2014

Network Path Capacity Comparison without Accurate Packet Time Information

Ertong Zhang
University of Nebraska - Lincoln, ezhang@cse.unl.edu

Lisong Xu
University of Nebraska - Lincoln, xu@cse.unl.edu

Follow this and additional works at: http://digitalcommons.unl.edu/cseconfwork
Network Path Capacity Comparison without Accurate Packet Time Information

Ertong Zhang and Lisong Xu
Department of Computer Science and Engineering
University of Nebraska-Lincoln
Lincoln, NE 68588-0115
Email: {ezhang, xu}@cse.unl.edu

Abstract—A fundamental problem of current bandwidth estimation methods is that they require accurate packet time information. However, it is hard to accurately measure packet time information in an increasing number of network environments, such as widely deployed highspeed networks, and emerging cloud computing networks. Motivated by the observation that many applications only need the relative bandwidth information of different paths instead of the actual bandwidth information of a single path, we propose sequence-based bandwidth comparison. Specifically, this paper proposes a capacity comparison method, called PathComp, which can relatively compare the capacities of the paths from two senders to the same receiver. PathComp mainly uses the arrival sequence information of packets, and does not require any accurate packet time information. Our testbed, campus network, and EC2 experiments show that PathComp can not only determine which path is faster but also accurately determine how much faster in a variety of network environments.

Index Terms—Internet measurement; Bandwidth estimation; Capacity estimation.

I. INTRODUCTION

A rich body of bandwidth estimation methods [20] have been proposed and studied in the past two decades, due to the wide range of applications of bandwidth estimation. However, there is a fundamental problem with the current bandwidth estimation methods. Most (if not all) of them need to accurately measure certain time information of network packets, such as the arrival time difference (ATD) between two consecutive packets [6], the one-way or round-trip delay of each packet [10], and the queueing delay of each packet [8]. However, it is hard and sometimes impossible to accurately measure these time information in an increasing number of network environments, such as widely deployed highspeed networks, and emerging cloud computing networks.

There are two major reasons why it is sometimes hard to accurately measure the packet time information. First, it takes very short times to send or receive packets at very high speeds. However, it is hard to measure such short times due to the limited system capability [9], [17]. Second, various software and hardware factors at the receiver of packets, such as interrupt moderation [19], [9] (commonly used in highspeed network cards) and virtual machine (VM) scheduling [21], [4] (commonly used in cloud computing), greatly change the original packet time information. As a result, the packet time information measured by the packet receiver is not correct.

Our work is motivated by the observation that many applications only need to relatively compare the bandwidth information of different paths. For example, in a peer-to-peer (P2P) network, a new peer needs to select several fastest peers as its neighbors from a set of existing peers. More motivating examples will be discussed in Section II-A. In these cases, we do not need to measure the actual bandwidth information of each path, instead, we only need to relatively compare the bandwidth information of different paths, and then rank them according to their bandwidth information.

In this paper, we study how to relatively compare the bandwidth information of multiple paths without requiring accurate packet time information. There are several important bandwidth metrics [20]. As the first step, this paper considers only the capacity of a path that is the capacity of the narrow link in the path, and the narrow link of a path is the link with the smallest capacity among all links in a path. The path capacity is a basic bandwidth metric and will provide useful information for studying other bandwidth metrics, such as available bandwidth and bulk TCP throughput, which will be considered in our future work.

Specifically, this paper proposes a capacity comparison method, called PathComp, which can relatively compare the path capacities from two senders to the same receiver. Basically, PathComp actively sends probing packets from both senders to the receiver, measures the arrival sequence of these packets at the receiver, and then relatively compares the capacities of the two paths.

PathComp is based on the fact that the inter-arrival gap between two consecutive packets from the same sender is related to the capacity of their path. This fact is also the basis of the current capacity estimation methods [6], [10]. The uniqueness of PathComp is that it measures the packet inter-arrival gap using the packet arrival sequence information, whereas the current capacity estimation methods measure the packet inter-arrival gap using the packet arrival time information. Therefore, PathComp does not require any accurate packet time information, and is fundamentally different from the current capacity estimation methods.

The contributions of this paper are as follows. First, we expand the design space of traditional time-based bandwidth
estimation methods by introducing a new class of sequence-
based bandwidth comparison methods. Note that bandwidth
comparison methods are inherently more scalable than tradi-
tional bandwidth estimation methods in terms of the mea-
asurement time for a large number of paths. This is because
bandwidth comparison methods are designed to simultane-
ously measure multiple paths, whereas traditional bandwidth
estimation methods are designed to measure a single path and
are sensitive to the interference among multiple concurrent
measurements [5].

Second, we propose a capacity comparison method, called
PathComp, which can determine not only which path is faster
but also how much faster in terms of the path capacity. In
the paper, we thoroughly study the impact of various types
of cross traffic on capacity comparison, and we also discuss
some implementation challenge, such as Receiver Side Scal-
ing [14]. Our testbed, campus network, and Amazon EC2 [1]
experiments show that PathComp can accurately compare the
capacities of two paths in a variety of network environments.

The remainder of this paper is organized as follows: Sec-
tion II gives more detailed description of our motivations.
Section III summarizes the related work and discusses the
design space of bandwidth estimation methods. Section IV
describes the basic idea of sequence-based capacity compa-
rison. Section V studies the impact of cross traffic. Section VI
presents our proposed PathComp. Section VII shows the
evaluation results. Finally, Section VIII concludes the paper.

II. MOTIVATION

A. Bandwidth Comparison Scenarios

Our work is motivated by the observation that many ap-
lications only need to relatively compare the bandwidth
information of different paths.

P2P neighbor selection: When a new peer joins a P2P
network [13], it usually needs to select its neighbors from a set
of existing peers. Typically, the new peer selects the existing
peers with fast network bandwidth as its neighbors so that it
can quickly download data from its neighbors. Bandwidth
comparison methods can be used to quickly select several
fastest peers from a set of existing peers.

Network-aware task placement [11]: Consider a bandwidth-
intensive cloud application with three tasks: \( T_1 \), \( T_2 \), and \( T_3 \),
and a cloud consisting of three interconnected VMs: \( V_1 \), \( V_2 \),
and \( V_3 \). Assume that tasks \( T_1 \) and \( T_2 \) communicate often with
task \( T_3 \), but not much with each other. If we find that the
path between \( V_1 \) and \( V_2 \) is the slowest one using a bandwidth
comparison method, and network measurements show that the
latency between any two of these three VMs is the same, then
the application performance can be improved with the optimal
task placement that places task \( T_3 \) on \( V_3 \), and places the other
two tasks on the other two VMs.

B. Difficulties in Obtaining Accurate Packet Time Information

Another motivation of our work is that the current time-
based capacity measurement algorithms do not work well in
some network environments, such as high-speed networks and
cloud computing networks, where it is hard to accurately
measure the packet time information. There are two major
reasons.

First, it takes very short times to send or receive packets at
very high speeds. For example, it takes only 12 \( \mu s \) to send
or receive a 1500-byte packet at 1 Gbps, and only 1.2 \( \mu s \) at
10 Gbps. However, it is hard to accurately measure such short
times due to the limited system capability [9], [17], such as
clock time resolutions, clock frequency differences between
the sender and the receiver, and the system call overhead.

Second, there are various software and hardware factors at
the receiver of packets, such as interrupt coalescence, context
switching, and virtual machine scheduling, which change the
original packet time information, and thus the packet time
information measured by the packet receiver is not correct.

Interrupt coalescence (IC, also called interrupt modera-
tion) [9], [19] is commonly used in high-speed network inter-
facing cards (NIC), and it reduces the CPU load by generating
an interrupt for a group of packets instead of each packet.
As a result, the packet time information, such as the ATD of
two consecutive packets, is changed (could be enlarged or be
reduced). Figure 1 shows the measured arrival times of 200
packets from a 10Gbps NIC with IC enabled, and we can see
that these packets are handled by 6 interrupts.

![Fig. 1. Impact of interrupt coalescence on packet arrival times and ATDs.](image)

VM scheduling [21], [4] is commonly used in cloud com-
puting, and it enables multiple VMs to share the same pool
of CPUs on a physical machine. However, it interferes with
packet timestamping of VMs. For example, when a VM is not
running, all packets arriving at the VM must wait until the
VM is scheduled to run again. As a result, the packet delays
and ATDs measured by the VM may be drastically different
from the actual values.

III. DESIGN SPACE AND RELATED WORK

We discuss the design space of capacity estimation methods
in Figure 2, which helps us to understand the relation between
the current capacity estimation methods and our proposed
capacity comparison method. Some methods measure the
capacity information of the path from a computer to another
computer, and we refer to the first computer as the sender
and the second computer as the receiver. Some other methods
measure the capacity information of the round-trip path from
a computer to another computer and then back to the first one.
For these methods, we refer to the first computer as both the
sender and receiver.
Both paths merge with each other at router R5. Network 5 in the figure represents everything between R5 (including R5) and RCV. Network 1 represents everything between SNDa and the narrow link of path a (called narrow link a), and network 2 for everything between narrow link a and R5. Similarly networks 3 and 4 for path b. Let $C_a$ denote the capacity of path $a$ that is the capacity of narrow link a, and let $C_b$ denote the capacity of path $b$ that is the capacity of narrow link $b$.

Note that Figure 3 assumes that paths $a$ and $b$ do not have a shared narrow link (i.e., if the narrow link is located in network 5). This is a reasonable assumption for a variety of scenarios. For example, consider the P2P neighbor selection problem described in Section II-A. The narrow link of the path from a neighbor to a peer is usually the upload link of the neighbor, and thus different neighbors usually do not have a shared narrow link. As another example, consider the network-aware task placement problem in Section II-A. The narrow link of the path from a sender VM to another receiver VM is usually located near the sender VM due to the rate limiting of the sender VM, and thus the paths from different sender VMs usually do not have a shared narrow link.

In cases where two paths have a shared narrow link, there are two options. First, capacity comparison reports the capacity ratio of the narrow links of the distinct segments of the two paths. Second, capacity comparison does not report anything, if a shared narrow link is detected. However, the method to
detect a shared narrow link is out of the scope of this paper. We choose the first option in this paper.

B. Traditional Time-based Capacity Estimation

The traditional capacity estimation methods, such as PathRate [6], and CapProbe [10], are usually based on packet arrival time differences (also called inter-arrival times, and dispersion times). The packet arrival time difference (ATD, denoted by \( \tau \)) of two packets is the time difference between their arrival times. For example, Figure 4 shows two SNDa packets, \( a_1 \) and \( a_2 \), on the link from network 5 to receiver RCV, and the time difference \( \tau \) is their ATD at RCV.

\[ \tau \text{ (seconds)} \]

network 5

RCV

\[ a_2 \]

\[ a_1 \]

Fig. 4. The ATD \( \tau \) at RCV between two SNDa packets (\( a_1 \) and \( a_2 \)) is their arrival time difference at RCV.

Assuming that there is no cross traffic on path \( a \), and assuming that RCV can accurately measure the ATD, the capacity \( C_a \) can be obtained as follows [6], [10], where \( S \) is the packet size and \( \tau \) is the ATD.

\[ C_a = \frac{S}{\tau} \quad (3) \]

The traditional capacity estimation methods mainly differ in how to accurately estimate \( C_a \) in the presence of cross traffic. However, if RCV cannot accurately measure the ATD, none of these methods works.

C. Proposed Sequence-based Capacity Comparison

1) ASND Definition: We propose to tackle the capacity comparison problem using packet arrival sequence number differences instead of packet arrival time differences, so that our method does not require accurate packet time information. Below we use an example to explain the concept of the packet arrival sequence number differences.

Each of the two senders, SNDa and SNDb, sends a train of \( L = 5 \) packets of the same packet size \( S \) to the receiver RCV at approximately the same time. These packets are sent back-to-back by their senders (i.e., at their maximum rates). In this example, we assume that there is no cross traffic in all 5 networks in Figure 3.

The top line of Figure 5 shows the 5 SNDa packets on the link from network 2 to router R5. Since there is no cross traffic, the ATD between every two consecutive SNDa packets at router R5 is inversely proportional to the capacity \( C_a \) of narrow link \( a \). The bottom line of Figure 5 shows the 5 SNDb packets on the link from network 4 to router R5, and the ATD between every two consecutive SNDb packets at the router is inversely proportional to the capacity \( C_b \) of narrow link \( b \). In this example, we set \( C_a = C_b/2 \), and thus the ATD between two consecutive SNDa packets is twice the ATD between two consecutive SNDb packets as illustrated in Figure 5.

These 10 packets merge with one another at router R5. In this example, we assume that these packets arrive at RCV in the order of their arrival times at router R5. For example, Figure 5 shows that packet \( b_1 \) arrives at R5 earlier than packet \( a_1 \), and thus packet \( b_1 \) arrives at RCV earlier than packet \( a_1 \).

In Section VI-C, we will discuss cases where this assumption does not hold and describe our solution. RCV assigns the first received packet an arrival sequence number of 1, and the next received packet an arrival sequence number of 2, and so on. Figure 6 shows the arrival order of these 10 packets at RCV, and their corresponding arrival sequence numbers.

2) ASND Histograms: We can infer the capacity ratio \( \gamma \) of paths \( a \) and \( b \) by analyzing their ASND histograms. In this subsection, we present the concept of ASND histograms, and in Section V, we will thoroughly study the impact of cross traffic on the ASND histograms.

Let \( H_a(i) \) (respectively, \( H_b(i) \)) denote the total number of pairs of two consecutive packets that are sent by SNDa (respectively, SNDb) and separated by \( \delta = i \) packets at RCV. For example, \( H_a(0) = 2 \) pairs in Figure 6, because the ASND between packets \( a_3 \) and \( a_4 \) is 0 and that between packets \( a_4 \) and \( a_5 \) is also 0. As another example, \( H_a(2) = 2 \) pairs, because the ASND between packets \( a_1 \) and \( a_2 \) is 2 and that between packets \( a_2 \) and \( a_3 \) is also 2.

The ASND histogram of the SNDa train is vector \( H_a = (H_a(0), H_a(1), H_a(2), \ldots) \), and that of the SNDb train is vector \( H_b = (H_b(0), H_b(1), H_b(2), \ldots) \). For example, Figure 7 shows ASND histograms \( H_a \) and \( H_b \) for SNDa and SNDb trains, respectively.

We have the following theorem to simplify our analysis of ASND histograms.

\[ \text{Theorem 1: } |H_a(0) - H_b(0)| \leq 1, \text{ if two trains have the same number } L \text{ of packets.} \]

\[ \text{Proof: Let symbols } a \text{ and } b \text{ (without the subscripts) to} \]
denote a packet of $\text{SND}a$ and $\text{SND}b$, respectively. The arrival order of the $2L$ packets can be described by a string consisting of $L$ symbol $a$’s and $L$ symbol $b$’s. For example, if $L = 5$, the arrival order of the 10 packets in Figure 6 can be described by string $aaabbbabb$, where the rightmost symbol (i.e., $b$) is the first packet received by $RCV$ (i.e., $b_1$), and the leftmost symbol (i.e., $a$) is the last packet received by $RCV$ (i.e., $a_5$).

A string of $L$ symbol $a$’s and $L$ symbol $b$’s can be classified into the following four cases, according to the leftmost and rightmost symbols. We will prove only cases 1 and 2, and the other two cases can be proved very similarly.

- **Case 1:** The leftmost one: $a$, and the rightmost one: $a$.
- **Case 2:** The leftmost one: $a$, and the rightmost one: $b$.
- **Case 3:** The leftmost one: $b$, and the rightmost one: $a$.
- **Case 4:** The leftmost one: $b$, and the rightmost one: $b$.

**Case 1:** For a string with $2L$ symbols, there are the total of $2L - 1$ pairs of two consecutive symbols. Let $n(aa)$, $n(ab)$, $n(ba)$, and $n(bb)$ denote the number of pairs $aa$, $ab$, $ba$, and $bb$, respectively. Let $n(*a)$ and $n(*b)$ denote the number of pairs whose right symbol is $a$ and $b$, respectively. By definition, we have $n(*a) = n(ba) + n(aa)$, and $n(*b) = n(ab) + n(bb)$.

We have $n(*a) = L - 1$, because the leftmost $a$ cannot be the right symbol of a pair. We also have $n(*b) = L$. Therefore, we have $n(*a) = n(*b) - 1$.

Since both the leftmost and the rightmost symbols are $a$, we have $n(ab) = n(ba)$. Therefore, $H_a(0) = \lceil n(aa) - n(ab) \rceil = (n(*b) - 1) - n(ab) = n(bb) - 1 = H_b(0) - 1$.

**Case 2:** We have $n(*a) = L - 1$, because the leftmost $a$ cannot be the right symbol of a pair. We also have $n(*b) = L$. Therefore, we have $n(*a) = n(*b) - 1$.

Since the leftmost symbol is $a$ and the rightmost one is $b$, we have $n(ab) = n(ba) + 1$. Therefore, $H_a(0) = n(aa) = n(*a) - n(ba) = (n(*b) - 1) - (n(ab) - 1) = n(bb) = H_b(0)$.

For example, $H_a(0) - H_b(0) = 2 - 2 = 0$ in Figure 7. Note that, Theorem 1 holds no matter whether there is cross traffic or not and no matter how long the train size $L$ is.

We will not show and will not use $H_a(0)$ and $H_b(0)$ in the rest of the paper for the following two reasons. First, usually the $\text{SND}a$ and $\text{SND}b$ trains only partially overlap with each other, and thus $H_a(0)$ and $H_b(0)$ are mainly due to the non-overlapping packets of the two trains. Second, Theorem 1 shows that $H_a(0)$ and $H_b(0)$ are very close to each other, and thus do not provide much useful information.

A peak (also called a mode) in a histogram is a local maximum that is higher than its right neighbors and no less than its left neighbor (if exists). For example, histogram $H_a$ in Figure 7 has a peak at 2 packets, and histogram $H_b$ has a peak at 1 packet (note that it does not have the left neighbor, since we do not consider $H_b(0)$).

We introduce the second theorem about the peaks in ASND histograms. Without loss of generality, this theorem considers only case $C_a \leq C_b$.

**Theorem 2:** In the absence of cross traffic, if $C_a \leq C_b$, histogram $H_a$ has only one peak and the peak is located at $\Gamma$ packets, and histogram $H_b$ has only one peak and the peak is located at 1 packet. The capacity ratio $\gamma$ can be obtained as follow, where $H_a(\Gamma - 1)$ should be set to 0 if $\Gamma = 1$.

$$
\gamma = \frac{(\Gamma - 1)H_a(\Gamma - 1) + \Gamma H_a(\Gamma) + (\Gamma + 1)H_a(\Gamma + 1)}{H_a(\Gamma - 1) + H_a(\Gamma) + H_a(\Gamma + 1)}
$$

**Proof:** When there is no cross traffic, the ATD of a pair of two consecutive $\text{SND}a$ packets is $S/C_a$. The average number of $\text{SND}b$ packets that can be transmitted during an $S/C_a$ interval is $(S/C_a) \times (C_b/S) = C_b/C_a$. Therefore, the average number of $\text{SND}b$ packets between a pair of two consecutive $\text{SND}a$ packets is $C_b/C_a$. We consider the following three possible cases:

- **Case 1:** $C_b/C_a$ is a positive integer. That is, $\Gamma = C_b/C_a$. In this case, there are exactly $\Gamma$ $\text{SND}b$ packets between a pair of two consecutive $\text{SND}a$ packets. Therefore, the peak of $H_b$ is at $\Gamma$ packets. In this case, there are either 0 or 1 $\text{SND}a$ packet between a pair of two consecutive $\text{SND}b$ packets. Therefore, the peak of $H_a$ is at 1 packet. Note that, $H_a(\Gamma - 1) = H_a(\Gamma + 1) = 0$, and thus Equation (4) can be proved.

- **Case 2:** $C_b/C_a$ is a decimal greater than 1, and $\Gamma = \lceil C_b/C_a \rceil$ and $\Gamma + 1 = \lceil C_b/C_a \rceil$. In this case, there are either $\lceil C_b/C_a \rceil$ or $\lceil C_b/C_a \rceil - 1$ $\text{SND}b$ packets between a pair of two consecutive $\text{SND}a$ packets. Because $\Gamma = \lceil \gamma \rceil = \lceil C_b/C_a \rceil$, we have $H_a(\Gamma) > H_a(\Gamma + 1) > 0$. Therefore, the peak of $H_a$ is at $\Gamma$ packets. In this case, there are either 0 or 1 $\text{SND}a$ packet between a pair of two consecutive $\text{SND}b$ packets. Therefore, the peak of $H_b$ is at 1 packet. Note that, $H_a(\Gamma - 1) = 0$, and thus Equation (4) can be proved.

- **Case 3:** $C_b/C_a$ is a decimal greater than 1, and $\Gamma - 1 = \lceil C_b/C_a \rceil$ and $\Gamma = \lceil C_b/C_a \rceil$. This case can be proved in a similar way to case 2.

For example, in Figure 7, because $C_a < C_b$, histogram $H_a$ has a peak at $\Gamma = \lceil C_b/C_a \rceil = 2$ packets, and histogram $H_b$ has a peak at 1 packet.

3) **ASND-based Capacity Comparison:** Theorem 2 provides the foundation of our proposed capacity comparison method in the absence of cross traffic. Given the histogram $H$ of the slower path, algorithm EST-RATIO can estimate the capacity ratio $\gamma$ using Theorem 2. Since initially we do not know which path is slower, algorithm COMPARE calculates two ratio estimates: $\gamma_a$ assuming path $a$ is slower, and $\gamma_b$ assuming path $b$ is slower. Then it selects the bigger ratio as the final result.
Algorithm 1: Estimate the capacity ratio from histogram $H$ using Theorem 2 in the absence of cross traffic.

1: function EST-RATIO($H$)
2: $\Gamma \leftarrow \max(H(1), H(2), H(3), \ldots)$ \triangleright Find the peak
3: $\gamma \leftarrow \frac{H((\Gamma+1))}{H((\Gamma+1)) + H((\Gamma+1))}$ \hfill $H(\Gamma)$
4: return $\gamma$
5: end function

Algorithm 2: Compare the capacities of two paths using their histograms $H_a$ and $H_b$ in the absence of cross traffic.

1: function COMPARE($H_a$, $H_b$)
2: \begin{align*}
\gamma_a & \leftarrow \text{EST-RATIO}($H_a$) \quad \triangleright \text{Assuming } a \text{ is slower} \\
\gamma_b & \leftarrow \text{EST-RATIO}($H_b$) \quad \triangleright \text{Assuming } b \text{ is slower}
\end{align*}
3: if $\gamma_a = \gamma_b$ then
4: \hspace{1em} print Path $a$ is as fast as path $b$.
5: else if $\gamma_a > \gamma_b$ then
6: \hspace{1em} print Path $a$ is slower than path $b$.
7: \hspace{1em} print $C_b/C_a = \gamma_a$
8: else
9: \hspace{1em} print Path $a$ is faster than path $b$.
10: \hspace{1em} print $C_a/C_b = \gamma_b$
11: end if
12: end if
13: end function

V. IMPACT OF CROSS TRAFFIC

In this section, we study the impact of various types of cross traffic on the ASND histograms using our lab testbed.

We study five possible types of cross traffic as illustrated in Figure 8, which is very similar to Figure 3 and just simplifies each network to a single router. The capacity of each link is chosen to demonstrate the impact of the cross traffic on that link. We emulate this network using our 10Gbps testbed, and each link is emulated by a Linux token bucket filter (tbf).

![Fig. 8. Five possible sources of cross traffic. Link capacity unit: Mbps.](image)

The narrow link of path $a$ from SNDa to RCV is the link between routers $R1$ and $R2$, and thus the capacity of path $a$ is $C_a = 200Mbps$. The narrow link of path $b$ from SNDb to RCV is the link between routers $R3$ and $R4$, and thus the capacity of path $b$ is $C_b = 800Mbps$. Therefore, we have $\Gamma = \gamma = 4$.

In each of the following experiments, each sender sends a train of $L = 500$ packets at approximately the same time, and RCV measures the ASND histograms. Because path $b$ is the faster one, all SNDb histograms concentrate at 0 and 1 packet (similar to Figure 7(b)), and thus we do not show the SNDb histograms. For the SNDa histograms, we do not show the result for 0 packet, as explained in Section IV.

1) No Cross Traffic: As a reference case, first we do not generate any cross traffic. Since $\gamma = 4$, there should be 4 SNDb packets between a pair of two consecutive SNDa packets, as illustrated in Figure 9. The SNDa histogram is shown in Figure 14. As we expect, the ASND of most SNDa pairs is 4 packets. But there are a small number of SNDa pairs with other ASNDs, which are mainly caused by the randomness of the routers that are emulated using our lab computers and Linux tbf.

2) Cross Traffic between R1 and R2: This experiment shows the impact of cross traffic before or on the narrow link of path $a$ (i.e., the slower path). Random cross traffic is generated using MGEM [15] at an average rate of $200*50\% = 100$ Mbps between $R1$ and $R2$.

Let’s consider the example shown in Figure 10. There are still the same 8 SNDb packets passing the link between $R3$ and $R4$ as in Figure 9. But during this time interval, a cross traffic packet is inserted between the first (i.e., the rightmost one) and second SNDa packets (the third SNDa packet is further delayed, and not shown in the figure). As a result, the ASND between the first and second SNDa packets is doubled and becomes 8 packets.

This is why the SNDa histogram in Figure 15 has a non-negligible number of SNDa pairs with $\delta = 8$ packets. Further more, the numbers of SNDa pairs with $\delta = \gamma_i = 4i$ packets approximately follow a Geometric distribution described by Equation (5), where $N$ is the total number of pairs with $\delta = 4i$ packets, and $p$ is the occurrence probability of a cross traffic.
packet. For example, the dotted line in Figure 15 is obtained using Equation (5) with the corresponding $N$ and $p$.

$$H_a(i * 4) = N p^{i-1} (1 - p) \quad 1 \leq i \quad (5)$$

3) Cross Traffic between R2 and R5: This experiment shows the impact of cross traffic beyond the narrow link of path $a$ but still before the shared segment. Random cross traffic is generated at an average rate of 400 * 80% = 320 Mbps between R2 and R5.

In the example shown in Figure 11, there are still the same 8 SNDa packets passing the link between R4 and R5 as in Figure 9. But a cross traffic packet is inserted between the first and second SNDa packets. Because the link capacity between R2 and R5 is twice that between R1 and R2, the third SNDa packet can still be transmitted at the original time as in Figure 9. As a result, the ASND between the first and second SNDa packets increases to 5 packets, but the ASND between the next two SNDa packets decreases to 3 packets.

This is why the SNDa histogram shown in Figure 16 has a non-negligible number of SNDa pairs with ASNDs around 4 packets, such as 3 and 5 packets.

4) Cross Traffic between R3 and R4: This experiment shows the impact of cross traffic before or on the narrow link of path $b$ (i.e., the faster path). Random cross traffic is generated at an average rate of 800 * 50% = 400 Mbps between R3 and R4.

In the example shown in Figure 12, there are still the same 3 SNDa packets passing the link between R1 and R2 as in Figure 9. But three cross traffic packets are inserted between these SNDb packets (the rightmost three SNDb packets in Figure 9 are further delayed, and not shown in Figure 12). As a result, the ASND between the first and second SNDa packets decreases to 2 packets, and the ASND between the next two SNDa packets decreases to 3 packets.

This is why the SNDa histogram in Figure 17 has a large number of SNDa pairs with ASND less than 4 packets. The numbers of SNDa pairs with ASNDs between 1 and 4 packets follow a Binomial distribution described by Equation (6), where $N$ is the total number of SNDa pairs with ASNDs between 1 and 4 packets, and $p$ is the occurrence probability of a cross traffic packet. The dotted line in Figure 17 is obtained using Equation (6) with the corresponding $N$ and $p$.

$$H_a(i) = N \binom{4}{i} (1-p)^i p^{4-i}/(1-p^4) \quad 1 \leq i \leq 4 \quad (6)$$

5) Cross Traffic between R4 and R5: This experiment studies the impact of cross traffic beyond the narrow link of path $b$ but still before the shared segment. Random cross traffic is generated at an average rate of 1200 * 50% = 600 Mbps between R4 and R5.

In the example shown in Figure 13, there are still the same 3 SNDa packets passing the link between R2 and R5 as in Figure 9. But during this time interval, several cross traffic packets are inserted between the first and second SNDb packets. Because the link capacity between R4 and R5 is higher than that between R3 and R4, the remaining SNDa packets are only slightly delayed than in Figure 9. As a result, the ASND between the first and second SNDa packets decreases to 1 packet, and the ASND between the next two SNDa packets becomes 6 packets, which is the capacity ratio of the link between R4 and R5 to the link between R1 and R2 (i.e., 1200/200=6).

This is why the SNDa histogram shown in Figure 18 has a large number of SNDa pairs with ASND not equal to 4 packets, such as 1 and 6 packets.
6) Cross Traffic between R5 and RCV: This experiment shows the impact of cross traffic in the shared segment of both paths. As we expect, this type of cross traffic does not have any impact on the histograms.

7) Cross Traffic on All Five Links: Finally, we generate all five types of cross traffic. The SNDa histogram is shown in Figure 19, and it shows the combination of the impact of all five types of cross traffic.

8) Summary: We have the following observations about the ASND histogram of the slower path.

Observation 1: In practice, the ASND histogram could have a small number of ASND values caused by the randomness of the end-systems and the networks. These ASND values should be treated as noises and be discarded.

Observation 2: With little or no cross traffic, there is only one peak and the peak is located at the rounded capacity ratio \( \Gamma \) (e.g., Figure 14). This is consistent with Theorem 2.

Observation 3: In the presence of cross traffic, it is possible that there are multiple peaks (also called multi-mode), and \( \Gamma \) may or may not be the location of a peak. For example, Figure 18 has peaks at 1 and 6 packets, but no peak at 4 packets. We observe that multiple peaks are usually caused by the cross traffic on the faster path. More specifically, they are usually caused by the cross traffic beyond the narrow link of the faster path, e.g., in Figures 18 and 19.

Observation 4: In the presence of cross traffic, if there is only one peak, the peak location tends to be a lower bound of \( \Gamma \) (e.g., Figure 16 has a single peak at 4 packets, and Figure 17 has a single peak at 3 packets). Intuitively, this is because an ASND value greater than \( \Gamma \) usually leads to another ASND value smaller than \( \Gamma \) (e.g., Figures 11 and 13). Therefore, if there is a peak to the right of \( \Gamma \), then there is usually another peak to the left of \( \Gamma \). That is, there will be multiple peaks.

Observation 5: We have calculated and verified that the average of all ASND values of a histogram could be higher than or lower than \( \gamma \) (i.e., neither an upper bound nor a lower bound), depending on the amounts and locations of cross traffic on both paths.

VI. PATHCOMP

In this section, we present our proposed PathComp to relative compare the capacity ratio of two paths to the same receiver without requiring accurate packet time information.

A. The PathComp Method

PathComp follows the basic idea of algorithms EST-RATIO and COMPARE as described in Section IV-C3. However, there are two problems with algorithm EST-RATIO in the presence of cross traffic. 1) It is possible to have multiple peaks in a histogram mainly due to the cross traffic on the faster path (i.e., Observation 3 in Section V-8), however EST-RATIO assumes only one peak in a histogram. To tackle this problem, we divide the long packet train on the faster path into multiple short packet blocks, in order to reduce the impact of cross traffic. 2) If there is a single peak in the histogram of the slower path, the peak location tends to be a lower bound of \( \Gamma \) (i.e., Observation 4 in Section V-8). To tackle this problem, we estimate \( \Gamma \) by the peak of the weighted histogram.

PathComp consists of three phases as shown in Figure 20.

- 1) Preliminary phase measures some basic network information, such as the round-trip times (RTTs).
- 2) Phase I measures the histograms of the two paths. If there is a single peak in the histogram of the slower path, PathComp estimates \( \gamma \) using algorithms COMPARE and EST-RATIO2; otherwise, it starts Phase II.
- 3) Phase II re-measures the histograms using multiple packet blocks on the faster path, and then estimates \( \gamma \) using algorithm EST-RATIO2.

Figure 20 still considers the two paths shown in Figure 3. But to simplify the figure, Figure 20 assumes that there is only one link between SNDa and R5 that is the narrow link of path \( a \), and there is only one link between SNDb and R5 that is the narrow link of path \( b \).

1) Preliminary Phase: This phase measures the RTT difference \( \Delta RTT \) between SNDa-RCV and SNDb-RCV as illustrated in Figure 20, so that in the next two phases the packets of SNDa and SNDb can overlap with each other. PathComp measures \( \Delta RTT \) multiple times, and calculates the mean (denoted by \( \bar{\Delta RTT} \)) and standard deviation (denoted by \( \sigma(\Delta RTT) \)) of measured \( \Delta RTT \) values.

2) Phase I: RCV first tells the sender with a longer RTT (i.e., SNDb in Figure 20) to start its packet transmission, and after a delay of \( \bar{\Delta RTT} \), RCV then tells the sender with a shorter RTT (i.e., SNDa) to start its packet transmission. Each sender sends a train of \( L \) consecutive packets with the same packet size \( S \). In Figure 20, the capacity \( C_a \) of path \( a \) is lower
than the capacity $C_b$ of path $b$, so $SNDa$ takes a longer time to transmit the same number $L$ of packets than $SNDb$.

After $RCV$ receives these two trains, PathComp measures ASND histograms $H_a$ and $H_b$ of the two paths, and uses Algorithms EST-RATIO2 and COMPARE2 to estimate the capacity ratio. The difference between EST-RATIO and EST-RATIO2 is that the former selects the peak from the original histogram $H = (H(1), H(2), H(3), ...)$, whereas the latter selects the peak from the weighted histogram $(H(1), 2H(2), 3H(3), ...)$. This is motivated by Observation 4 in Section V-8. Note that the peak location of the weighted histogram is greater than or the same as that of the original histogram. The difference between COMPARE and COMPARE2 is that the former calls EST-RATIO whereas the latter called EST-RATIO2. In addition, if multiple peaks are detected in the histogram of the slower path, Algorithm COMPARE2 starts phase II.

**Algorithm 3** Estimate the capacity ratio from histogram $H$ in the presence of cross traffic

1: function EST-RATIO2($H$)  
2: Remove measurement noises from $H$  
3: $\Gamma \leftarrow \max(H(1), 2H(2), 3H(3), ...)$ \hspace{1cm} ▶ Weighted  
4: $\gamma \leftarrow \frac{(\Gamma - 1)H(\Gamma - 1) + H(\Gamma) + (\Gamma + 1)H(\Gamma + 1)}{H(\Gamma - 1) + H(\Gamma) + H(\Gamma + 1)}$  
5: return $\gamma$  
6: end function

**Algorithm 4** Compare the capacities of two paths using their histograms $H_a$ and $H_b$ in the presence of cross traffic

1: function COMPARE2($H_a$, $H_b$)  
2: $\gamma_a \leftarrow$ EST-RATIO2($H_a$) \hspace{1cm} ▶ Assuming $a$ is slower  
3: $\gamma_b \leftarrow$ EST-RATIO2($H_b$) \hspace{1cm} ▶ Assuming $b$ is slower  
4: if $\gamma_a = \gamma_b$ then  
5: print Path $a$ is as fast as path $b$.  
6: else if $\gamma_a > \gamma_b$ then  
7: print Path $a$ is slower than path $b$.  
8: if $H_a$ has multiple peaks then  
9: starts Phase II  
10: else  
11: print $C_b/C_a = \gamma_a$  
12: end if  
13: else  
14: print Path $a$ is faster than path $b$.  
15: if $H_b$ has multiple peaks then  
16: starts Phase II  
17: else  
18: print $C_a/C_b = \gamma_b$  
19: end if  
20: end if  
21: end function

**Parameter Setting:** If $\sigma(\Delta RTT) = 0$, the two trains should arrive at $RCV$ at the same time as illustrated in Figure 20. In practice, $\sigma(\Delta RTT) > 0$, and the train size $L$ should be sufficiently long so that the two trains can still overlap with each other. For example, consider a cloud computing network in a data center with $\sigma(\Delta RTT)=1$ ms and with the capacity=1 Gbps, $L$ should be at least 83 packets longer to compensate for the RTT variance if packet size $S$ is 1500 Byte. By default, PathComp sets the train size $L$ to 500 packets.

If the two trains could not overlap with each other, or overlap for only a small portion of each train, PathComp increases the train size and re-sends the two trains. However, if excessive packet loss is detected at $RCV$, PathComp quits the estimation.

By default, PathComp sets the packet size $S$ to 1500 bytes. This is because our experiments show that ASND histograms become hard to predict and analyze when the packet size is small. Intuitively, this is because the randomness of the end-systems and networks have a big impact on small packets, and thus there are much more noises in the ASND histograms.

3) Phase II: PathComp enters this phase, if there are multiple peaks in the histogram of the slower path. As observed in Section V-8, this is usually due to the high cross traffic load on the faster path. Therefore, we divide the long packet train on the faster path into multiple short packet blocks, in order to reduce the impact of cross traffic.

Specifically, PathComp still sends a train of $L$ packets back-to-back on the slower path. But on the faster path, PathComp sends a block of $B$ packets back-to-back every $\Delta t$ time interval, until all $L$ packets have been sent out, as illustrated in Figure 20. After $RCV$ receives these two trains, PathComp measures only the ASND histogram of the slower path, and uses Algorithm EST-RATIO2 to estimate the capacity ratio.

**Parameter Setting:** The block size $B$ should be much larger than the capacity ratio $\gamma$, because $B$ limits the maximum ASND between two consecutive packets on the slower path. By default, PathComp sets $B$ to 20 packets, which is larger than most typical ratios, such as 2 and 10.

The interval $\Delta t$ should be long enough in order to sufficiently separate different packet blocks, but should not be too long so that most packets on the faster path can still overlap with the packets on the slower path. By default, PathComp sets $\Delta t$ to $2T_f/(L/B) = 2BT_f/L$, so that the average transmission rate of all packets is approximately reduced by half and the total transmission time of all packets is approximately doubled. $T_f$ is the time for $RCV$ to receive the packet train from the faster path in Phase I.

PathComp checks whether $\Delta t$ is too long or too short as follows. If less than half of the packet blocks on the faster path overlap with the packet train on the slower path, it is likely that $\Delta t$ is too big. If the ASND histogram of the faster path contains very few large ASND values (e.g. $\delta \geq 5$), it is likely that $\Delta t$ is too short. In these cases, PathComp adjusts the interval $\Delta t$ and re-sends the packets.

As an example, Figure 19 shows the original $SNDa$ histogram with multiple peaks obtained using the packet train on the faster path, and Figure 21 shows the new $SNDa$ histogram obtained using packet blocks on the faster path. We can see that in the new histogram, there are still multiple peaks, but there is a peak at $\Gamma = 4$, and it is the highest peak.
is a relatively long time (e.g., 0.6 ms at 4 Gbps, due to RTT is the time for receiving a train of packets, not an individual packet. Therefore, due to the relatively large value of L (e.g., 500), T_f is a relatively long time (e.g., 0.6 ms at 10 Gbps). In addition, too large or too small interval Δt due to the inaccuracy in measuring T_f will be detected and adjusted by PathComp in Phase II.

B. An Implementation Challenge: RSS and IC

Two features of high-speed NICs may interfere with PathComp: Receiver Side Scaling (RSS) and Interrupt Coalescence (IC). Each of them alone does not affect PathComp, but when both of them are enabled, they greatly interfere with PathComp. Below we explain the reasons and our solution.

RSS [14] is a relatively new NIC feature to allow a NIC to balance interrupts among multiple CPUs in a computer. RSS distributes incoming packets into different NIC Rx queues according to their hash values calculated using the packet information, such as source IP. As a result, the probing packets from two different senders are placed into different NIC Rx queues and handled by different CPUs on RCV. IC [9] is a NIC feature to reduce the CPU load by generating an interrupt for a group of packets instead of each packet. When an interrupt is generated as each packet arrives (i.e., IC disabled), RSS alone does not affect PathComp because the interrupt sequence follows the packet arrival sequence. When there is only a single NIC Rx queue (i.e., RSS disabled), IC along does not affect PathComp because IC changes only the packet arrival times but not the packet arrival sequence. However, when both RSS and IC are enabled, they greatly interfere with PathComp as illustrated in Figure 22. Packets from different senders are placed into different NIC Rx queues, and an interrupt is generated only for a group of packets from a Rx queue. As a result, the packet arrival sequence measured by PathComp is different from the original packet arrival sequence at the NIC.

Fig. 21. The difference between this figure and Figure 19 shows the effectiveness of Phase II in the presence of cross traffic.

A simple solution is IP address spoofing. We modify the packet source IP address of one sender to the same as that of the other sender, in order to conceal RCV that all packets are from the same sender. RCV therefore places all packets to the same Rx queue. Although packets with a forged source IP address may be filtered by some firewall, this is a more practical solution compared with disabling either RSS or IC on RCV. We have successfully tested this solution on our campus network, Amazon EC2 [1], and PlanetLab [18].

Figure 23 shows the SNDa histogram when both RSS and IC are enabled. It is obtained with exactly the same testbed setting (including cross traffic, RSS, and IC) as Figure 14, except that the latter uses IP address spoofing. We can see that Figure 23 is greatly different from Figure 14. That is, without IP address spoofing, RSS and IC greatly change the packet arrival sequence.

Fig. 22. Impact of RSS and IC on the packet arrival sequence.

VII. EVALUATION

In this section, we evaluate PathComp using our lab testbed, our campus network, and Amazon EC2.

A. Testbed Results

We conduct the following three groups of testbed experiments to evaluate PathComp with default parameters. For each experiment, we run it for 50 times, and report the average with a 95% confidence interval. The emulated network topology is the same as the one shown in Figure 8 but with different link capacities. We use Linux tbf with the minimum token burst size to emulate a link capacity, except 100 Mbps, 1 Gbps, and 10 Gbps. We notice that Linux tbf on our testbed can only emulate up to 1.6 Gbps links due to limited system capability. Thus, the maximum link capacity in our testbed experiments is 1.6 Gbps, except 10 Gbps.

Group 1 - Impact of Large Capacity Ratios: This group of experiments study the accuracy of PathComp when two paths have a capacity ratio at least 2. For path a in Figure 8, we set C_{a1} = 500 Mbps, C_{a2} = 200 Mbps, C_{a3} = 1 Gbps, and thus the capacity of path a is C_a = C_{a2} = 200 Mbps. For path b, we set C_{b1} = 1.6 Gbps, C_{b2} = 400 Mbps to 1.6 Gbps,
$C_{45} = 10$ Gbps, and thus the capacity of path $b$ is $C_b = C_{34}$. Therefore, the capacity ratio $\gamma = C_b/C_a$ varies from 2 to 8.

The estimated capacity ratios are shown in Figure 24(a), where each link marked in Figure 8 has 30% cross traffic. We can see that PathComp can accurately measure these large capacity ratios. The large confidence interval at $\gamma = 8$ is partially because that Linux tbf has almost reached its max performance limit on our testbed.

Group 2 - Impact of Small Capacity Ratios: This group of experiments study the accuracy of PathComp when two paths have a capacity ratio no more than 2. Path $a$ has the same link capacities as in group 1, and thus the capacity of path $a$ is still $C_a = C_{12} = 200$ Mbps. For path $b$, we set $C_{34} = 1$ Gbps, $C_{35} = 200$ Mbps to 400 Mbps, $C_{45} = 1$ Gbps, and thus the capacity of path $b$ is $C_b = C_{34}$. Therefore, the capacity ratio $\gamma = C_b/C_a$ varies from 1 to 2.

The estimated capacity ratios are shown in Figure 24(b), where each link marked in Figure 8 has 30% cross traffic. We can see that PathComp can accurately measure these small capacity ratios. Even when $\gamma = 2$, the average estimated ratio is 1.88, and is very close to the actual ratio. Note that results with $\gamma = 2$ in Figures 24(a) and 24(b) are obtained using different link capacities (e.g., $C_{45}$) and then different amounts of cross traffic. In the latter, $C_{45}$ is smaller, and thus its link is more congested. This is why the estimation error with $\gamma = 2$ in the latter is larger than that in the former.

![Fig. 24. Impact of large and small capacity ratios.](image)

(a) Capacity ratio in range (2, 8) (b) Capacity ratio in range (1, 2)

Group 3 - Impact of Cross Traffic: This group of experiments study the accuracy of PathComp under different amounts of cross traffic. We use the same link capacities as in group 2, except that we set $C_{34}$ to 400 Mbps. Therefore, $\gamma = C_b/C_a$ is fixed to 2.

Figure 25(a) shows the estimated capacity ratios when the cross traffic on path $a$ varies from 10% to 60% and that on path $b$ is fixed to 30%. Figure 25(b) shows the estimated capacity ratios when the cross traffic on path $a$ is fixed to 30%, and that on path $b$ varies from 10% to 60%.

We can see that cross traffic on path $b$ (i.e., the faster path) has a bigger impact than that on path $a$ (i.e., the slower path). The reason is the probing traffic on path $b$ is sent at a higher rate. With the same percentage of crossing traffic, path $b$ is more congested than path $a$. For example, with 60% crossing traffic, the link utilization between $R4$ and $R5$ on path $b$ can reach up to $0.6 + 400/1000 = 100\%$, but only up to $0.6 + 200/1000 = 80\%$ for the link between $R2$ and $R5$ on path $a$. This is consistent with our observation in Section V, and this is also the motivation why PathComp in Phase II divides a long packet train into multiple short packet blocks on the faster path.

![Fig. 25. Impact of cross traffic.](image)

(a) Varying cross traffic on path $a$ (b) Varying cross traffic on path $b$

Remarks: We also run PathRate [6] on our testbed, which is one of the most well studied and widely used capacity estimation methods. However, it could not accurately estimate the capacity of a path on our testbed. For example, in Group 1, it reports a capacity of 1100~1400 Mbps (results of multiple runs) for path $a$, and reports an insufficient number of packet dispersion estimates for path $b$. This is partially due to interference of IC and Linux tbf.

B. Campus Network Results

We also evaluate PathComp using some servers in our campus network, where we know the network and server information.

Intra-Department Network: We choose three servers, denoted by $SNDa$, $SNDb$, and $RCV$, in our department. $SNDa$ is connected to the department 1 Gbps network through a 100 Mbps switch, and both $SNDb$ and $RCV$ are connected to the department network through 1 Gbps Ethernet. Figure 26(a) shows the ASND histograms of $SNDa$ and $SNDb$, and note that there are some ASND values at 9 and 11 packets which are caused by cross traffic. PathComp correctly estimates that the capacity ratio is 10 (corresponding to the peak at $\delta = 10$ packets). We also run PathRate, and it correctly estimates the capacity between $SNDa$ and $RCV$ as 100 Mbps, but it mistakenly reports the capacity between $SNDb$ and $RCV$ as 1900~2100 Mbps.

![Fig. 26. Campus network experiments.](image)

(a) Intra-Department Network (b) Inter-Department Network

Inter-Department Network: We choose three servers, denoted by $SNDa$, $SNDb$, and $RCV$, in three different departments in our campus network. $SNDa$ has a 100 Mbps NIC, and both $SNDb$ and $RCV$ have a 1 Gbps NIC. All three servers are connected to the campus 1 Gbps network. Each of the two paths passes four routers, and they share only the last router.

129
just before RCV. Figure 26(b) shows the ASND histograms of SNDa and SNDb, and PathComp correctly estimates that the capacity ratio is 10. We also run PathRate, and it correctly estimates the capacities of both paths: SNDa: 100 Mbps, and SNDb: 970–990 Mbps.

C. Amazon EC2 Results

We also evaluate PathComp using VMs on Amazon Elastic Compute Cloud (EC2) [1], which is a very popular public cloud computing platform. The EC2 facilities are located at multiple locations, and we choose the one in the US West (Oregon) region that includes three zones. We select three micro instances from different zones as three senders denoted by SNDa, SNDb, and SNDc, and we select one medium instance as the receiver RCV.

We relatively compare the path capacities from the three senders to the receiver for 100 times, and Figure 27 shows the cumulative distribution function (CDF) of the estimated capacity ratios. PathComp reports that SNDa is slightly faster than SNDb, SNDb is about 2.2–2.4 times faster than SNDc, and SNDa is about 2.4–2.7 times faster than SNDc. We can also see that the results are highly consistent. For example, among estimated ratios between SNDc and SNDa, most of them are about 2.4–2.7, and about 10% of them are smaller than 2.4. This is possibly due to the interference of VM scheduling, as micro instances are scheduled much more frequently than other types of instances.

![Graph showing estimated capacity ratios for different instances](image)

Fig. 27. Amazon EC2 Results.

In order to verify our estimated capacity ratios, we also run PathRate and iperf on EC2. PathRate reports that IC is detected and there is an insufficient number of packet dispersion estimates. Since this paper considers the capacity of a path that indicates the short-term peak rate of the path, we use the iperf(tcp highest 1-second throughput) in its first ten seconds. For SNDa, the iperf results are 540–980 Mbps. For SNDb, the iperf results are 530–760 Mbps. For SNDc, the iperf results are 280–290 Mbps. The iperf results are consistent with our estimated capacity ratios. We guess that the SNDa capacity is possibly 1 Gbps, and the SNDb and SNDc capacities are limited possibly by the virtual machine capability and by rate limiters (e.g., a token bucket shaper).

Note that PathComp sends out much less traffic than iperf. For example, PathComp sends less than 1 MBytes from SNDa, whereas iperf sends 65–117 MBytes just in the first second.

VIII. Conclusions

In this paper, we proposed a method called PathComp to relatively compare the capacities of two paths using the packet arrival sequence information instead of packet time information. In the future, we plan to extend PathComp to simultaneously compare more than two paths, and plan to relatively compare the available bandwidths of different paths.

ACKNOWLEDGMENT

The work reported in this paper is supported in part by NSF CNS-1017561.

REFERENCES