Inline bandwidth measurement techniques for gigabit networks

Cao Le Thanh Man*

Graduate School of Information Science and Technology,
Osaka University, T5600043 Osaka-Fu,
Toyonaka-Shi, Machikaneyamacho 1-32, Japan
Fax: 0081-6850-6868
E-mail: mlt-cao@ist.osaka-u.ac.jp
*Corresponding author

Go Hasegawa

Cybermedia Center,
Osaka University, T5600043 Osaka-Fu,
Toyonaka-Shi, Machikaneyamacho 1-32, Japan
Fax: 0081-6850-6868
E-mail: hasegawa@cmc.osaka-u.ac.jp

Masayuki Murata

Graduate School of Information Science and Technology,
Osaka University, T5600043 Osaka-Fu,
Toyonaka-Shi, Machikaneyamacho 1-32, Japan
Fax: 0081-6850-6868
E-mail: murata@ist.osaka-u.ac.jp

Abstract: We introduce an inline measurement method that can overcome difficulties in measurement task in high-speed network environment, such as short packet transmission intervals and Interrupt Coalescence function deployed in the high-speed Network Interface Cards. The method adjusts the number of packets involved in a packet burst of an active TCP connection, and utilises the inter-intervals of the bursts of the corresponding ACK packets for bandwidth measurement. Experiment results show that the proposed inline measurement method can measure the bandwidth in the network paths at least 1-Gbps or higher.

Keywords: gigabit network; available bandwidth; capacity; TCP; inline measurement; Interrupt Coalescence; IC.

Reference to this paper should be made as follows: Man, C.L.T., Hasegawa, G. and Murata, M. (xxxx) 'Inline bandwidth measurement techniques for gigabit networks', *Int. J. Internet Protocol Technology*, Vol. x, No. x, pp.xxx–xxx.

Biographical notes: Cao Le Thanh Man received the ME and DE Degrees from Graduate School of Information Science and Technology, Osaka University, Japan, in 2004 and 2007, respectively. He is now working as a research scientist at Systems Development Laboratory, Hitachi Ltd. His research interests include network performance measurement and evaluation, TCP protocol design and evaluation. He is a member of IEEE.

Go Hasegawa received the ME and DE Degrees in Information and Computer Sciences from Osaka University, Osaka, Japan, in 1997 and 2000, respectively. From July 1997 to June 2000, he was a research assistant of Graduate School of Economics, Osaka University. He is now an Associate Professor of Cybermedia Center, Osaka University. His research work is in the area of transport architecture for future high-speed networks. He is a member of the IEEE and IEICE.

Masayuki Murata received the ME and DE Degrees in Information and Computer Sciences from Osaka University, Japan, in 1984 and 1988, respectively. He is now a Professor of Graduate School of Information Science and Technology, Osaka University. He has more than 200 papers of international and domestic journals and conferences. His research interests include computer communication networks, performance modelling and evaluation. He is a member of IEEE, ACM, The Internet Society, IEICE and IPSJ.

1 Introduction

Current efforts to improve TCP performance over high-speed networks can be divided into two main approaches. The first one changes the Additive-Increase-Multiplicative-Decrease (AIMD) congestion control of Reno TCP toward a faster increase of window size when there is no congestion and a smaller decrease when packet loss occurs. The representatives of this approach are High-Speed TCP (Floyd, 2003), Scalable TCP (Kelly and Ott, 2002) and Binary Increase Congestion (BIC) TCP (Xu et al., 2004). They can perform better than Reno/NewReno TCP in high-speed networks. However, these TCP versions do not have optimal speed for increasing the window size for all network environments because they still rely on network feedback, that is, a packet loss event, for adjustment of the speed. The packet loss event requires TCP to retransmit a certain amount of data, which can be large in high-speed networks. Therefore, these TCP variants cannot fully utilise the bandwidth of a network path. The second approach tries to avoid packet loss by deploying a delay-based congestion control mechanism. This approach uses queuing delay as a multi-bit congestion signal. The representative of this approach is FAST TCP (Jin et al., 2004). The difficulty with this approach is matching the delay with the congestion signal because the relation is affected by the capacity of the network path. A mismatch may cause oscillation of the congestion window size, leading to serious performance degradation.

The effective way for transport protocols to achieve the desired throughput on a high-bandwidth and long-delay network path is to observe the available bandwidth/capacity of an end-to-end network and adapt accordingly. Active measurement of the available bandwidth/capacity of an end-to-end network path has been vigorously investigated (Jain and Dovrolis, 2002; Hu and Steenkiste, 2003; Strauss et al., 2003; Ribeiro et al., 2003; Kapoor et al., 2004). 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 Strauss et al. (2003), Pathload (Jain and Dovrolis, 2002) generated between 2.5 MB 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 (Hu and Steenkiste, 2003) is 130 KB and that generated by Spruce (Strauss et al., 2003) 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.

We have proposed an active measurement method that overcomes the problem mentioned above (Man et al., 2004). 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. When the sender transmits data packets, TCP adjusts the transmission rate of some packets, and considering arrival intervals of the corresponding ACK packets, the TCP sender estimates the available bandwidth/capacity of the network path between the sender an the receiver of the TCP connection.

Inline measurement results enable TCP to optimise its bandwidth utilisation. Our research group has previously proposed a congestion control mechanism based on a logistic equation and a Lotka-Volterra competition model (Hasegawa and Murata, 2006), by which TCP can fully utilise the bandwidth when the available bandwidth and capacity of the network path are provided. We have also proposed a new TCP version that sets the upper limit of the congestion window based on the results of the inline network measurement (Tsugawa et al., 2006). In this case, TCP can provide background data transfer without affecting the foreground traffic, whereas previous methods cannot avoid network congestion essentially. Another study (Marcondes et al., 2006) shows the performance of TCP by using the capacity information for congestion control in the Internet. The performance of the TCP proposed by Marcondes et al. (2006) is much better than that of Reno TCP, while the fairness with Reno TCP is still maintained. Besides its use as a part of congestion control mechanism, inline measurement can be used in many other caces, such as for routing or server selection in grid or overlay networks.

In this study, we focus on the problem: Can TCP still perform inline measurement in Gbps-level bandwidth? The problem arises because even stand-alone active measurement tools such as Pathload and CapProbe still cannot work well in a Gbps network for two reasons. First, measurement in fast networks requires short transmission intervals of the probe packets (for example, 12 μs for a 1-Gbps link and 1500-byte packet). However, regulating such short intervals causes a heavy CPU load. Second, network cards for high-speed networks usually employ IC (Intel, 2003; Syskonnect, 2003), which rearranges the arrival intervals of packets and causes bursty transmission, and, therefore, algorithms utilising packet arrival intervals do not work properly.

We introduce a new inline measurement mechanism that works well in high-speed networks. We call this Interrupt Coalescence-aware Inline Measurement (ICIM). Unlike other active measurement tools which observe the inter-intervals of the packets, ICIM adjusts the number of packets that are transmitted in a burst caused by IC and estimates the capacity and available bandwidth by checking whether the inter-intervals of the bursts of corresponding ACK packets are increased or not as they pass through the network. ICIM does not set the sending interval of the packets, so the overhead for packet spacing at the sender is eliminated.

The contributions of this study are as follows.

- We propose Interrupt Coalescence-aware Inline Measurement for available bandwidth (ICIM-abw), an inline measurement algorithm for available bandwidth that works in a gigabit network.
- We validate the measurement results of ICIM-abw in many simulation scenarios. The results show that the algorithm works well in a 1-Gbps and faster network.
- We introduce Interrupt Coalescence-aware Inline Measurement for capacity (ICIM-cap), an algorithm for inline measurement of the capacity in a gigabit network. Unlike current measurement algorithms, the ICIM-cap algorithm works well in extremely high-load networks. However, the algorithm can work well only when the tight link of the path, which has the smallest available bandwidth, is identical to the bottleneck link, which has the smallest capacity. We then discuss the range of errors if this supposition is not true, and show that the range of errors is small.
- We implement ICIM-abw in the FreeBSD system to test its performance in a laboratory environment. The measurement results show that ICIM-abw can work well in a real system.

The remainder of this paper is organised as follows. In Section 2, we discuss the problems of bandwidth measurement in high-speed networks and look at a number of related studies. In Section 3, we introduce ICIM-abw and explain how to realise it in Reno TCP. We then evaluate the performance of Reno TCP that is utilising ICIM-abw. In Section 4, we introduce ICIM-cap and discuss the range of errors that may occur with this algorithm. In Section 5, we validate the performance of ICIM-abw in a real environment. Finally, in Section 6, we present concluding remarks and discuss future projects.

2 Bandwidth measurement in high-speed networks

In this section, we discuss some of the difficulties encountered by existing active available bandwidth/capacity measurement tools 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 network bandwidth measurement.

2.1 Limitation of packet pacing in general-purpose machines

In current active measurement tools, probe packets must be sent at a rate higher than the bandwidth of the network path, otherwise the packet space will not be expanded and the tools will not be able to determine the bandwidth. When the bandwidth reaches 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) (Jin and Tierney, 2003). 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 2.4 LINUX kernel, the `hardclock()` system call is called by the interrupt of hardware clock every 0.01 s. In the 2.6 LINUX kernel, it is called every 0.001 s. However, with this low interrupt frequency, the program called by `hardclock()` can only send packets at the rate of up to 12 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 (Zec et al., 2002). 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.

2.2 Effects of Interrupt Coalescence (IC)

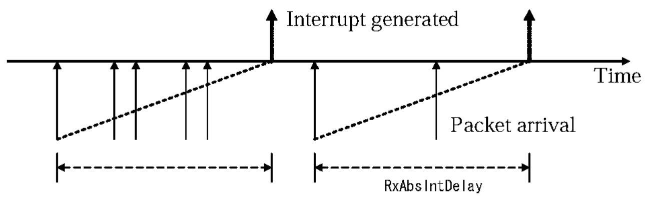
Another reason for the difficulty in the task of measurement in high-speed networks is IC, which is deployed in most high-bandwidth NICs. IC is a technique in which NICs group multiple packets that arrive in a short time interval and pass them to the OS in a single interrupt. IC reduces the CPU overhead 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 utilised for bandwidth estimation.

There are a number of types of timer setting in IC. For example, Intel Gigabit Ethernet Controllers (Intel) 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 (Intel). 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 (Intel(R) PRO/1000 Adapter. README file). 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).

Figure 1 Receive absolute timer



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 using the existing IC. For example, one study (Jin and Tierney, 2003) suggests that in order to obtain the real arrival intervals of packets, the onboard timestamp of some network cards (for example SysKonnect GigE NIC (Syskonnect)) 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 (Prasad et al., 2004) 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 (Jain and Dovrolis, 2002) 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 affects the TCP transmission performance.

2.3 Bursty transmission in TCP

The behaviour of TCP when the network cards enable IC has been investigated in previous studies (Zec et al., 2002; Prasad et al., 2004), 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 (Prasad et al., 2004), 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/capacity 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 bandwidth. ICIM can be employed into any version of TCP. Using previously reported results (Prasad et al., 2004), 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.

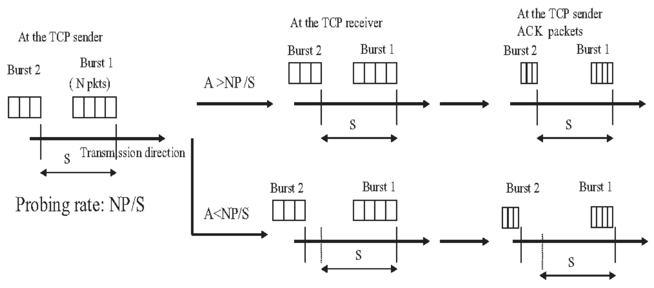
3 ICIM-abw: Interrupt Coalescence-aware Inline Measurement for available bandwidth

3.1 Packet burst-based available bandwidth measurement algorithm

The basic idea of measuring bandwidth larger than 1-Gbps is that, we consider a burst of packets as a big packet. Two consecutive big packets are then treated as a packet pair, of which the time interval in large enought so that the general purpose OS can read exactly. The measurement algorithm is described below. Though the algorithm can work somehow when Delayed ACK in the TCP receiver is on, we recommend the Delayed ACK is off so that the algorithm can works properly. 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.

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. We suppose that C_{Cross} is not changed much by the TCP connection performing inline measurement. Otherwise, the available bandwidth is not determined; the measurement becomes meaningless. 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., $C_{\text{Cross}}\, S + N \cdot P$. 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.

Figure 2 Packet burst-based available-bandwidth measurement principle



We can write that the burst interval will be increased if

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

or

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

Note that $C - C_{\text{Cross}}$ is the available bandwidth (*A*) of the bottleneck link. Therefore, equation (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. Note that the bursty transmission in TCP appears when IC is enables, as mentioned in Section 2.3. ICIM-abw performing in reasonable measurement intervals do increase the burstiness but the increase is not effect much the performance of TCP, as shown in the simulation results in Section 3.3.

3.2 ICIM-abw

ICIM-abw inherits the concept of the *search range* from the measurement algorithm in ImTCP (Man et al., 2004). 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:

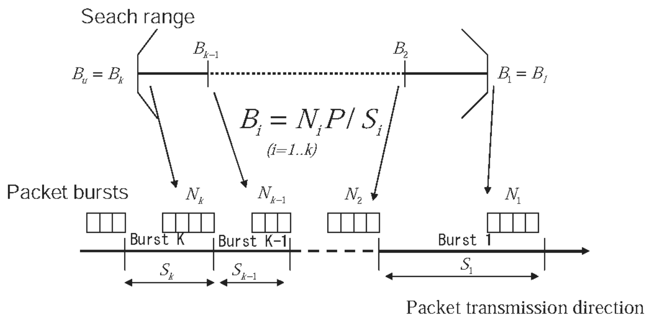
$$B_i = B_l + \frac{B_u - B_l}{k-1}(i-1) \quad (i = 1, \ldots, k).$$

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:

$$\frac{N \cdot P}{S_i} = B_i.$$

We illustrate the setting in Figure 3.

Figure 3 Probing a search range in ICIM-abw



Realisation of equation (2) requires the following:

- The value of S_i is required at the timing of the transmission of Burst i.

  Infact, 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 equation (2). We therefore estimate the value of S_i by assuming that the amount of data in Burst i is proportional to the interval as follows:

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

  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-abw utilises 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-abw stores all of the packets of the burst that preceded Burst 1 in the buffer. Packets are added to Burst i ($i = 1 \ldots k$) when necessary in order to maintain the desired number of packets (N_i) in these bursts.

  ICIM-abw sends k bursts and checks the corresponding ACK of the bursts. If from burst number j, $j = 1 \ldots k$, 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-abw first checks whether IC is enabled for the network card. For the reasons explained in Section 3.1, ICIM-abw checks the arrival intervals of the ACK packets. If more than 50% of the intervals are less than 1 µs, then ICIM-abw decides that IC is enabled. If the IC is enabled, then ICIM-abw continues the following measurement steps. Otherwise, the measurement algorithm introduced in ImTCP is used.

The measurement algorithm of ICIM-abw 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-abw 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 follows:

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

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

  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).

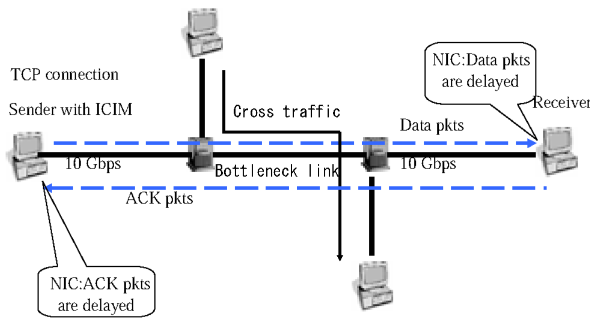
4 Wait for Q seconds then return to Step 2 and start the next measurement. During the waiting time Q, TCP transmits packets in the normal manner. The waiting time is needed for the TCP transmission to return to the normal state after the packets store-and-forward process at Step 2.

3.3 Simulation experiment

Measurement results

We show the measurement results for ICIM-abw through ns-2 (NS homepage) simulations. We implement ICIM-abw via Reno TCP, the most popular version of TCP, and use the topology shown in Figure 4 for the simulation.

Figure 4 Simulation topology



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 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 (NLANR Website), as shown in Table 1. The capacity of the bottleneck link is 5-Gbps, and the available bandwidth (A-bw) is 2-Gbps (from 0 s to 15 s), 3-Gbps (from 15 s to 35 s) and 4-Gbps (from 35 s to 50 s).

Figure 5(a) and (b) show the measurement results for ICIM-abw 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-abw 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 equation (3) becomes incorrect. Therefore, the probing rate of each Burst i may not be exactly equal to B_i (in Step 2 of Section 3.2). 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/s) becomes half of that when $Q = 1$ (34.2 results/s)

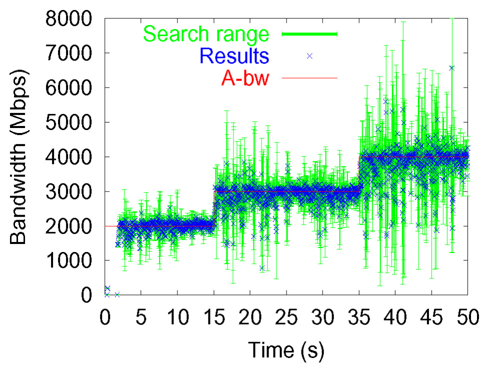
Table 1 Distribution of packet size of cross traffic

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

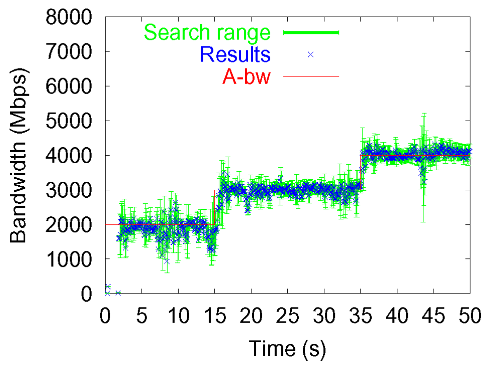
Comparison with IC-aware Pathload

We compare ICIM-abw with the only measurement tool we have found that can work in Gbps network. That is a version of Pathload that can detect and filter the effects of IC (Prasad et al., 2004). We call this version IC-aware Pathload. For the comparison between ICIM-abw and Pathload, the TCP sender and receivers are next replaced by the sender and receiver of Pathload. 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.

Figure 5 Measurement results for ICIM (a) measuring intervals $Q = 1$ RTT and (b) $Q = 2$ RTTs



(a)



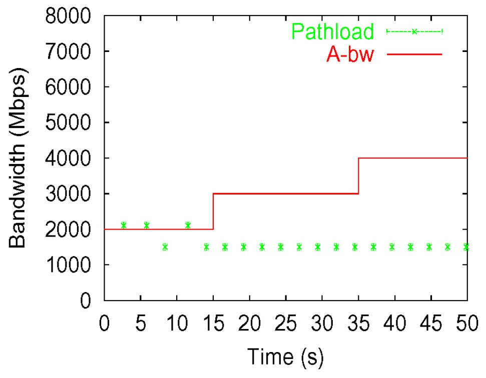
(b)

The measurement results of IC-aware 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.

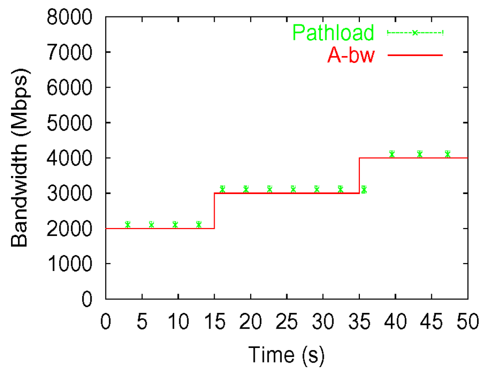
The measurement frequency is only 0.28 results/s, which is 60 times smaller than that of ICIM-abw (with $Q = 2$ RTTs).

Figure 6 Measurement results for IC-aware Pathload (a) number of packets in a stream $K = 160$ packets (b) $K = 200$ packets



(a)

Figure 6 Measurement results for IC-aware Pathload (a) number of packets in a stream $K = 160$ packets (b) $K = 200$ packets (continued)



(b)

Figure 6(b) shows that, if the A-bw changes during a measurement, Pathload may not detect the change well. At 15 s, 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 s, the probing rate of the ongoing measurement reaches 3-Gbps before the change in the A-bw from 3-Gbps 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.

Table 2 compares the number of packets used in the measurement of ICIM, and Pathload. ICIM-abw 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-abw is less than 1% of that of Pathload.

Table 2 Number of packets required for a measurement

A-bw	*ICIM-abw*	*IC-aware Pathload*
2-Gbps	110	$200 \times 12 \times 8 = 19{,}200$
3-Gbps	130	$200 \times 12 \times 9 = 21{,}600$
4-Gbps	154	$200 \times 12 \times 10 = 24{,}000$

Figures 5 and 6 show that the measurement results of ICIM-abw have a larger dispersion compared to Pathload because, based on the nature of the algorithm, ICIM-abw 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.

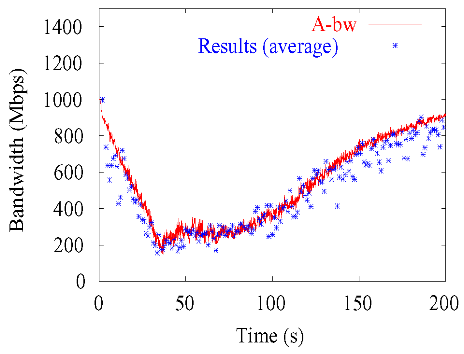
Measurement results in web traffic environment

We next investigate the measurement results for ICIM-abw 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 1-Gbps. The access links are also set to 1-Gbps.

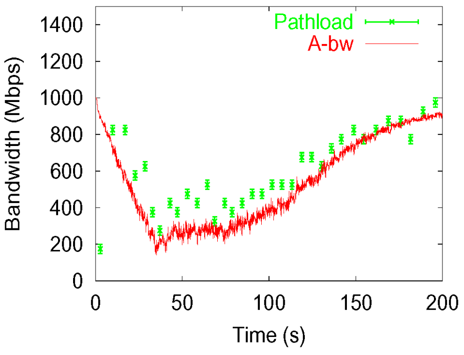
The available bandwidth is calculated as the capacity of the bottleneck link minuses the total amount of web traffic passing the link. Figure 7(a) shows the changes of available bandwidth and the average measurement results for each second. ICIM-abw underestimates the available bandwidth a little because 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-abw 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 100 Mbps 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 overestimates in most of the time. This frequent overestimation of bandwidth may lead to more aggressive systems. A conservative system caused by frequent underestimation of ICIM-abw will give less effect to the others sharing the same network environment.

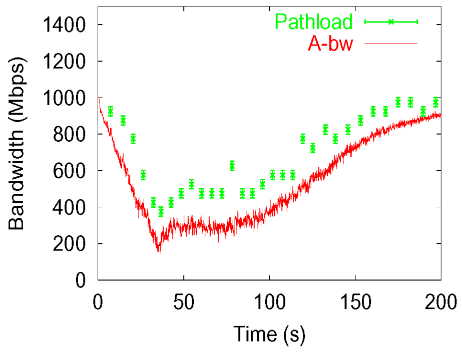
Figure 7 Measurement results in Web traffic environment (a) average measurement results of ICIM-abw for each second; (b) measurement results for IC-aware Pathload. $K = 160$ and (c) measurement results for normal Pathload. $K = 200$



(a)



(b)



(c)

TCP compatibility

We finally examine the data transmission performance of Reno TCP when it employs ICIM-abw. We perform a simulation where a number of Reno TCP connections that have ICIM-abw conflict with the same number of Reno TCP connections that do not have ICIM-abw 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 ten times, and the throughputs of the TCP connections that have and do not have ICIM-abw (and the ratio of thereof) are calculated and compared.

Table 3 shows the results when Q of ICIM-abw is set to 1 RTT and 2 RTTs and when ICIM-abw does not perform. In case ICIM-abw performs measurement in every RTT, the TCP achieves lower throughput than TCP that does not perform ICIM-abw when conflicts occur because ICIM-abw has to delay several data packets for measurement in this case. As shown in Table 3, the ratio of throughput between TCP with ICIM-abw 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-abw takes a lower measurement frequency, for example, when $Q = 2$ RTTs, then the TCP connections performing ICIM-abw can obtain the same throughput as normal Reno TCP, as shown in the third column of the table. We also disable ICIM-bw in all the TCP connections and show the throughput in this case in the fourth column of the table. We can see that that total throughput of TCP connection without ICIM-bw is almost the same of that when ICIM-abw with $Q = 2$RTT is enabled. This means that ICIM-abw with reasonable measurement intervals does not effect the TCP connection performance.

Figure 8 Simulation topology for examining TCP compatibility

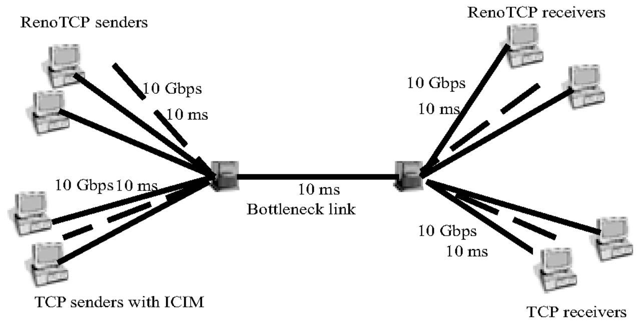


Table 3 Throughput (MBPs) of reno TCP using ICIM: normal Reno TCP (ratio)

#con.	*Q = 1 RTT*	*Q = 2 RTTs*	*No measurement*
4	466.4 : 490.6 (0.95 : 1)	483.7 : 475.6 (1.01 : 1)	489.5 : 478.9 (1.02 : 1)
8	451.1 : 544.4 (0.82 : 1)	505.1 : 490.5 (1.02 : 1)	510.7 : 485.9 (1.05 : 1)
12	418.7 : 577.7 (0.72 : 1)	503.5 : 493.2 (1.02 : 1)	503.1 : 493.3 (1.02 : 1)

4 ICIM-cap: Interrupt Coalescence-aware Inline Measurement for capacity

In this section, we focus on measuring the bandwidth-related metric: the end-to-end capacity of a network path. Together with the available bandwidth, capacity information is important for adaptive control in a transport protocol.

4.1 Existing capacity measurement technique and their problems

Many measurement techniques have been proposed for capacity measurement, such as Bprobe (Carter and Crovella, 1996), Pathrate (Dovrolis et al., 2004), CapProbe (Kapoor et al., 2004). All of these techniques utilise packet pairs for measurement. However, because packets transmitting back-to-back are always grouped at the NIC under the effect of IC, the receiver cannot read the correct inter-arrival intervals of the packets. Thus, the packet pair-based techniques fail to perform the measurement when IC is enabled.

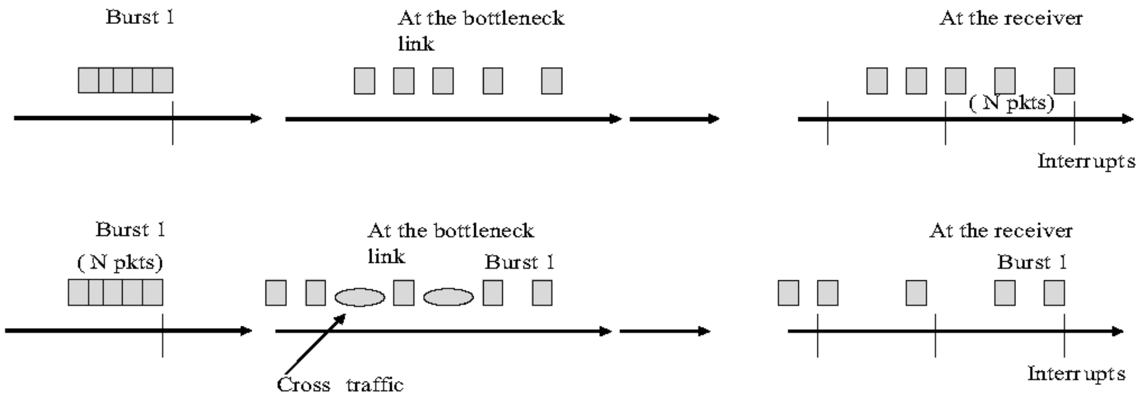
The first algorithm that can work in an IC environment is an enhanced version of Pathrate, suggested by Prasad et al. (2004). The work of the algorithm is as follows.

The sender sends a measurement train (a group of packets) that is long enough that at least two bursts are observed in the received train. Then the number of the packets in the first burst (*N*) is used for the calculation of capacity:

$$C = \frac{N_i \cdot P}{L}.$$

where *L* is the inter-arrival interval of the first and second burst (see the upper part of Figure 9).

Figure 9 Enhanced pathrate algorithm



However, the approach can work only when there is no cross traffic in the network path. If cross traffic exists, the cross traffic may cut into the measurement burst so that the number of packets received in the first interrupt may not reflect the value of the capacity correctly (See the lower part of Figure 9). This means the approach will not work well when the traffic load on the network is high.

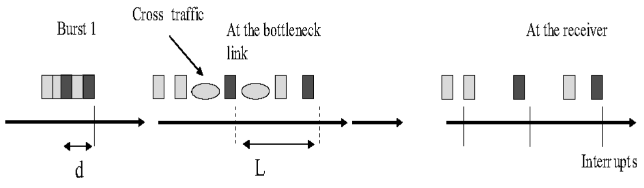
4.2 ICIM-cap

We propose a burst-based capacity measurement algorithm that can overcome the problems mentioned above. The main concept of the proposed algorithm is that the available bandwidth information, which can be yielded periodically due to ICIM-abw (introduced in the last section), is exploited. The available bandwidth information is used to estimate the quantity of cross traffic that cuts into the first burst. In Figure 10, the top packets of the bursts are drawn in black. The sending interval of the two packets is *d*. The amount of traffic that arrives at the bottleneck link between the arrivals of the two packets is as follows:

$$C \cdot L = N \cdot P + d \cdot C_{\text{Cross}}$$

where C_{Cross} is the average arrival rate of the cross traffic at the bottleneck link. Existing capacity measurement algorithms do not know the amount of C_{Cross}, so they cannot find the exact value of *C* in the case shown in the lower part of Figure 9. To the contrary, ICIM-cap can know the C_{Cross}, so it can perform well in such a case. That is the biggest feature of this algorithm.

Figure 10 ICIM-cap algorithm



If we suppose that the bottleneck link, which has the smallest capacity, is identical to the tight link, which has the smallest available bandwidth, we can write:

$$C_{\text{Cross}} + A = C.$$

We can then calculate the capacity as follows:

$$C = \frac{N \cdot P - d \cdot A}{L - d}. \tag{4}$$

4.3 Applying ICIM-cap measurement algorithm into TCP

The TCP sender sends the bursts for ICIM-abw alternately with the bursts for ICIM-cap. The newest result of ICIM-abw is used for the next measurement of ICIM-cap.

The length of the burst of ICIM-cap must be decided properly. It must be long enough that it can arrive in two interrupts. However, if it is too long, the TCP transmission will be adversely affected. We, therefore, propose a dynamic setting for the length, as follows:

- Quickly determine the initiative value.

  During the starting phase of the TCP connection, the TCP sender observes the length of the burst of packets and records the length of the longest one (*G*). The longest value is near the maximum number of packets that can be grouped in the same interrupt. So, we set the initial length of the measurement burst *L* to $1.5 \cdot G$ in order to be sure that the burst can be divided into two small bursts at the receiver.

- Adapt the length dynamically to the changes of the environment.

  If *L* is too short, which can be noticed when the measurement burst is not divided into multiple bursts at the receiver, then the sender doubles the length. If *L* is too long, which can be noticed when the measurement burst is divided into more than two bursts, the sender sets the length to $1.5 \cdot B$, where *B* is the number of packets passed to the receiver in the first burst.

4.4 Simulation experiment

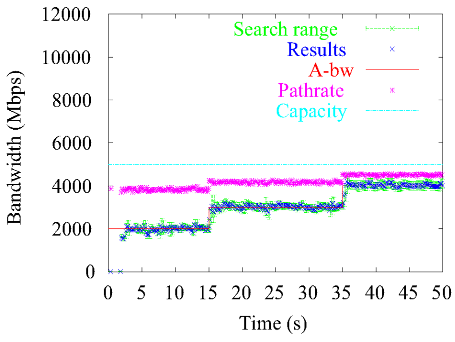
Through simulation validations, we show that ICIM-cap can deliver capacity measurement results quickly and correctly. Especially, it can deliver good results in extremely high-load networks, where current measurement algorithms such as (enhanced) Pathrate do not work well.

We repeat the simulation using the topology shown in Figure 4. This time, the TCP sender performs both ICIM-abw and ICIM-cap. For comparison, we also show the measurement results when we replace the ICIM-cap measurement algorithm by enhanced Pathrate. The measurement results are shown in Figures 11(a) and (b). In these figures, the measurement results of ICIM-abw are almost the same as those in Figure 5. Figure 11(a) shows the measurement results of enhanced Pathrate. As we can see, when the traffic load on the bottleneck link is heavy (when the available bandwidth is 2-Gbps while the capacity is 5-Gbps), Pathrate underestimates the capacity. On the other hand, as shown in Figure 11(b), ICIM-cap can deliver good measurement results regardless of the load on the network.

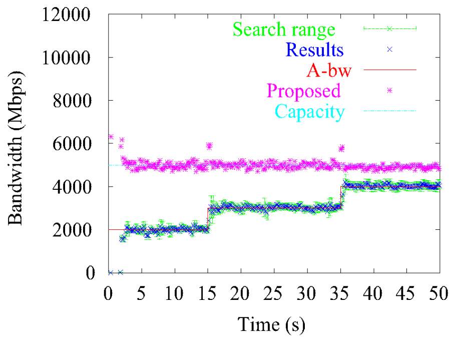
4.5 Discussion

ICIM-cap relies on the supposition that the bottleneck link and tight link are identical. In this session, we discuss the errors in the measurement results of ICIM-cap when the above supposition is incorrect.

Figure 11 Measurement results for IC-aware Pathload and ICIM-cap (a) measurement results for enhanced Pathload and (b) measurement results for ICIM-cap



(a)

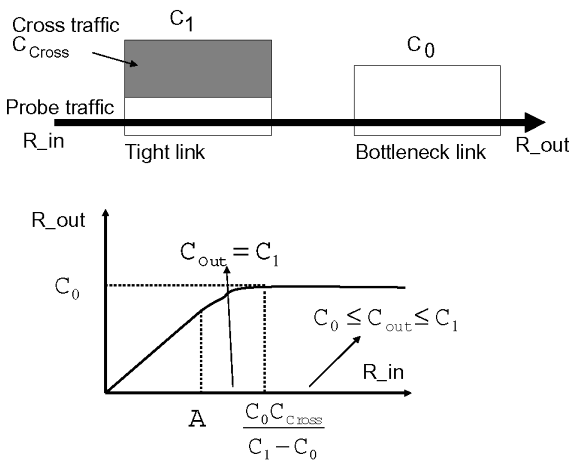


(b)

The case when the tight link is the upper link of the bottleneck link

Figure 12 shows the case when the tight link is the upper link of the bottleneck link. In this case, we suppose that traffic on another link does not affect much of the probe traffic. Moreover, the effect from the cross traffic on the bottleneck link is small. When the supposed condition is not true, the curve showing the relation between R_{in}, R_{out} will be more complex, but the tendency in general is unchanged.

Figure 12 The case when the tight link is the upper link of the bottleneck link



R_{in}, R_{out} are the sending rate and arrival rate, respectively, of the probe traffic. From equation (4), we can write:

$$C = \frac{NP/L}{NP/d}\left(C_{\mathrm{Cross}} + \frac{NP}{d}\right).$$

Because $NP/L = R_{\mathrm{in}}$, $NP/d = R_{\mathrm{out}}$, we can rewrite this using R_{in} and R_{out} as follows:

$$C = \frac{R_{in}}{R_{\mathrm{out}}}(C_{\mathrm{Cross}} + R_{\mathrm{out}}).$$

We call C_{out} the result of the above calculation. C_{out} is the measurement result given by ICIM-cap. In this case, we examine the relation between C_{out} and the real capacity value (C_0) as well as the capacity of the tight link (C_1).

Figure 12 shows the changes of R_{out} when R_{out} increases. When R_{in} is smaller than the available bandwidth (A), R_{out} increases in proportion to R_{in}. When R_{in} reaches A (but is still smaller than $(C_0 \cdot C_{\mathrm{Cross}})/(C_1 - C_0)$, the probe traffic starts to conflict with the cross traffic, and the increasing trend of R_{out} becomes slower. When R_{in} becomes larger than $(C_0 \cdot C_{\mathrm{Cross}})/(C_1 - C_0)$, R_{out} does not change regardless of the value of R_{in}. That happens because R_{out} is limited by the bottleneck link C_0 in this case. We summarise the results as follows:

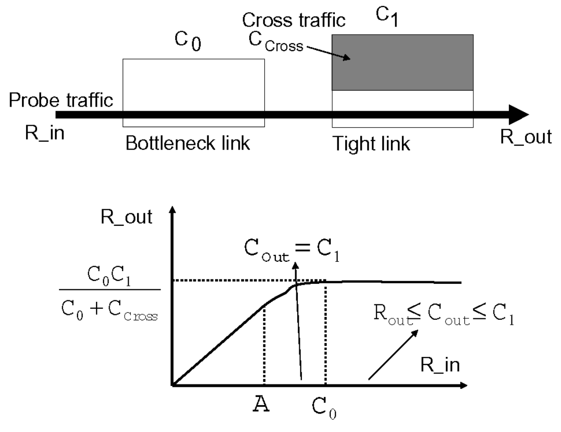
- When $A < R_{\mathrm{in}} < (C_0 \cdot C_{\mathrm{Cross}})/(C_1 - C_0)$, the measurement result of ICIM-cap C_{out} will be: $C_0 \le C_{\mathrm{out}} \le C_1$
- When $R_{\mathrm{in}} \le (C_0 \cdot C_{\mathrm{Cross}})/(C_1 - C_0)$, we also have $C_0 \le C_{\mathrm{out}} \le C_1$.

The case when the bottleneck link is the upper link of the tight link

If the tight link is the upper link of the bottleneck link, with the same observations as the above case, we can go to the following results (Figure 13).

- When $A < R_{\mathrm{in}} < C_0$, the result of ICIM-cap C_{out} is equal to C_1 ($C_{\mathrm{out}} = C_1$)
- When $C_0 \le R_{\mathrm{in}}$, the result of ICIM-cap C_{out} is included in the range: $R_{\mathrm{out}} \le C_{\mathrm{out}} \le C_1$.

Figure 13 The case when the bottleneck link is the upper link of the tight link



4.6 Interpretation of the results

When a tight link and bottleneck link are not identical, ICIM-cap's measurement result can overestimate, but the measurement never gets higher than the capacity of the tight link. In fact, a link with a large capacity does not often become a tight link, so the overestimation will not be very large. The measurement results can be an underestimation, however, the result is never smaller than R_{out}, which is the measurement result of enhanced Pathrate. Thus, the measurement results for ICIM-cap may not be correct when the supposition of the tight link and bottleneck link is not true, but the error is not large.

5 Experiment in a real environment

In this section, we present the experiment result in a real environment to validate the burst-based measurement algorithm. We implement the basic algorithm, ICIM-abw, in a FreeBSD system and use the simple network shown in Figure 14 to examine whether the algorithm works well. This network consists of two switches equipped with 1-Gbps Ethernet ports; all links are 1-Gbps. Table 4 shows the specifications of the PCs. The cross traffic is made up of UDP traffic sent by Iperf. One TCP connection is established between the sender and the receiver. In the TCP sender program, the ICIM-abw program is implemented. In this case, the link connecting the two switches becomes the bottleneck link. We control the Iperf flows so that the available bandwidth on the bottleneck link is 600 Mbps from 0 s to 50 s, 300 Mbps from 50 s to 100 s and 500 Mbps from 100 s to 150 s. Both NICs of the sender and the receiver enable IC; the *RxAbsIntDelay* parameter of IC is set to 0.1312 ms.

Figure 14 Network topology

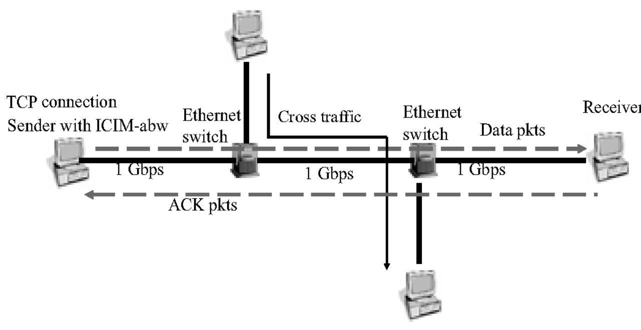
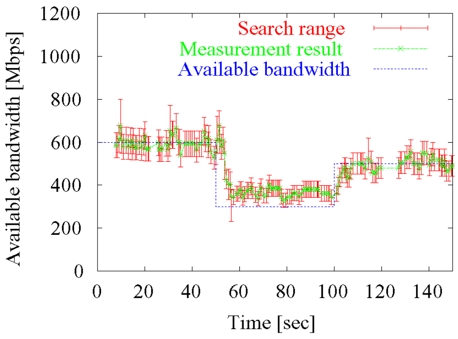


Table 4 Specifications of the PCs in the experiment

	Sender	*Receiver*
CPU	Intel P4 3.0 GHz	Intel P4 3.4 GHz
Mem.	1,024 MB	1,024 MB
OS	Free BSD 4.10	FedoraCore 4
NIC	Int. PRO/1000 Adapter	Int. PRO/1000 Adapter

In Figure 15, we plot the correct values of the available bandwidth. The measurement results for ICIM-abw and the search ranges are shown in the same figure. We can see that ICIM-abw can suitably measure the available bandwidth in this experimental network. Moreover, the measurement accuracy is as high as the evaluation of the simulation experiments in Section 3. As future studies, we will perform an experiment on ICIM algorithms in a large-scale network as well as on the internet.

Figure 15 Changes of the available bandwidth and the measurement results



6 Conclusion and future studies

In the present paper, we introduced ICIM-abw and ICIM-cap the methods that can measure the available bandwidth and capacity on a 1-Gbps or higher network path. The proposed measurement algorithms do not require regulation of packet transmission intervals and work well with IC. The simulation experiments showed that the proposed measurement algorithm works well in networks as high or higher than 1-Gbps.

At present, we are evaluating the performance of ICIM in a real internet environment. We are also testing the performance of the congestion control mechanism proposed by Hasegawa and Murata (2006) when ICIM is used for bandwidth estimation, in the real network environments.

References

Carter, R. and Crovella, M. (1996) *Measuring Bottleneck Link Speed in Packet-Switched Networks*, Technical Report, TR-96-006, March, Computer Science Department, Boston University, Boston.

Dovrolis, C., Ramanathan, P. and Moore, D. (2004) ‘Packet dispersion techniques and a capacity-estimation methodology’, *IEEE/ACM Transactions on Networking*, Vol. 12, No. 6, pp.963–977.

Floyd, S. (2003) *Highspeed TCP for Large Congestion Windows*, RFC 3649, December, Available at ftp://ftp.rfc-editor.org/in-notes/rfc3649.txt

Hasegawa, G. and Murata, M. (2006) ‘TCP symbiosis: congestion control mechanisms of TCP based on Lotka-Volterra competition model’, *Proceedings of Workshop on Interdisciplinary Systems Approach in Performance Evaluation and Design of Computer and Communications Systems (Inter-Perf. 2006)*, October, Pisa, Italy.

Hu, N. and Steenkiste, P. (2003) ‘Evaluation and characterization of available bandwidth probing techniques’, *IEEE Journal on Selected Areas in Communications*, Vol. 21, No. 6, August, pp.879–894.

Intel (2003) *Interrupt moderation using Intel Gigabit Ethernet Controllers*, Available at http://www.intel.com/design/network/applnots/ap450.pdf

Jain, M. and Dovrolis, C. (2002) ‘End-to-end available bandwidth: measurement methodology, dynamics, and relation with TCP throughput’, *Proceedings of ACM SIGCOMM 2002*, August, Pittsburgh, PA, USA.

Jin, C., Wei, D. and Low, S. (2004) ‘FAST TCP: motivation, architecture, algorithms, performance’, *Proceedings of INFOCOM 2004*, March, HongKong.

Jin, G. and Tierney, B. (2003) ‘System capability effect on algorithms for network bandwidth measurement’, *Proceedings of Internet Measurement Conference 2003*, October, Florida, USA.

Kapoor, R., Chen, L., Lao, L., Gerla, M. and Sanadidi, M. (2004) ‘CapProbe: a simple and accurate capacity estimation technique’, *Proceedings of ACM SIGCOMM 2004*, August, Philadelphia, USA.

Kelly, T. and Ott, T. (2002) ‘Performance sensitivity and fairness of ECN-aware modified TCP’, *Proceedings of Second International IFIP-TC6 Networking Conference*, May, Pisa, Italy.

Man, C., Hasegawa, G. and Murata, M. (2004) ‘Available bandwidth measurement via TCP connection’, *Proceedings of the 2nd Workshop on End-to-End Monitoring Techniques and Services E2EMON*, October, San Diego, CA, USA.

Marcondes, C., Persson, A., Sanadidi, M., Gerla, M., Shimonishi, H., Hama, T. and Murase, T. (2006) ‘Inline path characteristic estimation to improve TCP performance in high bandwidth-delay networks’, *Proceedings of the International Workshop on Protocols for Fast Long-Distance Networks*, February, Nara, Japan.

Prasad, R., Jain, M. and Dovrolis, C. (2004) ‘Effects of interrupt coalescence on network measurements’, *Proceedings of the 5th Passive and Active Measurement Workshop (PAM 2004)*, April, Antibes Juan-les-Pins, France.

Ribeiro, V., Riedi, R., Baraniuk, R., Navratil, J. and Cottrell, L. (2003) ‘PathChirp: efficient available bandwidth estimation for network paths’, *Proceedings of the 4th Passive and Active Measurement Workshop (PAM 2003)*, April, San Diego, CA, USA.

Strauss, J., Katabi, D. and Kaashoek, F. (2003) ‘A measurement study of available bandwidth estimation tools’, *Proceedings of Internet Measurement Conference 2003*, October, Miami, Florida, USA.

Syskonnect (2003) *SK-NET GE Gigabit Ethernet Server Adapter*, Available at http://www.syskonnect.com/syskonnect/technology/SK-NET_GE.PDF

Tsugawa, T., Hasegawa, G. and Murata, M. (2006) ‘Background TCP data transfer with inline network measurement’, *IEICE Transactions on Communications*, Vol. E89-B, No. 8, August, pp.2152–2160.

Xu, L., Harfoush, K., and Rhee, I. (2004) ‘Binary increase congestion control for fast long-distance networks’, *Proceedings of INFOCOM 2004*, March, HongKong.

Zec, M., Mikuc, M. and Zagar, M. (2002) ‘Estimating the impact of interrupt coalescing delays on steady state TCP’, *Proceedings of the 10th SoftCOM Conference*, October, Dalmacija, Italy.

Websites

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

Iperf. available at http://dast.nlanr.net/Projects/Iperf/

NLANR Web site. available at http://moat.nlanr.net/Datacube/

NS homepage. available at http://www.isi.edu/nsnam/ns/

Appendix: List of acronyms

ImTCP	Inline measurement TCP
IC	Interrupt Coalescence
ICIM	Interrupt Coalescence-aware Inline Measurement
ICIM-abw	Interrupt Coalescence-aware Inline Measurement for available bandwidth
ICIM-cap	Interrupt Coalescence-aware Inline Measurement for capacity