ICIM: An Inline Network Measurement Mechanism for Highspeed Networks

Cao Le Thanh Man, Go Hasegawa and Masayuki Murata
Graduate School of Information Science and Technology, Osaka University
1-3 Yamadagaoka, Suita, Osaka 560-0871, Japan
E-mail: {mlt-cao, hasegawa, murata}@ist.osaka-u.ac.jp

***Abstract*— In high-speed networks, such as 1-Gbps or higher networks, bandwidth measurement algorithms that utilize packet transmission/arrival intervals, such as packet trains and packet pairs, have a number of problems. First, network measurement for large bandwidth requires short packet transmission intervals, which causes a heavy load on the CPU. Second, network interface cards for high-speed networks usually employ Interrupt Coalescence (IC), which rearranges the arrival intervals of packets and causes bursty transmission of packets. In the present study, we introduce ICIM (Interrupt Coalescence -aware inline measurement), a new bandwidth measurement approach that overcomes these two problems. ICIM utilizes the data packets of an active TCP connection for the measurement. In order to determine the available bandwidth, rather than adjusting the packet transmission intervals, the TCP sender instead adjusts the number of packets involved in a burst and checks whether the inter-intervals of the bursts of corresponding ACK packets are increased or not. Simulation results show that ICIM can measure the bandwidth as high as some Gbps while requiring a number of data packets that is only 1/100 of that of the existing stream-based algorithm.**

I. Introduction

Active measurement of the available bandwidth of an end-to-end network path has been vigorously investigated [1-6]. Compared with passive measurement, active measurement can deliver faster and more accurate results because the network can be investigated in detail using probe traffic. However, the sending of probe traffic is a drawback of active measurement. According to [3], Pathload [1] generated between 2.5 to 10 MB of probe traffic per measurement. Newer tools have succeeded in reducing the amount of probe traffic. The average per-measurement probe traffic generated by IGI [2] is 130 KB and that generated by Spruce [3] is 300 KB. Although a few KB of probe traffic for a single measurement is a negligible load on the network, for routing in overlay networks, or adaptive control in transmission protocols, these measurements may be repeated continuously and simultaneously from numerous network nodes and end hosts. In such cases, probe traffic of a few KB per measurement will generate a large amount of traffic that may interfere with data transmission in the network, as well as degrading the measurement itself.

Previously, we proposed an active measurement method that overcomes the problem mentioned above [7]. We proposed the concept of *inline measurement*, that is, the idea of "plugging" the active measurement mechanism into an active TCP connection. This method has the advantage of requiring no extra traffic to be sent on the network, and provides fast and accurate measurement. We refer to RenoTCP employing this mechanism as ImTCP (Inline measurement TCP). When the sender transmits data packets, ImTCP adjusts the transmission rate of some packets, and considering arrival intervals of the corresponding ACK packets, the ImTCP sender estimates the available bandwidth. ImTCP utilizes a measurement algorithm similar to that of Pathload [1]. That is, the arrival intervals of packets that are sent back-to-back at a specified rate are used to estimate the available bandwidth. ImTCP delivers measurement results repeatedly in short intervals, such as a few RTTs, and the number of packets involved in each measurement is far fewer than that for Pathload.

In the present study, we focus on a new challenge regarding active measurement. Specifically, we investigate the bandwidth measurement of 1-Gbps or faster network paths, which are becoming increasingly popular. In such high-speed networks, ImTCP, Pathload and other active measurement tools based on packet spacing [2-5] must overcome the following problems. First, measurement in fast networks requires short transmission intervals of the probe packets (for example, 12 μs for a 1-Gbps link). However, regulating such short intervals causes a heavy load on the CPU. Second, network cards for high-speed networks usually employ Interrupt Coalescence (IC) [8, 9], which rearranges the arrival intervals of packets and causing bursty transmission, so that the algorithms utilizing the packet arrival intervals do not work properly.

We introduce a new inline measurement mechanism that works well in high-speed networks. We call this ICIM (Interrupt Coalescence-aware Inline Measurement). Unlike other active measurement tools, ICIM adjusts the number of packets that are transmitted in a burst caused by IC and estimates the available bandwidth by observing the number of packets in the burst as it passes through the network, rather than by observing the inter-intervals of the packets. ICIM does not set the sending interval of the packets, so the overhead for packet spacing at the sender is eliminated. The measurement results show that TCP with ICIM can transmit data with the same performance as Reno TCP and can measure the available bandwidth of high-speed networks.

The remainder of this paper is organized as follows. In Section 2, we discuss problems of measurement in high-speed networks and look at a number of related studies. In Section 3, we introduce ICIM and explain how to realize it in Reno TCP. In Section 4, we evaluate the performance of Reno TCP that

is utilizing ICIM. Finally, in Section 5, we present concluding remarks and discuss future projects.

II. Available bandwidth measurement in high-speed networks

In this section, we discuss some of the difficulties encountered by existing active available bandwidth measurement tools, including ImTCP, in high-speed networks (1 Gbps or higher). We assume that the machines that run the measurement tools are general purpose machines, for example, a x86-based CPU machine with a normal OS, such as 4.4 BSD or Gnu/Linux (or similar). The problems mentioned here may not occur in high-performance machines that are designed especially for measurement.

A. Limitation of packet pacing in general-purpose machines

In current active measurement tools, probe packets must be sent at a rate higher than the available bandwidth of the network path, otherwise the packet space will not be expanded and the tools will not be able to determine the available bandwidth. When the available bandwidth can reach 1 Gbps or higher, the transmission intervals of the probe packets must be 12 μs (for measuring 1-Gbps bandwidth) or smaller. As we discuss below, for a general-purpose machine, sending packets in such small intervals causes high CPU overhead.

For pacing packets, there are two approaches. The first is to continuously check the hardware clock (for example, using `gettimeofday()` in UNIX systems) and send the packets when the clock reaches a determined timing. In a Linux system with an x86-based CPU, one access of the hardware clock requires approximately 1.9 μs (in the FreeBSD system, one access requires 9 μs) [10]. The `write()` system call requires an average of 2 μs (in the case of a Pentium III CPU). Therefore, a Linux system can only send packets in intervals greater than 2 + 1.9 = 3.9 μs. This means that, the system can measure the bandwidth up to 3 Gbps (for the case in which the probe packet size is 1,500 Bytes). However, in order to send packets at 3 Gbps, the CPU has to spend most of the time checking the hardware clock overhead. If the measurement is repeated continuously, then the CPU will not be able to process tasks from other applications. The system performance then will be deteriorated. Thus, checking the hardware clock to send packets in a high-speed network is not a good approach.

The second approach is to register the packet sending program to an Interrupt Service Routine (ISR) of the hardware clock interrupt. In a general-purpose UNIX OS, the ISR `hardclock()` is provided for this purpose. In 4.4BSD OS and LINUX, the `hardclock()` system call is called by the interrupt of hardware clock every 0.01 s. However, with this low interrupt frequency, the program called by `hardclock()` can only send packets at the rate of 1.2 Mbps (assuming that the packet size is 1,500 Bytes). To obtain a higher interrupt frequency, a new interrupt schedule of the hardware clock can be implemented. However, one hardware interrupt (in 4.4 BSD OS) normally requires more than 1 μs [11]. If the packet transmission rate is 1 Gbps, then the sending interval is 12 μs. This means that, in this case, the overhead of the hardware interrupt is as high as 1/12 of the total working time of the CPU. In addition, a new interrupt schedule for the hardware clock requires many changes in the OS.

B. Effects of Interrupt Coalescence

Another reason for the difficulty in the task of measurement in high-speed networks is IC, which is deployed in most high-bandwidth Network Interface Cards (NICs). IC is a technique in which NICs group multiple packets that arrive in a short time interval and pass them to the OS in a single interrupt. IC reduces the CPU overhead when the arrival intervals of packets become small. Because the inter-arrival intervals of the packets observed by the kernel are changed, IC has an enormous impact on bandwidth measurement tools, in which the arrival intervals of packets are utilized for bandwidth estimation.

There are a number of types of timer setting in IC. For example, Intel Gigabit Ethernet Controllers [8] contains the following mechanisms for IC:

- Absolute timer: The absolute timer delays the assertion of an interrupt to allow the controller to collect additional interrupt events before delivering them to software.
- Packet timer: The packet timers are inactivity timers, triggering interrupts when the link has been idle for an appropriately long interval.
- Master timer for throttling all interrupt sources: An interrupt throttling mechanism is used to set an upper bound for the interrupt rate.

Under sustained loads, the absolute timers will be the primary source of device interrupts [8]. We investigate the absolute timers in greater detail. There are two absolute timers. One is for transmit interrupts, and the other is for receive interrupts. Because transmit interrupts only inform the kernel as to the completion of packet sending, delays in transmit interrupts do not affect the real transmission intervals of the packets. In contrast, delays in receive interrupts change the intervals of all receiving packets observed by the kernel. As shown in Figure 1, the receive absolute timer starts to count down upon receipt of the first packet. Subsequent packets do not alter the countdown. Once the timer reaches zero, the controllers generate an interrupt to pass all of the packets to the OS in a bursty manner. The length of the timer is decided by the parameter $RxAbsIntDelay$, which is defaulted to 0.1312 ms in Intel Gigabit Ethernet Controllers [12]. Thus, all packets that have time intervals smaller than $RxAbsIntDelay$ will belong either to the same burst, in which case the time interval between the packets becomes zero, or to two successive bursts, in which case the time interval becomes $RxAbsIntDelay$ or larger. Therefore, the software cannot detect packet intervals smaller than $RxAbsIntDelay$. With the default value of 0.1312 ms for $RxAbsIntDelay$, the software cannot perceive transmission rates larger than 100 Mbps (if the packet size is 1,500 Bytes).

Without IC, an OS interrupt occurs whenever a single packet arrives; this leads to a high CPU overhead when the system performs high speed data transmission. Therefore, we should not disable IC feature for the purpose of measurement. There are some studies that have discussed measuring bandwidth

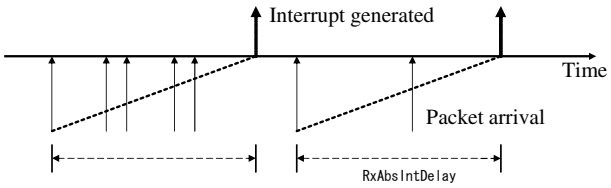


Fig. 1. Receive Absolute Timer

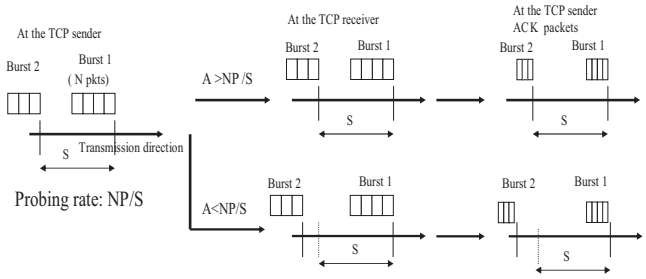


Fig. 2. Packet burst-based available-bandwidth measurement principle

using the existing IC. For example, one study [10] suggests that in order to obtain the real arrival intervals of packets, the onboard timestamp of some network cards (for example SysKonnect GigE NIC [9]) should be used. However, the same study also concludes that this solution is not useful for general-purpose network measurement tools, because very few NICs have an onboard timer. Furthermore, using an onboard NIC timer requires modification of the device driver. This prevents the tool from being easy to run on numerous systems.

Another study [13] reports that since the last packet in a burst formed by IC has the smallest delay in the NIC buffer, the intervals of the last packets in the bursts can be used for estimation of the available bandwidth, according to the Pathload [1] algorithm. However, because only a small part of stream is used for the measurement, the stream must be very long. This is not suitable in inline measurement, because making long measurement streams in TCP badly effects the TCP transmission performance.

III. Interrupt Coalescence-aware Inline Measurement (ICIM)

A. Effects of Interrupt Coalescence on TCP

The behavior of TCP when the network cards enable IC has been investigated in previous studies [11, 13], and IC has been shown to be detrimental to TCP self-clocking. IC causes the ACK packets to arrive at the sender in bursts, and this bursty arrival in turn causes bursty transmission of data packets and, subsequently, bursty transmission of ACK packets from the TCP receiver. According to one study [13], with IC, 65% of ACKs arrive with intervals of less than 1 μs, because they are delivered to the kernel with a single interrupt. Meanwhile, without IC, almost no ACK packets arrive with small intervals.

In the present study, we propose an algorithm that can exploit the burst of data packets in TCP under the effects of IC to measure the available bandwidth of the network path between TCP sender and receiver. The TCP sender adjusts the number of packets involved in a burst and checks whether the inter-intervals of the bursts of corresponding ACK packets are increased or not to investigate the available bandwidth. ICIM can be employed into any version of TCP. Using previously reported results [13], ICIM first checks to see if the network card has IC enabled. If the IC is enabled, ICIM continues measurement based on the bursty transmission of TCP.

B. Packet burst-based available-bandwidth measurement algorithm

Because the absolute timer (described in Section 2) is the primary source of device interrupts in the high speed transmission, we assume that the NIC uses the absolute timer when receiving packets. The measurement algorithm using bursts of packets is described below.

As shown in Figure 2, we consider the situation in which two bursts of packets are sent at the interval S. The number of packets in the first burst (Burst 1) is N. Assume that C is the capacity of the bottleneck link. C_{Cross} is the average transmission rate of cross traffic over the bottleneck link when the two bursts pass the link, and P is the packet size. Then, the amount of traffic that enters the bottleneck link during the period from the point at which the first packet of Burst 1 reaches the link until the point at which the first packet of Burst 2 reaches the link will be the sum of the packets in Burst 1 and the cross traffic packets arriving in S, i.e., $C_{Cross} \cdot S + N \cdot P$. If the amount is larger than the transfer ability of the link during this period, considered to be $C \cdot S$, then Burst 2 will go to the buffer of the link. This results in a tendency for the interval between the two bursts to increase after leaving the bottleneck link.

We can write that the burst interval will be increased if

$$C_{Cross} \cdot S + N \cdot P > C \cdot S \tag{1}$$

or,

$$\frac{N \cdot P}{S} > C - C_{Cross}$$

Note that $C - C_{Cross}$ is the available bandwidth (A) of the bottleneck link. Therefore, Eq. (1) becomes

$$\frac{N \cdot P}{S} > A$$

Since we assume that the absolute timer is used, S is always larger than $RxAbsIntDelay$. Therefore, at the NIC of the TCP receiver, since the arrival interval of the two bursts are larger or equal to S, the two bursts are passed to the kernel in two different interrupts. The TCP receiver then sends the ACK of the two bursts in the same intervals to the sender TCP. Thus, by checking the arrival intervals of the corresponding ACK packets of the two bursts, the TCP sender can determine if $A > NP/S$. By sending numerous bursts with various values of NP/S (by changing N), we can search for the value of the available bandwidth A. This is the measurement principle of the proposed inline measurement mechanism.

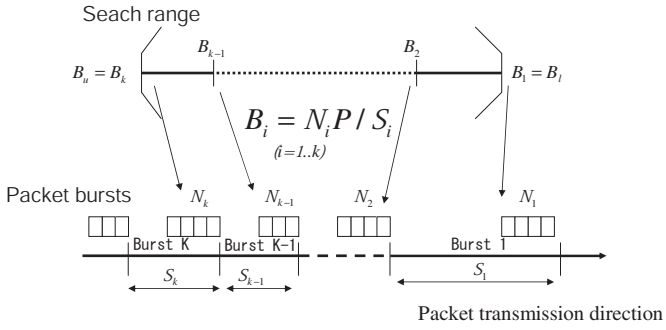


Fig. 3. Probing a search range in ICIM

C. ICIM

ICIM inherits the concept of the *search range* from the measurement algorithm in ImTCP [7]. This is the idea of limiting the bandwidth measurement range using statistical information from previous measurement results rather than searching from 0 bps to the upper limit of the physical bandwidth for every measurement. By limiting the measurement range, we can keep the number of probe packets small.

At first, we explain how to search for the available bandwidth in a determined search range and then we present an overview of the measurement algorithm.

Assume that the search range for a measurement is (B_l, B_u). The algorithm then check k values in the range to determine which is nearest to the real available bandwidth. We use $k = 4$ in the following simulations. The k points are:

$$B_i = B_l + \frac{B_u - B_l}{k-1}(i-1) \;\; (i = 1, ..., k)$$

The TCP sender then sends k consequence bursts and the number of packets are adjusted so that the probe rate of Burst i is B_i:

$$\frac{N_i \cdot P}{S_i} = B_i \tag{2}$$

We illustrate the setting in Figure 3.

Realization of Eq. (2) requires the following:

- The value of S_i must be estimated at the timing of the transmission of Burst i. In fact, S_i is unknown until Burst $i+1$ is transmitted. But we need the value at the timing of the transmission of Burst i in order to guarantee Eq. (3). We therefore estimate the value of S_i by assuming that the amount of data in Burst i is proportional to the length of the interval as follow:

$$S_i = \frac{N_i \cdot P}{T} \tag{3}$$

  where T is the average throughput of TCP.
- In case the number of packets in Burst i is smaller than N_i, additional packets must be added to the burst so that the packet number becomes N_i. ICIM utilizes a buffer located at the bottom of the TCP layer in order to store the packets temporarily before sending them to the IP layer, in the manner of ImTCP. ICIM stores all of the packets of the burst that preceded Burst 1 in the buffer. Packets are added to Burst i $(i = 1..k)$ when necessary in order to maintain the desired number of packets (N_i) in these bursts.

  ICIM sends k bursts and checks the corresponding ACK of the bursts. If from burst number j, $j = 1..k$, the arrival interval of the bursts becomes larger, then B_j is considered to be the value of the available bandwidth in that measurement. Here, the burst interval is consider to become larger if the arrival interval is larger then λ times of the sending interval. We set λ to 1.01 in the following simulations.

ICIM first checks whether IC is enabled for the network card. For the reasons explained in Subsection 3.1, ICIM checks the arrival intervals of the ACK packets. If more than 50% of the intervals are less than 1 μs, then ICIM decides that IC is enabled. If the IC is enabled, then ICIM continues the following measurement steps. Otherwise, the measurement algorithm introduced in ImTCP is used.

The measurement algorithm of ICIM is as follows:

1) Set the initial search range
   We set the initial search range as $(T, 2 \cdot T)$ where T is the throughput of TCP.
2) Search for the available bandwidth in the decided search range.
   ICIM waits until the window size ($cwnd$) is larger than C_{min} (large enough to create bursts for measurement). We use $C_{min} = 50$ in the following simulations. Data packets are then sent in order to search the available bandwidth in the decided search range, as described above.
3) Add the new measurement result to the database and calculate the new search range.
   The measurement result in the last step is added to a dabatase of measurement results. We then calculate the new search range (B'_l, B'_u) from the database. We use the 95% confidential interval of the data stored in the database as the width of the next search range, and the current available bandwidth is used as the center of the search range. The search range is calculated as follow:

$$B'_l = R - \max\left(1.96\frac{V}{\sqrt{q}}, \frac{R}{10}\right)$$

$$B'_u = R + \max\left(1.96\frac{V}{\sqrt{q}}, \frac{R}{10}\right)$$

   where R is the latest measurement result. V is the variance of stored values of the available bandwidth and q is the number of stored values. $R/10$ is a value that ensures that the search range does not become too small. Moreover, when measurement result in Step 3 falls to B_l (B_u), it is possible to consider that the network has changed greatly so that the real value of the available bandwidth is lower (higher) than the search range. In this case, we discard the accumulated measurement results because they become unreliable as statistic data and enlarge the search range (B_l, B_u) twice towards the lower (higher) direction to create (B'_l, B'_u).
4) Wait for Q seconds then return to Step 2 and start the next measurement. During the waiting time Q, TCP

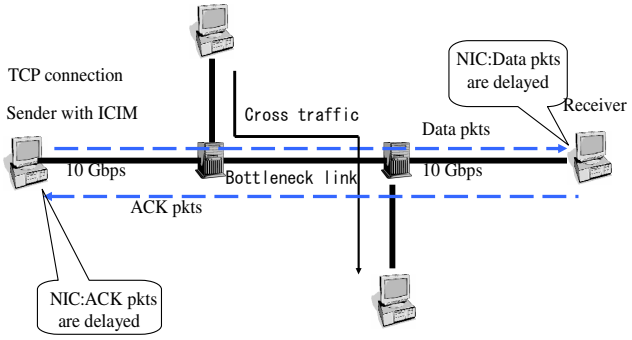


Fig. 4. Simulation topology

TABLE I
DISTRIBUTION OF PACKET SIZE OF CROSS TRAFFIC.

Packet size (Bytes)	Proportion of bandwidth (%)
28	0.08
40	0.51
44	0.22
48	0.24
52	0.45
552	1.10
576	16.40
628	1.50
1420	10.50
1500	37.10
40-80 (range)	4.60
80-576 (range)	9.60
576-1500 (range)	17.70

transmits packets in the normal manner. The waiting time is needed for the TCP transmission to return to the normal state after the packets store-and-forward process at Step 2.

IV. SIMULATION EXPERIMENT

A. Measurement results

We show the measurement results for ICIM through ns-2 [14] simulations. We implement ICIM via Reno TCP, the most popular version of TCP, and use the topology shown in Figure 4 for the simulation. The sender and receiver of TCP are connected through 10-Gbps access links and a bottleneck link. The NICs of both the sender and receiver host employ IC with an absolute timer. The value of $RxAbsIntDelay$ used in NIC is 0.000132 (the default value). The cross traffic on the bottleneck link is made up of UDP flows in which various packet sizes are used, according to results monitored on the Internet [15], as shown in Table I. The capacity of the bottleneck link is 5 Gbps, and the available bandwidth (A-bw) is 2 Gbps (from 0 to 15 sec), 3 Gbps (from 15 to 35 sec) and 4 Gbps (from 35 to 50 sec).

Figures 5(a) and 5(b) show the measurement results for ICIM when the interval between two measurements is set to one RTT or two RTTs, respectively. Also shown are the search ranges for each measurements. The search ranges, in most cases, successfully cover the correct value of the A-bw. Therefore, ICIM can quickly detect the A-bw, even in such a high-speed network. When $Q = 1$, the throughput of TCP oscillates slightly, the estimation of the burst interval in Eq. (3) becomes incorrect. Therefore, the probing rate of each Burst i may not be exactly equal to B_i (in Step 2 of Section 3.3). This leads to a large dispersion of the measurement results in Figure 5(a). When $Q = 2$, the TCP sender creates fewer packet bursts so that the measurement results are nearer to the correct value of the A-bw, as shown in Figure 5(b). However, the measurement frequency (16.7 results/second) becomes half of that when $Q = 1$ (34.2 results/second)

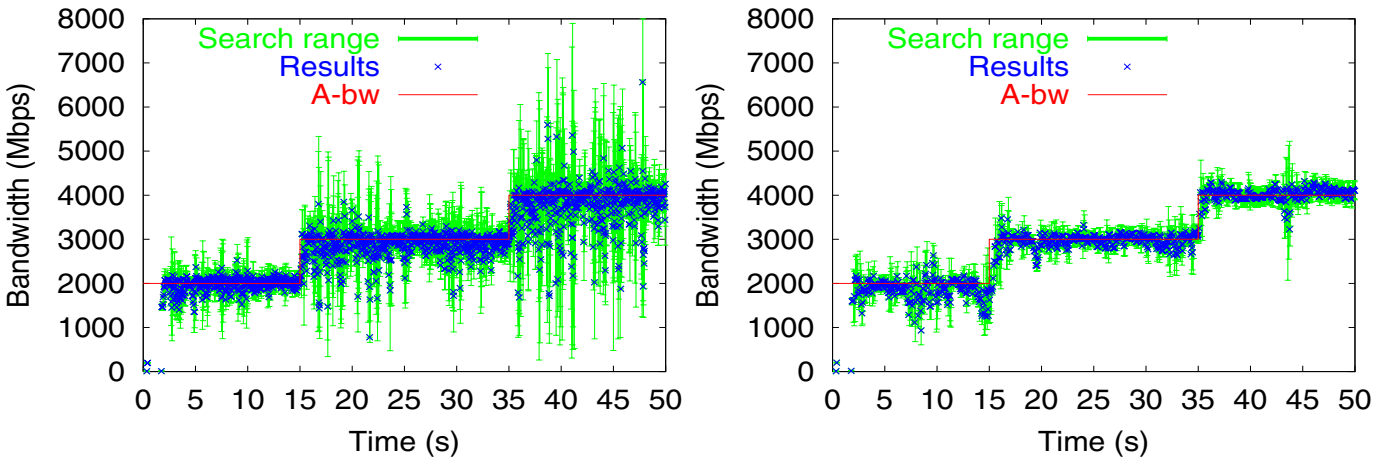
B. Comparison with IC-aware Pathload

For the comparison between ICIM and Pathload, the TCP sender and receivers are next replaced by the sender and receiver of Pathload. We used the version of Pathload that can detect and filter the effects of IC [13]. To make the measurement of Pathload faster, we set the starting probing rate to 200 Mbps (instead of the default setting of 1 Mbps). In addition, ω and χ are set to 200 Mbps and 150 Mbps, respectively, and the size of probing packets is set to 1,500 bytes.

The measurement results of Pathload when the number of packets in a stream K is set to 160 are shown in Figure 6(a). Because the default value of $RxAbsIntDelay$ used in NIC is 0.000132 (s) and the packet size is 1,500 bytes, the average number of packets in a burst is 22 when the A-bw is 2 Gbps, 33 when the A-bw is 3 Gbps and 44 when the A-bw is 44 Gbps. Therefore, when $K = 160$, there are approximately nine bursts in each stream when the A-bw is 2 Gbps. This means that Pathload has approximately nine packets (the last packet in the bursts) for measurement. The increasing trend in the stream in this case can be well determined so Pathload can deliver good measurement results. However, when the A-bw becomes 3 Gbps or greater, the number of bursts becomes approximately six or fewer. Then, Pathload does not have enough packets to detect well the increasing trend in the stream. Therefore, as shown in Figure 6(a), Pathload fails to deliver good measurement results when the bandwidth is equal to or greater than 3 Gbps.

Figure 6(b) shows the measurement results of Pathload when K is set to 200. In this case, Pathload has a sufficient number of packets for detecting the increasing trend of streams. Therefore, the measurement results are correct. However, since Pathload searches for the A-bw from a low value, a long time is required to yield one result. The measurement frequency is only 0.28 results/second, which is 60 times smaller than that of ICIM (with Q = 2 RTTs).

Figure 6(b) shows that, if the A-bw changes during a measurement, Pathload may not detect the change well. At 15 seconds, the A-bw changes from 2 Gbps to 3 Gbps while Pathload is probing a rate smaller than 2 Gbps. When the probing rate reaches 2 Gbps, the A-bw is already changed, therefore Pathload can successfully detect the value of 3 Gbps. However, at 35 seconds, the probing rate of the ongoing measurement reaches 3 Gbps before the change in the A-bw from 3 to 4 Gbps, so Pathload assumes that the A-bw is smaller than or equal to 3 Gbps. Therefore, Pathload delivers a value of approximately 3 Gbps at the end of that measurement, which is far from the value of the A-bw at this timing.



(a) Measuring intervals $Q = 1RTT$

(b) $Q = 2RTTs$

Fig. 5. Measurement results for ICIM

TABLE II
NUMBER OF PACKETS REQUIRED FOR A MEASUREMENT

$A-bw$	ICIM	IC-aware Pathload	Ratio ICIM:Pathload
2 Gbps	110	$200 \cdot 12 \cdot 8 = 19\ 200$	0.006
3 Gbps	130	$200 \cdot 12 \cdot 9 = 21\ 600$	0.006
4 Gbps	154	$200 \cdot 12 \cdot 10 = 24\ 000$	0.006

Table II compares the number of packets used in the measurement of ICIM, and Pathload. ICIM sends four bursts of packets for each measurement. The average number of total packets in four bursts are shown in the second column of the table. On the other hand, Pathload probes 8, 9 and 10 times for one measurement result when the A-bw is 2, 3 and 4 Gbps, respectively. Each probe requires 12 streams, the number of packets of which is 200. We can see that the number of packets used by ICIM is less than one percent of that of Pathload.

Figures 5 and 6 show that the measurement results of ICIM have a larger dispersion compared to Pathload because, based on the nature of the algorithm, ICIM cannot increase the length of each measurement burst to obtain high accuracy, as Pathload does. Instead, the accuracy can be improved by taking the exponential moving average in suitable intervals.

C. Measurement results in Web traffic environment

We next investigate the measurement results for ICIM in the network model depicted in Figure 4. Cross traffic is now changed to Web traffic involving a large number of active Web document accesses. We use a Pareto distribution for the Web object size distribution with 1.2 as the Pareto shape parameter and 12 KBytes as the average object size. The number of objects in a Web page is 20. The capacity of the bottleneck link is set to 1Gbps. The access links are also set to 1Gbps.

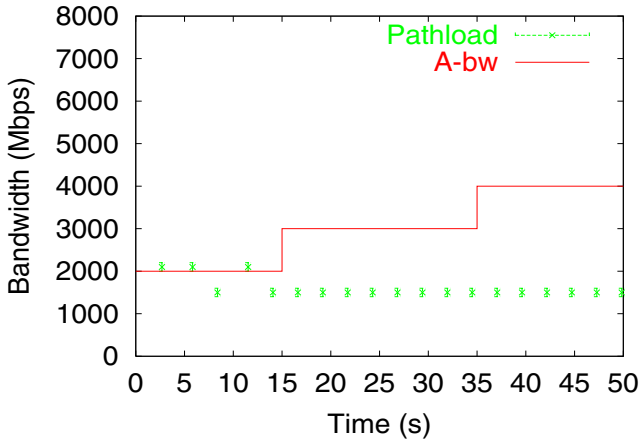
The available bandwidth is calculated as the capacity of the bottleneck link minuses the total amount of Web traffic passing the link. Figure 7(a) shows the changes of available bandwidth and the average measurement results for each second. ICIM under-estimates the available bandwidth a little because the cross traffic, composed of so many connections, arrives at the bottleneck link in a bursty fashion. The burst of cross traffic may enlarge the intervals of the measurement bursts of ICIM even when the probing rate is still lower than the average available bandwidth. However, the measurement results deviate only a litle from the correct values and in general they can follow the changes of available bandwidth.

Figure 7(b) shows the measurement results for IC-aware Pathload in the same environment. We set K to 160 and the starting probing rate to 100Mbps and ω and χ are both set to 50 Mbps. Overall, the results have a trend of over-estimation. We think that the problem can be solved if we adjust the PCT/PDT thresholds of Pathload appropriately, instead of using the default values. Figure 7(c) shows the measurement of normal Pathload. Because the probe packets are grouped at the NIC, the increasing trend in the measurement streams becomes difficult to discover. Therefore, Pathload over-estimates in most of the time.

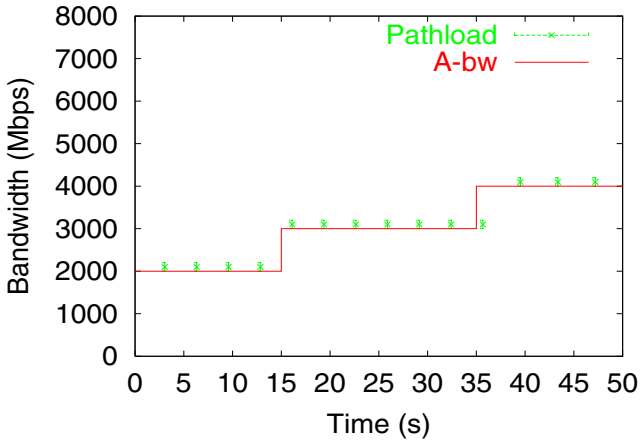
D. TCP compatibility

We finally examine the data transmission performance of Reno TCP when it employs ICIM. We perform a simulation where a number of Reno TCP connections that have ICIM conflict with the same number of Reno TCP connections that do not have ICIM through a 1 Gbps bottleneck link, as shown in Figure 8. All the connections have the same RTT (0.018 s) and the same access link's bandwidth (10 Gbps). The number of connections is set to 4, 8 and 12. For each value of connection numbers, simulation is repeated 10 times, and the throughputs of the TCP connections that have and do not have ICIM (and the ratio of thereof) are calculated and compared.

Table III shows the results when Q of ICIM is set to 1 RTT and 2 RTTs. In case ICIM performs measurement in every RTT, the TCP achieves lower throughput than TCP that does



(a) Number of packets in a stream $K = 160$ packets



(b) $K = 200$ packets

Fig. 6. Measurement results for IC-aware Pathload

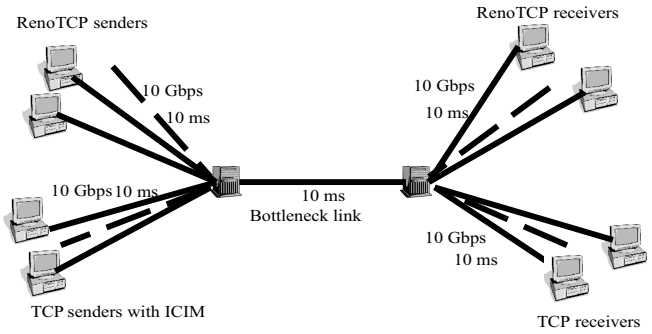


Fig. 8. Simulation topology for examining TCP compatibility

TABLE III
THROUGHPUT (MBPS) OF RENO TCP USING ICIM: NORMAL RENO TCP (RATIO)

#connections	Q=1 RTT	Q=2 RTTs
4	466.4 : 490.6 (0.95:1)	483.7 : 475.6 (1.01:1)
8	451.1 : 544.4 (0.82:1)	505.1 : 490.5 (1.02:1)
12	418.7 : 577.7(0.72:1)	503.5 : 493.2 (1.02:1)

not perform ICIM when conflicts occur because ICIM has to delay several data packets for measurement in this case. As shown in Table III, the ratio of throughput between TCP with ICIM compared to RenoTCP is always less than 1. When the number of connections increases, the ratio is lower because conflicts between TCP connections are more intense. If ICIM takes a lower measurement frequency, for example, when Q = 2 RTTs, then the TCP connections performing ICIM can obtain the same throughput as normal Reno TCP, as shown in the third column of the table.

V. CONCLUSION AND FUTURE STUDIES

In the present paper, we introduced ICIM, a new method that can measure the available bandwidth in a 1-Gbps or higher network path. The proposed measurement algorithm does not require regulation of packet transmission intervals and works well with Interrupt Coalescence. Simulation experiments showed that the proposed measurement algorithm works well with no degradation of TCP data transmission speed.

At present, we are evaluating the performance of ICIM in a real network environment. In addition, we are investigating the measurement mechanism for the capacity of high-speed networks that can be implemented in ICIM with the least change.

REFERENCES

[1] M. Jain and C. Dovrolis, "End-to-end available bandwidth: Measurement methodology, dynamics, and relation with TCP throughput," in *Proceedings of ACM SIGCOMM 2002*, Aug. 2002.

[2] N.Hu and P.Steenkiste, "Evaluation and characterization of available bandwidth probing techniques," *IEEE Journal on Selected Areas in Communications*, vol. 21, Aug. 2003.

[3] J.Strauss, D.Katabi and F.Kaashoek, "A measurement study of available bandwidth estimation tools," in *Proceedings of Internet Measurement Conference 2003*, Oct. 2003.

[4] V. Ribeiro, R. Riedi, R. Baraniuk, J. Navratil and L. Cottrell, "PathChirp: Efficient available bandwidth estimation for network paths," in *Proceedings of Passive and Active Measurement Workshop*, 2003.

[5] J. Navratil and R. Cottrell, "ABwE: A practical approach to available bandwidth estimation," in *Proceedings of the 4th Passive and Active Measurement Workshop PAM 2003*, Apr. 2003.

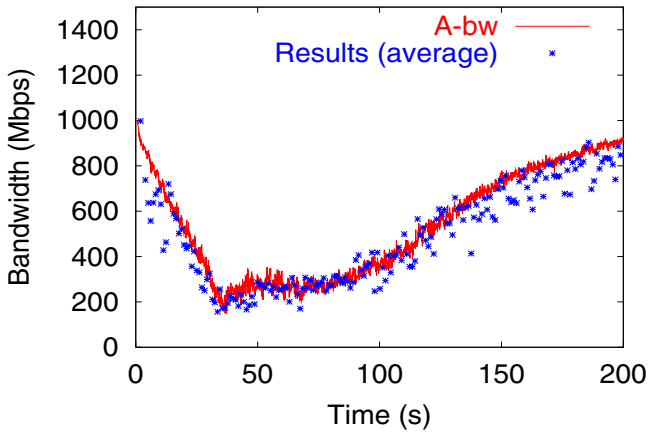
[6] A. Shriram, M. Murray, Y. Hyun, N. Brownlee, A. Broido, M. Fomenkov and k claffy, "Comparison of public end-to-end bandwidth estimation tools on high-speed links," in *Proceedings of the 6th Passive and Active Measurement Workshop PAM 2005*, Mar. 2005.

[7] Cao Le Thanh Man, Go Hasegawa and Masayuki Murata, "Available bandwidth measurement via TCP connection," in *Proceedings of the 2nd Workshop on End-to-End Monitoring Techniques and Services E2EMON*, Oct. 2004.

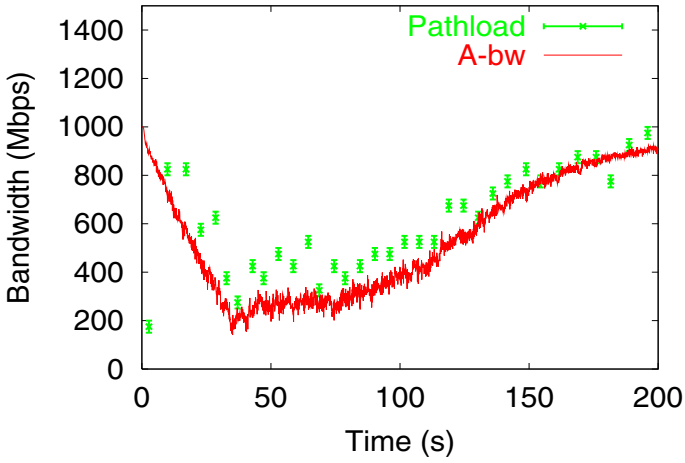
[8] Intel, "Interrupt Moderation Using Intel Gigabit Ethernet Controllers," available at `http://www.intel.com/design/network/applnots/ap450.pdf(2003)`.

[9] Syskonnect, "SK-NET GE Gigabit Ethernet Server Adapter," available at `http://www.syskonnect.com/syskonnect/technology/SK-NET_GE.PDF(2003)`.

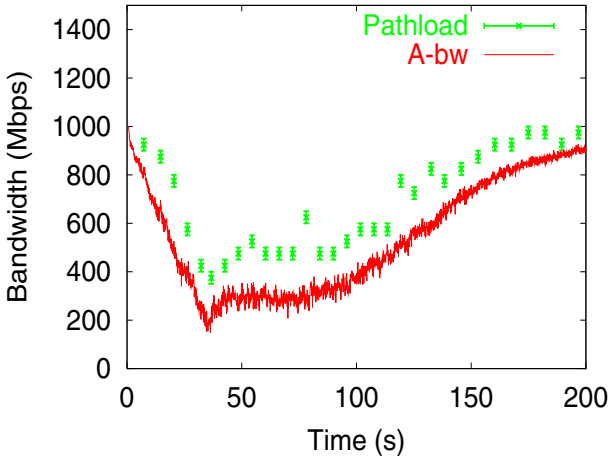
[10] G.Jin and B.Tierney, "System capability effect on algorithms for network bandwidth measurement," in *Proceedings of Internet Measurement Conference 2003*, Oct. 2003.



(a) Average measurement results of ICIM for each second



(b) Measurement results for IC-aware Pathload. $K = 160$



(c) Measurement results for normal Pathload. $K = 200$

Fig. 7. Measurement results in Web traffic environment

[11] M. Zec, M. Mikuc and M. Zagar, "Estimating the impact of interrupt coalescing delays on steady state TCP," in *Proceedings of the 10th SoftCOM 2002 conference*, 2002.

[12] Intel(R) PRO/1000 Adapter, "README file," available at `http://support.intel.co.jp/jp/support/network/adapter/1000/linux_readme%.htm`.

[13] R. Prasad, M. Jain and C. Dovrolis, "Effects of interrupt coalescence on network measurements," in *Proceedings of the 5th Passive and Active Measurement Workshop PAM 2004*, Apr. 2004.

[14] NS Home Page, "`http://www.isi.edu/nsnam/ns/`,"

[15] "NLANR web site," available at `http://moat.nlanr.net/Datacube/`.