PAPER *Special Section/Issue on New Technologies and their Applications of the Internet III*

Impact of Bottleneck Queue Size on TCP Protocols and Its Measurement

Masaki Hirabaru,† *non-member*

Summary

The queue size at a bottleneck would impact the performance of TCP protocols, especially when running a single TCP flow in networks with a large bandwidth-delay product. However, queue size has been not well considered in experiments. This paper shows how bottleneck queue size influences TCP protocols performance. Bursityness of advanced TCPs is examined. Ways of estimating queue size are introduced. Sending a UDP packet train until a loss is detected is a method to measure queuing delay to estimate queue size. Watching a loss in a TCP session to measure round trip time and calculate the queue size is also discussed. Results from experiments with a network emulator and a real network are reported. The results indicated that a layer-2 switch at a congestion point would be a major factor of decreasing TCP performance in a fast long distant path.

Key words:

TCP, Congestion Control, Performance Measurement.

1. Introduction

There have been many high-performance data transfer experiments for scientific applications performed on the high-speed research Internet under conditions where an end-to-end path has a large bandwidth-delay product and a single TCP stream dominates at a congestion point or bottleneck. However, they do not mention the affect of bottleneck queue size on TCP performance in this situation. One reason for this is that there are many measurement tools available for estimating bandwidth and capacity at a bottleneck, but no one has reported queue size.

It is important to know their performance with a practical range of bottleneck queue sizes because the dynamic behavior of TCP protocol implementations varies depending on bottleneck queue size. If we can determine the bottleneck queue size, we could repeat the same experiments and assess the result's reliability.

In this paper, the model illustrated in Figure 1 will be used, where C is a capacity that sender can send and R is an outgoing rate from the bottleneck. The L2/L3 Switch represents a bottleneck with a queue of size Q. RTT is a total round trip time from the sender to itself along with the path. In this paper, a packet is 1500-byte long and a byte is 8-bit long.

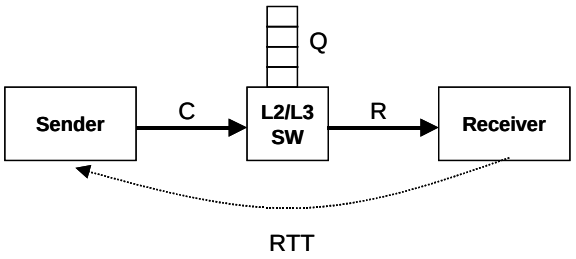

Fig. 1. Reduced model.

Note that in case of C > R, packets will be lost at the switch after filling up the queue. It happens in a packet switching network at traffic merging points or bottlenecks. Also, traditional loss-based TCPs intentionally raise those packet losses at the bottleneck to estimate the available bandwidth R. There can be another kind of packet losses called random losses derived from transmission errors but only the packet losses at a bottleneck due to congestion are examined in this paper. This paper focuses on a single TCP stream and do not mention another important TCP's feature of fairness among competing TCP flows.

Section 2 summarizes related works that measure queue size at a bottleneck. Section 3 explains how bottleneck queue sizes affect TCP performance. In Section 4, a way to measure bottleneck queue size is introduced and an active measurement tool is described. A passive way to measure TCP sessions is also discussed. Section 5 reports results from experiments with a network emulator and a research network. In Section 6, a typical problem of network configuration found during the measurements is described. Section 7 presents the conclusion and future work.

2. Related Works

There are few studies that measure queue size at a bottleneck in connection with TCP performance. Hedge et al. [1] measured and compared the performance of TCP protocols with different bottleneck queue sizes using a testbed. The effect of queue size is unknown in the real-network experiment.

Manuscript received March 12, 2005.
Manuscript revised June 10, 2005.
† The author is with NICT , Koganei, Tokyo 184-8795, Japan

Claypool et al. [2] reported a method to measure queue size in access networks under conditions of limited network topology, speed, and delays. There are difficulties in obtaining a delay value by pinging at the exact moment when a queue overflows in a high-speed backbone.

There are studies on round trip time (RTT) determined by analyzing packet traces. A way of extracting valid RTTs from traces is explained elsewhere [3]. It is unclear if maximum RTT can be used for calculating bottleneck queue size.

A maximum value of one-way delays should be measured immediately when a packet loss occurs because the loss indicates that the queue is overflowed. It also contributes to eliminate noises derived from queuing delays at a non-overflowing point. The method outlined by Lui and Covella [4] assesses loss pairs using a ns-2 simulator. It is important to develop a method and a tool to estimate bottleneck queue size in a high-speed network that has interference caused by delays.

3. Impact of Bottleneck Queue Size

Several TCP implementations were tested using a testbed, as shown in Figure 2 to ensure that bottleneck queue size affects throughput. All testbed equipment was tested prior to confirm operation at line speed. Three PC

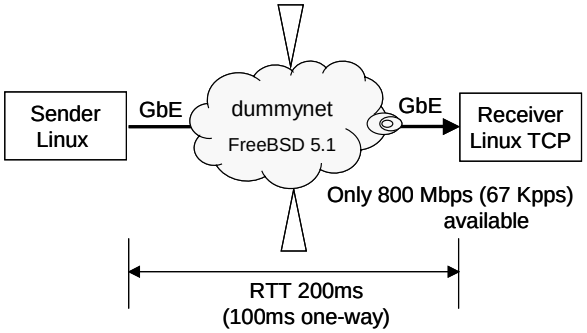

Fig. 2. Testbed configuration for measuring TCP performance.

servers with an Intel Xeon 2.4-3 GHz were used. The receiver was a regular Linux 2.4 Reno TCP, and only the sender operated with a modified version of the TCP. The emulator ran FreeBSD 5.1 with dummynet [15]. The dummynet was configured to have a 800-Mbps (67-Kpps) bottleneck and a 100-ms delay for both directions. IP MTU (packet size) is 1500 bytes. In this paper, a single TCP stream was treated.

3.1 TCP Average Throughput

Figure 3 shows the average throughputs of TCP protocols measured on the testbed for the first 5 minutes with the bottleneck queue sizes from 20 to 10000 packets.

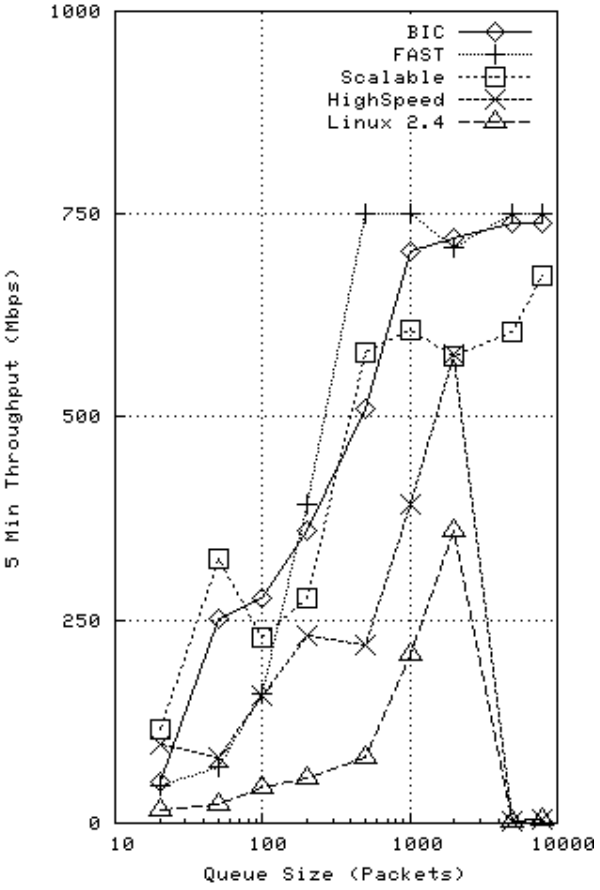

Fig. 3. TCP average throughput with different queue sizes.

With these queue sizes greater than 200 packets, Scalable [6], BIC [7], and FAST [8] TCPs indicated competing performance better than the other traditional TCPs tested.

The results clearly show that performance of these advanced TCPs is affected by bottleneck queue size. All of those TCPs did not work well with the small queue sizes less than 200 packets. Enough bottleneck queue size improves the performance of TCPs. It is important to know queue size in a path when an experimental result of TCP performance is described.

3.2 TCP Packets Queued at the Bottleneck

Because this paper investigates TCP performance affected by queue size, knowing the details behavior varying over time is important. By taking all the packet traces at the sender, it is possible to calculate the number of queued packets at the bottleneck by simulating drop-tail token buckets.

Figure 4 shows sending rates and the estimated number of packets queued on the dummynet every RTT intervals with a 1000-packet queue. The figures show that there are

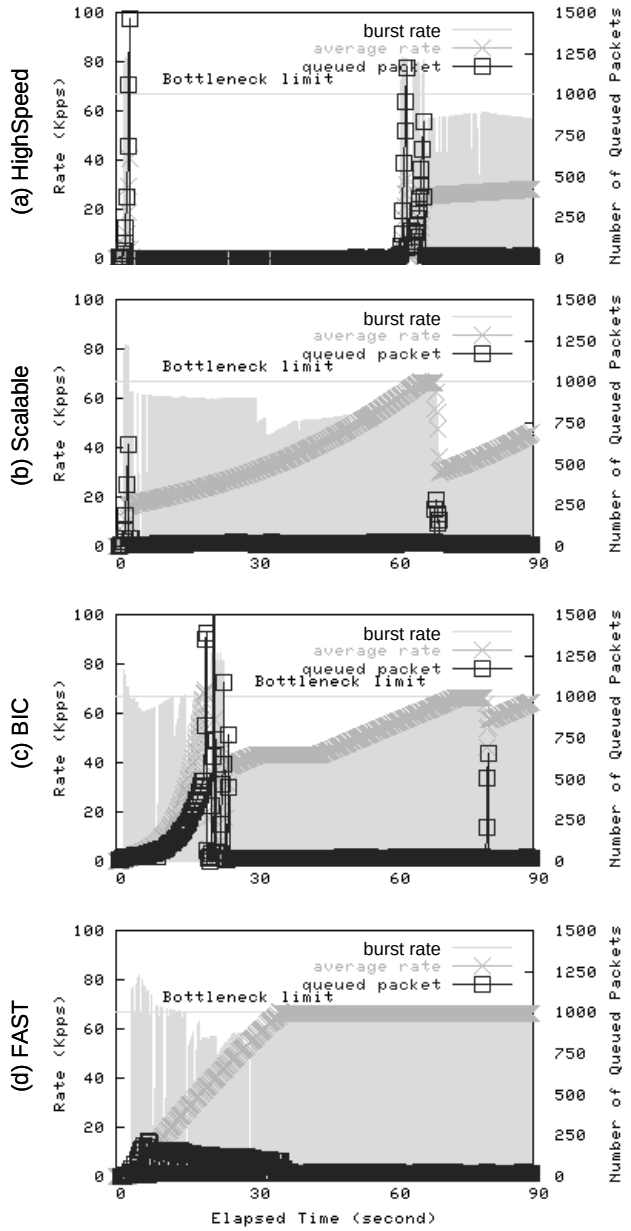

Fig. 4. TCP flow burstyness in the first 90 seconds.

bursts that overflow the bottleneck queue before finding an available bandwidth appropriately. The average rate is at what TCP intends to send, but there is no rate control during slow-start phase or finding the available bandwidth quickly. Thus, TCP sends out packets as a burst every RTT intervals at a rate up to the sender capacity (C in Figure 1).

The number of queued packets (N) at every RTT intervals in the slow-start phase is obtained by the following equations:

$$N = W \cdot RTT \cdot (R_i - R_o) / C \quad (1)$$

$$W = W_i \cdot 2^{(Time / RTT)} \quad (2)$$

where W is TCP's window size in packet, RTT is 200 ms, C (Capacity) is 1 Gbps, and R_o (Bottleneck rate) is 800 Mbps. W_i (Initial window size) is 4.

R_i is a burst rate at which the sender generates the traffic. With the equation (2), the traditional TCPs increment the windows size by two on receipt of every acknowledgement so that the R_i is limited to twice of the bottleneck rate. In this case, R_i is the same as C. This is TCP's nature of self-clocking similar to one we see in congestion avoidance phase. For example, a 1000-packet queue is expected to become full with a window size of 25K (150 Mbps average rate) if another queue and delay on a path is not considered. Traditional TCPs do not estimate bandwidth well with a short bottleneck queue.

In addition to the above TCP flow burstyness derived from its window-based rate control, there would be another reason that explains the queue overflow during the first phase. Because traditional TCP's growth of the window size almost jumps at every RTT intervals with the equation (2), the final increasement of window size may keep overflowing a bottleneck queue until a packet loss signal due to congestion reaches at the sender at most after RTT.

HighSpeed TCP in Figure 4 (a) indicates that the window incensement in the first few seconds is too aggressive, resulting in a lot of packet losses. Those lost packets are retransmitted in the following long recovery phase around for 60 seconds. Even with larger queue size (5000 in Figure 3), the situation becomes worse. With the larger queue size, TCP could estimate the available bandwidth better, but doubling window size ends up with more packets lost at the end of slow-start phase, as explained above, which followed by a longer recovery phase lasting more than 5 minutes. Note, during the congestion avoidance, burst rates are below the available bandwidth because of the self-clocking property of TCP. Linux 2.4 (Reno TCP), which is not included in Figure 4, shows a similar trend with HighSpeed TCP.

In Figure 4 (b) – (d), there are bursts in the first seconds, but it becomes close to average as the rate grows slowly. In Figure 4 (b), Scalable TCP estimates the available bandwidth at 40 Kpps (or 500 Mbps) restarting congestion avoidance at its half rate. Because its growth of windows size in the congestion avoidance phase is aggressive so that it reaches at the top in a minute unlike traditional ones including HighSpeed TCP. In Figure 4 (c), BIC TCP needs large queue size at the final stage of available bandwidth estimation around at 20 seconds although its estimation seems to be good enough in this case. In Figure 4 (d), FAST TCP does not require large queue as it is designed. It occupied the queue no more

than 250 packets with satisfactory and stable performance. In summery, Scalable, BIC and FAST TCPs take slower steps to find an available bandwidth. Slower incensement receives benefits from TCP's self-clocking property and minimizes the number of packets queued at the bottleneck.

4. Estimating Bottleneck Queue Size

This section describes the methods to estimate bottleneck queue size and the results of applying the methods to network switches and routers.

4.1 A Measurement Method for Queue Size

The method to measure queue size is straightforward. It is just to continue sending a packet train in UDP until the queue overflows and detect the losses on the receiving side, as shown in Figure 5. All the packets carry a sender's

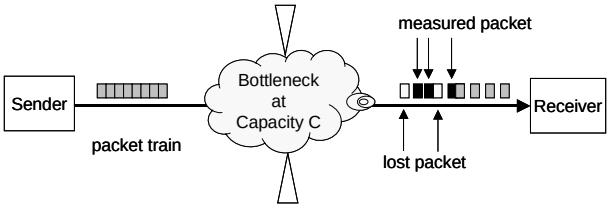

Fig. 5. Queue size measurement method.

timestamp so that the difference in the minimum and maximum one-way delays can be measured independent from the absolute clock errors on both ends. The calculated difference is the queuing delay so that an estimation of queue size in packet is the calculated difference times the capacity divided by MTU or packet size, as follows:

$$Queue\ Size = Capacity \cdot (Delay_{max} - Delay_{min}) \quad (3)$$

Although the receiver side can measure the capacity of the bottleneck if there is no cross traffic, it can be obtained from other measurement tools, like pathrate [10].

Note that a maximum queuing delay required here can be measured immediately before the loss. Taking this into account would remove noise from the data. This is the same technique used in loss pairs [4]. Apparently, the sender needs to send a packet train at a rate higher than the bottleneck.

There could be multiple samples where a loss occurs in a train so that currently the tool picks up a mode in the values. The first packet traveled is removed from measurement because it may be affected by delays in the ARP resolution and cache miss hit of a route look-up on a sender as well as intermediate routers.

The method can works two ways; 1) sending a UDP train at a constant rate up to seeing a packet loss, 2) sending a UDP train doubling its rate every RTT intervals. The former would take a longer test period because it expects that another traffic triggers overflow. The latter is certainly intrusive. However, stopping the train immediately after detecting a loss would minimize damage to cross traffic at the bottleneck point. It is like TCP's behavior to increase its sending rate until a packet loss occurs. In the both ways, outgoing rate is controlled well so that bursty traffic like TCP is not generated

A program called *pathq* was developed to perform this measurement with the testbed described in Section 3 and obtained enough accuracy to estimate bottleneck queue size. An active queue management method, like RED [5], may be used on a router in a path, but in this paper only a drop-tail FIFO queue is assumed.

4.2 Measuring a Path using TCP

The method described above requires that both ends run measurement programs. For situations where we do not have access to both ends, another passive method can be possible to measure delays while TCP runs. Monitoring TCP sessions on the sender's side enable possible detection of packet loss and measurement of RTT at the same time. Duplicate ACKs and SACK options are signs of packet loss. Duplicates ACKs for fast retransmission indicates a packet loss and a new SACK option field also indicates that there was a packet loss. Just taking a RTT value right before a packet loss may include a case of random packet loss so that it is important to see increase trend of RTTs.

RTTs kept in TCP stacks can be affected by another queue below the TCP stack, such as an interface queue or a queue in a device driver for DMA due to burstyness of TCP flows. The difference between CPU power of putting a packet into the queues and a line rate of the interface forms a bottleneck inside a host. Therefore, in my method, packets are captured on an interface using tcpdump. More precisely, packet traces are captured through an optical splitter on a different host to avoid running tcpdump on the sender host.

It was also confirmed that the delay from when a receiver receives a packet to when an acknowledgement signal is sent is about 400 us maximum. Note, the method described in Section 4 sends a packet at a controlled rate from the application, so it is not blocked at the bottleneck inside the host. TCP's delayed acknowledgment is enabled, but its delay is limited to one packet interval that is about 12 us at 1 Gbps with a 1500-byte continuous packet flow. Therefore, the delay is negligible compared to queuing delays. Caching TCP parameters, like ssthresh, in the kernel have been disabled to run the test independently from knowledge from previous TCP sessions. This makes sure that every time the TCP begins with slow-start and will end by seeing a packet loss.

4.3 Measuring Dummynet

It is important to determine the accuracy of the results. To do that, the methods were applied to dummynet that can be configured to provide an arbitrary delay and size of a queue. Iperf 1.7.0 [16] was used, and dummynet was configured to have a queue of 250 packets. Under these conditions, the expected queuing delay is 3750 us. The distribution of the measured queuing delays reported by the program is shown in Figure 6.

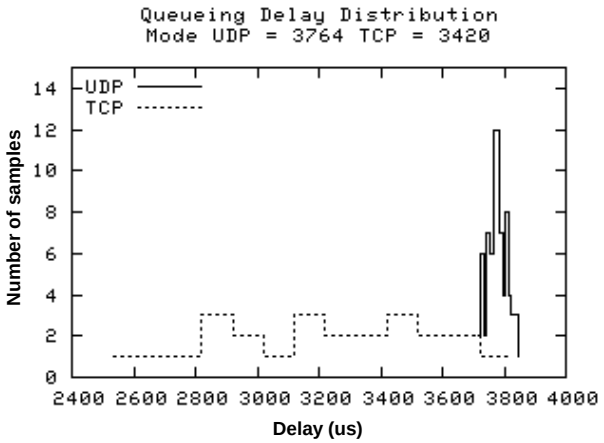

Fig. 6. Queuing delay distribution with queue size 250 packets.

The distribution results suggest that sending a UDP packet train is fairly accurate. Most values fell within a range of 100 us around the expected value. The errors may have resulted from an interrupt delay in the NIC. Although watching TCP sessions does not give us a sharp peak of exact queue size, it still gives us a rough estimate of queue size.

4.4 Measuring a Box

The next step is to measure switches and routers in order to know typical queue sizes using the way that a UDP packet train is sent. All the device's ports were Gigabit Ethernet, but in the case of Switch A to C the receiver's port was configured at 100 Mbps to instantly form a bottleneck inside the switch, shown in Figure 7(a). The other cases use three ports to inject cross traffic (200 Mbps) from one of them to the port where the receiver is connected, shown in Figure 7(b).

To make sure this method works, queuing delays was measured with a switch that has a 50-packet queue. The maximum queuing delay, 6000 us shown in Figure 8, indicates that the method measured accurately the delays to estimate the exact queue size. Note that the result of TCP exceeds by a few hundred of micro seconds caused

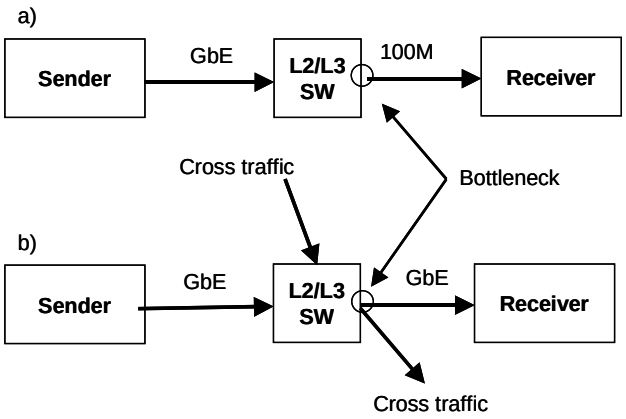

Fig. 7. Bottleneck formation with a switch.

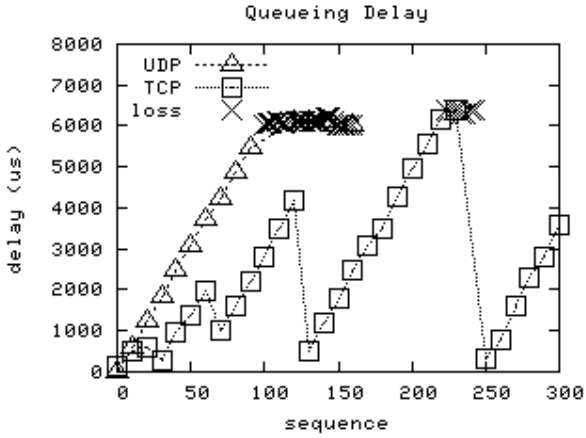

Fig. 8. Observed queuing delays in UDP and TCP.

by processing delay to return an ACK packet in TCP on the receiver host.

Table 1 lists the results measured in the laboratory. The UDP method was used. All calculated queue sizes are in packet of 1500 bytes. Even from those few samples, the

Table 1. Switch / Router queue size measurement result

Equipment	Max. Queuing Delay (us)	Capacity (Mbps)	Calculated Queue Size (1500B)
Switch A	6161	100	50
Switch B	22168	100	180
Switch C	20847	100	169
Switch D	738	1000	60
Switch E	3662	1000	298
Router F	148463	1000	12081
Router G	188627	1000	15350

so-called switches have much smaller queue sizes than routers. Note that the selection of those test devices are at random; the results state that there are a wide range of queue sizes depending on the performance and cost of the devices. Only the device placed on a possible congestion point need to be cared.

The queue sizes obtained here may not be buffer memory size in a device. The memory may be shared among several ports and may be installed on both input and output ports. The point is that TCP congestion control is affected as if a bottleneck equips a queue with these sizes measured from outside of the boxes.

5. Measuring a Research Network

This section shows an experiment that measures a 1000 km path from Tokyo to Fukuoka, in Japan, that contained three routers and more than five switches. There is another traffic flowing on the path so that congestion occurs between the user traffic and the test traffic. The results obtained are shown in Figure 9.

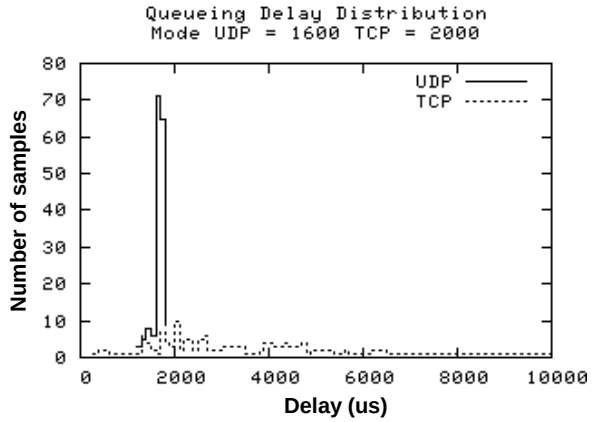

Fig. 9. Queuing delay distribution of path from Tokyo to Fukuoka.

Sending a UDP packet train gives a sharp peak at 1600 us, whereas watching the TCP connections implied that there would be a value around 2000 us. In each case, the queue sizes were calculated to be 133 and 166 packets and were too small to gain reasonable TCP performance as described in Section 3.

6. Typical Performance Problem

During the experiment an undesirable network configuration in terms of TCP performance was found. Similar to the configuration shown in Figure 8, a major congestion point existed on a switch that did not have a long enough queue for advanced TCPs according to the results in Section 3.

Figure 10 shows the configurations. The so-called switches are often used at a location of traffic exchange instead of installing interfaces on a router in order to reduce a cost. Common data sheets do not explain about queue size on an interface. Figure 10(a) indicates that if there is continuous commodity traffic from left to right, for example, at a rate of 200 Mbps, its condition for a second TCP flow is similar to one shown in Figure 2.

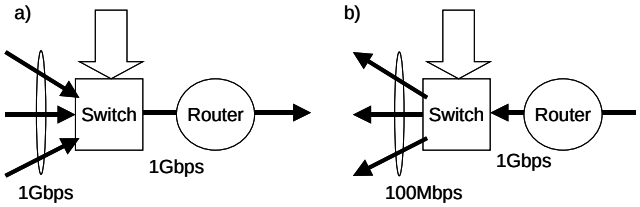

Fig. 10. Typical cases of congestion point.

Figure 10(b) situation is similar to Figure 10(a), but the reverse direction can be congested without any cross traffic. This is a typical case when network speeds are converted and often seen in a commodity network where a server is connected at a Gbps and a client is connected at 100 Mbps or slower speed. For example, clients on the 100 Mbps links downloading a file from the server on the 1 Gbps link may be affected by the problem described. Because burst traffic nature of TCP flows shown in Figure 4, a large 900 Mbps difference in speed forms a bottleneck harder than Figure 2. The number of queued packets increases as the difference in bandwidth becomes larger, as explained by equation (1).

Network designers need to be reminded how the queue size at a congestion point impacts TCP performance. An inadequate placement of a switch with a short queue on a backbone congestion point would decrease TCP performance on a fast long distant path. This kind of a layer-2 congestion point is difficult to be identified using an IP layer measurement tool like pathchar [11].

Note that Ethernet flow control defined in the IEEE standard 802.3x does not help this situation without introducing head-of-line blocking between the switches. The flow control stops all the traffic on a port, so using it to stop the congesting flow may result in stopping another independent flows on the same port that do not need to be stopped. Especially on the backbone, there are many flows running across a switch and a router, so it is limited to apply the port-based flow control without decreasing the total performance [17]. In the experiments, Ethernet flow control is disabled to keep the best performance of switches and routers.

7. Conclusions and Future Works

This paper demonstrates how a bottleneck queue size impacts TCP performance when transferring data long distances at a high rate in a single TCP flow. Knowing a bottleneck queue size helps in understanding TCP performance and is also important when evaluating the performance of TCP protocol implementations performed on the Internet. By examining dynamic behaviors of TCP implementations in respect of burstyness, it is shown that packet losses at a bottleneck would happen during quick rate growth of a TCP flow even when an average sending rate is less than the available bandwidth.

Sending a packet train until a packet loss occurs provided an accurate enough result for estimating bottleneck queue size in a simple and single bottleneck case. Watching TCP sessions can be used as a passive measurement method to provide approximate queue size number. Future work should focus on improving accuracy.

Results from research network experiments were reported. A network device with a short queue placed at a congestion point would be a major factor of decreasing TCP performance in a fast long distant path.

The tool *pathq* is expected to discover bottleneck queue size in seconds. However, running it for a long period, e.g. for a day and at different constant rates would produce statistical delay distribution that has multiple peaks in the samples of delays. That implies that bottleneck point changes over time and also there could be multiple bottlenecks. When measuring an unknown path the existence of active queue management (AQM) routers should be considered. Closely looking at the delay distribution obtained in a received packet train, it would be possible to infer RED setting. To improve accuracy, a method to eliminate noise should be considered in the way of watching losses in TCP connections. A test performed over a longer period would give me a delay distribution that implies changes of bottlenecks over time.

I am now running tests with measurement points over the research networks listed in the Internet2 piPEs PMP directory [12]. I am also using the BWCTL [13] tool in running test TCP sessions to allow me to collect packet traces to analyze delays. This effort is also used in e-VLBI [14] where a long distant high-performance data transfer is required.

Acknowledgments

I would like to thank my colleagues Dr. Katsushi Kobayashi and Noritoshi Demizu, NICT for their advices on TCP protocols. I also would like to thank PAM 2005 and IEICE reviewers for giving me valuable comments.

References

[1] Sanjay Hegde, David Lapsley, Bartek Wydrowski, Jan Lindheim, David Wei, Cheng Jin, Steven Low, and Harvey Newman, "FAST TCP in High Speed Networks: An Experimental Study," Proceeding of GridNets, Oct. 2004

[2] Mark Claypool, Robert Kinicki, Mingzhe Li, James Nichols, and Huahui Wu, "Inferring Queue Sizes in Access Networks by Active Measurement," PAM 2004.

[3] Jay Aikat, Jasleen Kaur, F. Donelson Smith, and Kevin Jeffay, "Variability in TCP Round-trip Times," ACM SIGCOMM Internet Measurement Conference '03, Oct. 2003.

[4] Jun Liu and Mark Crovella, "Using Loss Pairs to Discover Network Properties," ACM SIGCOMM Workshop on Internet Measurement, pp.127-138, 2001.

[5] Sally Floyd and Van Jacobson, "Random Early Detection Gateways for Congestion Avoidance," IEEE/ACM Transactions on Networking, 1993.

[6] Tom Kelly, "Scalable TCP: Improving Performance in Highspeed Wide Area Networks," ACM SIGCOMM Computer Communication Review 32(2), Apr. 2003.

[7] Lisong Xu, Khaled Harfoush, and Injong Rhee, "Binary Increase Congestion Control for Fast Long-Distance Networks," IEEE INFOCOM 2004.

[8] Cheng Jin, David X. Wei, and Steven H. Low, "FAST TCP: Motivation, Architecture, Algorithms, Performance," IEEE INFOCOM 2004.

[9] Sally Floyd, "HighSpeed TCP for Large Congestion Windows," RFC 3649, Dec. 2003.

[10] C. Dovrolis, P. Ramanathan, and D. Moore, "What do packet dispersion techniques measure?", IEEE INFOCOM, pp. 905-914, Apr. 2001.

[11] V. Jacobson, " pathchar -- A Tool to Infer Characteristics of Internet Paths," ftp://ee.lbl.gov/pathchar/, 1997.

[12] Performance Measurement Point Directory, Internet2, http://e2epi.internet2.edu/pipes/pmp/pmp-dir.html.

[13] Bandwidth Control (BWCTL), Internet2, http://e2epi.internet2.edu/bwctl/.

[14] Masaki Hirabaru, “Performance Measurement on Large Bandwidth-Delay Product Networks,” Proc. of 3rd e-VLBI Workshop, NICT TDC News, No.25, pp. 11-19, Nov. 2004.

[15] Rizzo, L., "Dummynet: a simple approach to the evaluation of network protocols", ACM Computer Communication Review, Vol.27, No.1, pp.31-41, Jan. 1997.

[16] Iperf, http://dast.nlanr.net/Projects/Iperf/.

[17] Molle, M. and Watson, G., “100Base-T / IEEE 802.12 / packet switching,” IEEE Communications Magazine, Vol.34, No.8, pp. 64-73, Aug .1996.

Masaki Hirabaru received the M.S. and D.Eng. degrees in computer science from Kyushu University in 1985 and 1989, respectively. He worked with Kyushu University, University of Tokyo and Nara Inst. of Sci. & Tech. During 1996-2000, he was with Merit Networks to study Internet routing, performance measurement and analysis. Since 2003, he has been a senior researcher of NICT (National Institute of Information and Communications Technology).

List of figures and tables: