU path.**

introduce? In other words, what is the total probing time and the number of packets sent. Note that the cost of the *IGI* algorithm depends on both the length of the packet trains used, and the number of phases needed to converge on the best input gap size (the turning point) for a path.

To minimize the number of probing phases, we optimized the source gap adjustment algorithm. The idea is to reduce the number of probing phases by carefully selecting the gap_step and src_gap values in *IGI* algorithm (Figure 8). We use the bottleneck link capacity to help by doing a first probing using a src_gap as small as possible. Then we set $gap\_step = g_B/8$, and $src\_gap = g_B/2$. We use this optimized algorithm in the following experiment. For the second question, i.e., the length of the packet train, we mentioned in Section 4.2 that 60 packets works well. We will experimentally evaluate whether we can reduce this number without affecting accuracy significantly.

To quantify the effect of these two parameters we ran a set of experiments with different parameter configurations. Since the available bandwidth on Internet paths is typically fairly stable, we ran experiments for 2 hour intervals. We use the same three Internet paths of Table 1 and the measurements are carried out using the user-level tool. We present the results for the three paths in Figures 16, 17, and 18. Each figure has graphs for the cumulative distribution of the probing time (left top), number of probing phases (left bottom), measured bottleneck bandwidth (right top), and measured competing bandwidth (right bottom). Each graph has five curves, corresponding to the results for packet trains of different lengths: 8, 16, 24, 32, and 64.

The graphs show that we only need 4-6 phases to converge on the final input gap value; this translates into a probing time of 4-6 round trip times. Probing with 8 probing packets does not work well for the NWU-CMU and NCTU-CMU paths, but when the probing packet number is higher than 16, we achieve fairly stable results for all three paths. We see that in 90% of the cases, probing can finish in 6 phases, and 80% of the competing traffic measurement cluster in a range smaller than 10% of the bottleneck link capacity. Note that the packet train length needed is different for the three paths. We need only short trains on the SLC-CMU paths but longer trains on the NCTU-CMU path. An analysis of the output gap distributions shows that the reason is the burstiness of the traffic: the competing traffic on the NCTU-CMU path is burstier than on the SLC-CMU path.

6 Multi-hop Networks

In Sections 4 and 5, we presented and evaluated the *IGI* method for estimating the available network bandwidth on a path. This method was based on the gap model presented in Section 3. The gap model is however based on a very simple single-hop network model and as such, it does not account for several factors that may affect the *IGI* method in practice. Let us briefly look at three factors:

1. *Non-bottleneck link effects*: the IGI method assumes that only the bottleneck link changes the gap. In practice, both upstream and downstream routers can affect the gap, thus introducing errors. For example, downstream routers may interleave the packet pairs with competing traffic, thus increasing the gap. Besides introducing a measurement error, this may also exacerbate the effect of the *DQR* region.

2. *Thick bottleneck link*: our method implicitly assumes that the bottleneck link is the tight link (the link with the smallest residual bandwidth on a network path). In practice, that may not be true.

3. *Accuracy under different traffic conditions*: the model does not show how the accuracy of the *IGI* method is affected by the traffic conditions. Understanding this relationship is important since it may allow us to, for example, tune the parameters better.

These are complex questions and more research is needed to resolve them in a satisfactory way. In this section we make a first step towards understanding these effects. We first introduce a very simple multi-hop gap model. We then use controlled experiment and simulation to quantify the effect of the factors listed above.

6.1 Multi-hop Gap Model

We simplify the single hop gap model into the model shown in Figure 19(a). That is, after a probing packet pair passes through a router, some of the gaps will be increased (with probability p_a, w_a is the average weight, i.e. the output gap is $w_a * g_I$), some stay the same (with probability p_b), and the remaining gaps are decreased (with probability p_c, the average output gap is $w_c * g_B$, where w_c is the weight).

Let us now consider the two-router network shown in Figure 19(b). The final output gap is the cascaded effect of routers R1 and R2. The combined effect of the two routers can result in 9 cases for the output gap values measured at R2, as shown in Table 2. Note that if the input gap g_I is increased by both R1 and R2 ($a1 \rightarrow a2$), the final gap is shown as g_{x+}, instead of $w_{a1}w_{a2}g_I$. This is because after the increase of R1, the input gap for R2 may have to reflect the effect of *DQR*, so we only know that $w_{a1}g_I \leq g_{x+} \leq w_{a2}w_{a1}g_I$. Similarly, when the probing gap is increased by R1 but decreased by R2, the final gap will be $g_{B2} \leq g_{x-} \leq w_{a1}g_I$.

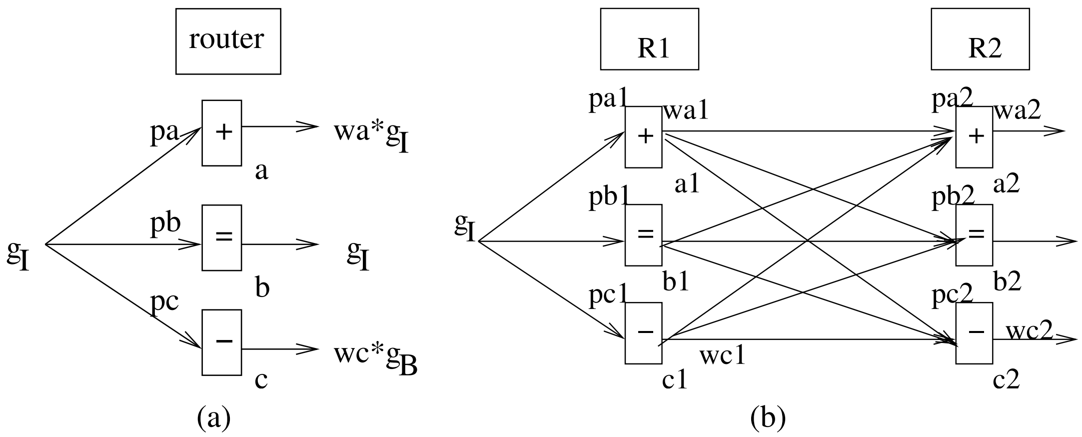

Figure 19: **(a) Simplified single hop gap model**. *The gap of a pair of probing packets can be increased, unchanged or decreased.* **(b) Simplified gap model for two routers**. *The final value of the probing packet gap is the cascade effect of two routers change.*

Gap Values	R1+ (p_{a1})	R1= (p_{b1})	R1$-$(p_{c1})
R2+(p_{a2})	g_{x+}	$w_{a2}g_I$	$w_{a2}w_{c1}g_{B1}$
R2=(p_{b2})	$w_{a1}g_I$	g_I	$w_{c1}g_{B1}$
R2$-$(p_{c2})	g_{x-}	$w_{c2}g_{B2}$	$w_{c2}g_{B2}$

Table 2: **Nine cases for the gap value after passing two routers**. *The "+", "=", and "$-$" indicate the effect (increase/unchange/decrease) of the router on the gap.*

Let us assume that <R1, R2> is the bottleneck link. Using the above notations, we can show (see Appendix A) that under heavy competing traffic on the outgoing link of router R2 (i.e., $p_{b1} \to 0, p_{b2} \to 0, g_{x+} \to w_{a1}w_{a2}g_I$), if $w_{a2}p_{a2} > 1$, R2 will increase R1's average output gap. The good news is that under light competing traffic on the outgoing link of R2 (i.e., $p_{b1} \to 0, p_{b2} \to 1, p_{a2} \to 0, p_{c2} \to 0$), R2 will not change R1's average output gap, which may explain why *IGI* often gives good results.

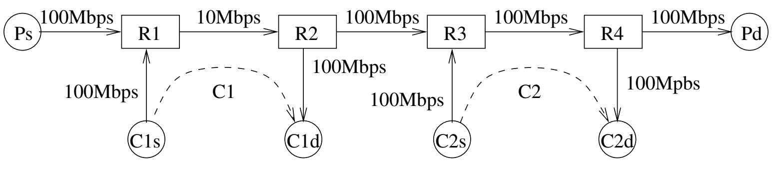

Figure 20: **Testbed configuration for multi-hop experiment**. *Ps and Pd are the probing source and destination; C1s and C1d are the source and destination nodes for the competing traffic C1 across the bottleneck link between R1 and R2, and C2s and C2d are the nodes that create the competing traffic C2 between R3 and R4.*

6.2 Testbed Experiment

In this section, we use a worst case scenario to explore how much non-bottleneck routers can affect the accuracy of the *IGI* method. The experimental setup is shown in Figure 20. The link <R1, R2> is the bottleneck link. It carries a competing flow C1 consisting of an *Iperf* stream with the source and destination TCP window sizes set as 512B and 2KB, resulting in a competing traffic flow of 308Kbps. For flow C2 on link R3-R4, the source and destination TCP window sizes are set to 16KB and 5KB, resulting in a much higher competing throughput on the non-bottleneck link of 20.7Mbps. Based on our analysis of the previous section, this is a very challenging case. Once the competing flows are stable, we sent out 1024 probing packets, with an input gap of $0.31ms$ and probing packet size of 100B.

At the destination host Pd, we get 1023 probing gap measurements. *tcpdump* results show that 116 gaps are increased gaps at R1, but as a result of flow C2 on R3-R4, 371 increased gaps are measured at the destination. Based on these measurements, the competing traffic throughput using the *IGI* formula is 1.433Mbps at R1, and 6.067Mbps at Pd. The reason for this big error is that the light competing traffic on bottleneck link <R1, R2> increases relatively few gap values, so the increases caused by R3 have a big impact. If the competing traffic on R1 is heavier, the impact of R3 is smaller. For example, after we change the competing traffic on R1-R2 and R3-R4 to 8.0Mbps and 50.0Mbps, respectively, the measured competing traffic throughput is 7.63Mbps at R1 and 8.89Mbps at Pd.

It is worthwhile to explore why the accuracy is low for low competing traffic. There are two reasons. First, light competing traffic only increases a small number of probing gaps, and the timing measurement error becomes more significant. The second reason is specific to our testbed setup. The stream of probing packets on the PC-based testbed router delays the processing the TCP ack packets, which slows down the source of the competing TCP flow. As a result, many probing packets are not interleaved as expected, thus fewer probing gaps are increased. In a more realistic setting with many parallel TCP flows, this problem does not occur. Over short time intervals, the cross traffic on a busy Internet link can be viewed as UDP traffic. For that reason, we primarily used UDP flows as competing traffic in our testbed experiments.

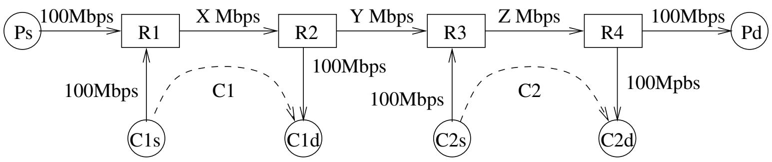

Figure 21: **Simulation configuration.** *Ps, Pd are the hosts used for probing, and C1s, C1d, C2s, C2d are used for the competing traffic generation. The numbers on the links denote the link capacity. The competing traffic flow C1 is the traffic we want to measure. The competing flow C2 is the competing traffic on a non-bottleneck link.*

6.3 Simulation of Non-bottleneck Effects

Due to the difficulty of setting up different scenarios on the testbed, we implemented the *IGI* algorithm in ns2 [2]. We now present the result of a sequence of experiments using the topology shown in Figure 21. By changing the values of the link capacities X, Y and Z, and of the competing loads C1 and C2, we can move the tight link and bottleneck link of the path. We set the router queue buffer size big enough so that no packet loss occurred in the simulation. We present the results for competing traffic consisting of a CBR UDP flow. We also ran simulations using TCP competing traffic and obtained very similar results [5].

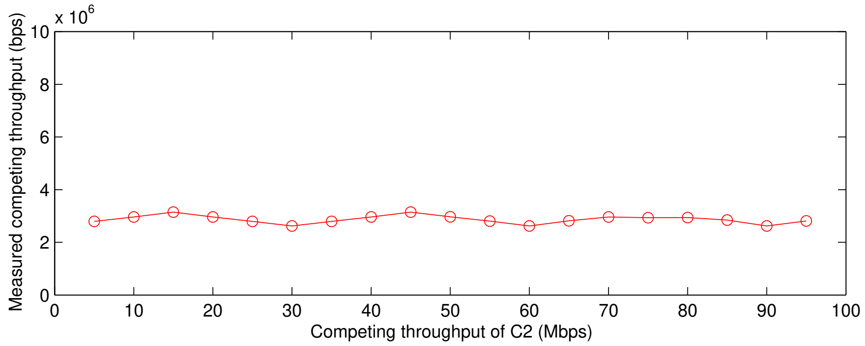

Figure 22: **Accurate measurement for competing traffic.** *X, Y and Z are set as 10Mbps, 100Mbps, and 100Mbps in Figure 21.*

In the first set of simulations we set (X, Y, Z) to (10 Mbps, 100 Mbps, 100Mbps). The competing load C1 is set to 3 Mbps, while we change the competing load C2 on <R3, R4> in the range 5-95 Mbps. Figure 22 shows that *IGI* accurately measures the competing load on the bottleneck link and that the results are not affected much by traffic on other links. This confirms some of our earlier experimental results.

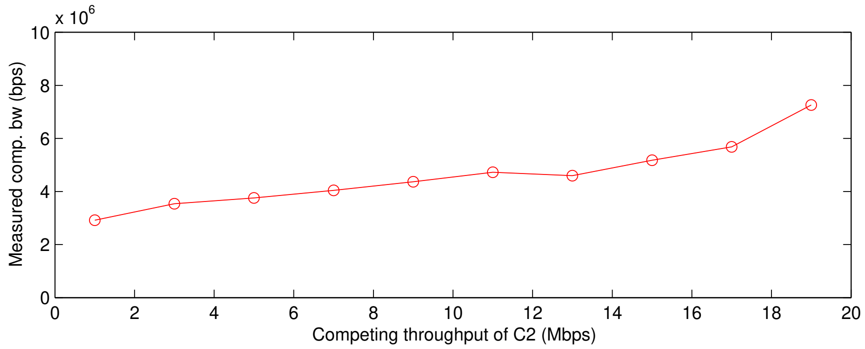

Figure 23: **Big error for competing traffic measurement.** *X, Y and Z are set as 10Mbps, 30Mbps, and 20Mbps in Figure 21.*

[5] The only difference is that we needed more probing packets (120) to capture the competing traffic.

Next we set (X, Y, Z) to (10Mbps, 30Mbps, 20Mbps). We set C1 to 3Mbps, and let C2 changes from 0 to 19Mbps. During this procedure, the tight link changes from <R1, R2> to <R3, R4>. The graph in Figure 23 shows how the *IGI* measurement of the competing traffic increases monotonically with C2. The result was obtained by assuming 10 Mbps <R1, R2> as the tight link, even though this is incorrect for the 13-19 Mbps range on <R3, R4>. Nevertheless, if we use these measurement to predict the available bandwidth, the result is fairly accurate, which is encouraging. We currently cannot explain this result using the gap model.

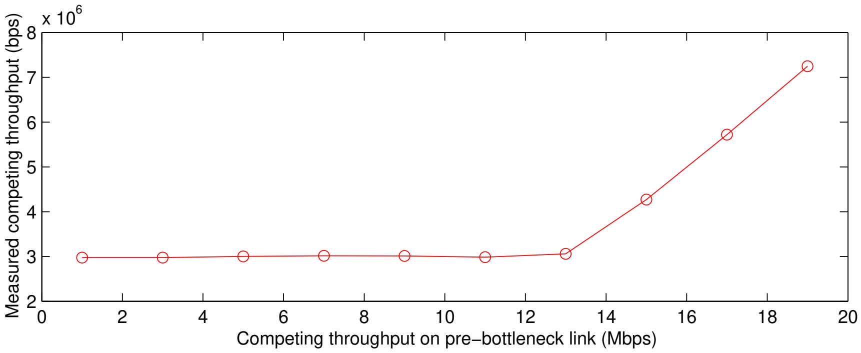

Figure 24: **Pre-bottleneck link does not significantly change the measurement.** *X, Y and Z are set as 20Mbps, 30Mbps, and 10Mbps in Figure 21.*

Finally, in Figure 24 we explore the effectiveness of *IGI* if we reorder the bottleneck and tight link. Using the network topology of Figure 21, we set (X, Y, Z) to (20Mbps, 30Mbps, 10Mbps), that is, we force <R3, R4> to be the bottleneck link. The bandwidth of the competing flow on <R1, R2> changes from 1 to 19Mbps, and the competing bandwidth on the bottleneck link (<R3, R4>) is again set to 3Mbps. The data in the graph shows that traffic on the pre-bottleneck link does not have any impact on the final measurements as long the traffic volume is less than 13Mbps. When the traffic volume on <R1, R2> increases above 13Mbps, that link becomes the tight link, and we see that IGI estimates an increase in the competing traffic bandwidth. The competing traffic measured by IGI is not an accurate estimate for the competing traffic on the tight link. However, if we subtract the estimated competing traffic from the bottleneck link capacity, we do get an accurate estimate for the available bandwidth on the end-to-end network path.

7 Conclusion

In this paper, we discuss the problem of estimating available bandwidth using measurements of the competing traffic based on packet pair probing. The idea is to measure increases in the gap value, with the assumption that the increases in the gap value reflect the amount of the competing traffic present on the tight link. We developed a single-hop gap model to help us understand the relationship between router queue size, competing traffic throughput, input gap and output gap. This model clearly shows the important role that the input gap plays in measuring competing traffic. In contrast to bottleneck link capacity measurements, in available bandwidth measurements, the input gap has to be dynamically tuned to reach the point where the packet pair gap values correctly reflect the available bandwidth. We also extend our analysis to deal with multi-hop networks, which provides insights into how non-bottleneck links may introduce measurements errors.

Based on what we learned from the gap model, we introduce a new measurement method called the Initial Gap Increase (*IGI*) algorithm. *IGI* uses a sequence of packet trains to identify the average input probing gap for which the average output gap is equal to the average input probing gap. At that point, called the turning point, the measurement errors should be minimal and we can accurately estimate the amount of competing traffic on the bottleneck router. Simulation, testbed experiments, and Internet experiments are used to study the properties and performance of our models and of the *IGI* algorithm. In 90% of our experiment, the *IGI* algorithm completes in 6 probing phases, and 80% competing traffic measurement cluster in a range smaller than 10% of the bottleneck link capacity.

Finally, we point out some areas for future work. A practical tools has to account for problems such as multiple-channel links and bottleneck link changes. Also, when the tight link is not the bottleneck link, there is some ambiguity on what the *IGI* algorithm measures. While we expect that in most cases, the bottleneck link will also be the tight link (e.g. the access link to the ISP), more research is needed to fine-tune *IGI* in these cases.

References

[1] libpcap. ftp://ftp.ee.lbl.gov/libpcap.tar.Z.

[2] Ns2. http://www.isi.edu/nsnam/ns.

[3] Paul Barford. *Modeling, Measurement and Performance of World Wide Web Transactions*. PhD thesis, December 2000.

[4] Jean-Chrysostome Bolot. End-to-end pakcet depay and loss behavior in the internet. In *Proc. ACM SIGCOMM'93*, San Francisco, CA, September 1993.

[5] Robert L. Carter and Mark E. Crovella. Dynamic server selection using bandwidth probing in wide-area networks. Technical report, Boston University Computer Science Department, March 1996.

[6] Robert L. Carter and Mark E. Crovella. Measuring bottleneck link speed in packet-switched networks. Technical report, Boston University Computer Science Department, March 1996.

[7] James Curtis and Tony McGregor. Review of bandwidth estimation techniques. In *Proceedings of the New Zealand Computer Science Research Students' Conference*, volume 8, University of Canterbury, New Zealand, April 2001.

[8] Constantinos Dovrolis, Parmesh Ramanathan, and David Moore. What do packet dispersion techniques measure? In *Proc. of ACM INFOCOM'01*, Anchorage, Alaska, USA, April 2001.

[9] Allen B. Downey. Clink: a tool for estimating internet link characteristics. http://rocky.wellesley.edu/downey/clink/.

[10] Van Jacobson. Congestion avoidance and control. In *Proc. ACM SIGCOMM 88*, August 1988.

[11] Van Jacobson. pathchar - a tool to infer characteristics of internet paths, 1997. presented as April 97 MSRI talk.

[12] Manish Jain and Constantinos Dovrolis. Pathload: A measurement tool for end-to-end available bandwidth. In *Passive and Active Measurements*, Fort Collins CO, March 2002.

[13] Srinivasan Keshav. Packet pair flow control. *IEEE/ACM Transactions on Networking*, February 1995.

[14] Kevin Lai and Mary Baker. Nettimer: A tool for measuring bottleneck link bandwidth. In *Proceedings of the USENIX Symposium on Internet Technologies and Systems*, March 2001.

[15] Bruce A. Mah. pchar: A tool for measuring internet path characteristics, 2001. http://www.employees.org/ bmah/Software/pchar/.

[16] Matthew Mathis and Jamshid Mahdavi. Diagnosing internet congestion with a transport layer performance tool. In *Proc. INET'96*, Montreal, Canada, June 1996.

[17] Steven McCanne and Van Jacobson. The BSD packet filter: A new architecture for user-level packet capture. In *USENIX Winter*, pages 259–270, 1993.

[18] Bob Melander, Mats Bjorkman, and Per Gunningberg. A new end-to-end probing and analysis method for estimating bandwidth bottlenecks. In *IEEE Globecom - Global Internet Symposium*, San Francisco, November 2000.

[19] Vern Paxson. *Measurements and Analysis of End-to-End Internet Dynamics*. PhD thesis, U.C. Berkeley, May 1996.

[20] Srinivasan Seshan, M. Stemm, and Randy H. Katz. Spand: shared passive network performance discovery. In *In Proc 1st Usenix Symposium on Internet Technologies and Systems (USITS '97)*, Monterey, CA, December 1997.

[21] Mark Stemm, Srinivasan Seshan, and Randy H. Katz. A network measurement architecture for adaptive applications. In *Proceedings of IEEE Infocom 2000*, Monterey, CA, March 2000.

[22] Ajay Tirumala and Jim Ferguson. Iperf. http://dast.nlanr.net/Projects/Iperf/.

APPENDIX

A The impact of multiple hops on the gap

Claim: *Under heavy competing traffic, if $w_2 p_{a2} > 1$, R2 will increase R1's output gap; and under light competing traffic, R2 will not change R1's output gap.* (see Section 6.1)

In Table 2, after the probing gaps pass R1, the average gap value will be:

$$p_{a1}w_1g_I + p_{b1}g_I + p_{c1}g_{B1}$$

Since R1 is the bottleneck router, we assume the competing traffic for it is heavy and most of the probing gaps will be increased or decreased, so $p_{b1} \to 0$. Then the above equation can be written as:

$$GAP_{R1} = p_{a1}w_1g_I + p_{c1}g_{B1}$$

After the probing gaps further pass R2, the average gap will be

$$p_{a1}(p_{a2}g_{x+} + p_{b2}w_1g_I + p_{c2}g_{x-}) + p_{b1}(p_{a2}w_2g_I + p_{b2}g_I + p_{c2}g_{B2}) + p_{c1}(p_{a2}w_2g_{B1} + p_{b2}g_{B1} + p_{c2}g_{B2})$$

Using $p_{b1} \to 0$, the above formula can be simplified as

$$GAP_{R2} = p_{a1}p_{a2}g_{x+} + p_{a1}p_{c2}g_{x-} + w_1p_{a1}p_{b2}g_I + p_{c1}(w_2p_{a2} + p_{b2})g_{B1} + p_{c2}p_{c1}g_{B2}$$

If the competing traffic on R2 is heavy, then we can assume $p_{b2} \to 0$ and $g_{x+} \to w_1w_2g_I$, so GAP_{R2} becomes:

$$p_{a1}p_{a2}w_1w_2g_I + p_{a1}p_{c2}g_{x-} + p_{c1}w_2p_{a2}g_{B1} + p_{c2}p_{c1}g_{B2}$$

And the difference between it and the average gap after R1 is:

$$\begin{aligned}
& p_{a1}p_{a2}w_1w_2g_I + p_{a1}p_{c2}g_{x-} + p_{c1}w_2p_{a2}g_{B1} + p_{c2}p_{c1}g_{B2} - p_{a1}w_1g_I - p_{c1}g_{B1} \\
> \;& p_{a1}p_{a2}w_1w_2g_I + p_{c1}w_2p_{a2}g_{B1} - p_{a1}w_1g_I - p_{c1}g_{B1} \\
= \;& (w_2p_{a2} - 1)(p_{a1}w_1g_I + p_{c1}g_{B1})
\end{aligned}$$

If $w_2p_{a2} > 1$, R2 will increase R1's average output gap.

On the other hand, if the competing traffic on R2 is very light, then we can assume $p_{b2} \to 1$ and $p_{a2} \to 0$, $p_{c2} \to 0$. GAP_{R2} becomes:

$$w_1p_{a1}g_I + p_{c1}g_{B1}$$

And the difference between R2's average output gap and R1's average output gap is

$$w_1p_{a1}g_I + p_{c1}g_{B1} - p_{a1}w_1g_I - p_{c1}g_{B1} = 0$$

That is, R2 will not change R1's average output gap.