Capacity Dimensioning Based on Traffic Measurement in the Internet

Kazumine Matoba † **Shingo Ata** ‡ **Masayuki Murata** §

†Network System Laboratories, Fujitsu Laboratories LTD.

4-1-1, Kamikodanaka, Nakaharaku, Kawasaki, Kanagawa 211-8588

Phone: +81–44–754–2629, Fax: +81–44–754–2786

E-mail: k-matoba@jp.fujitsu.com

‡Faculty of Engineering, Osaka City University

3–3–138 Sugimoto, Sumiyoshi-ku, Osaka 558–8585

Phone, Fax: +81–6–6605–2191

E-mail: ata@info.eng.osaka-cu.ac.jp

§Cybermedia Center, Osaka University

1–30, Machikaneyama, Toyonaka, Osaka 560-0043

Phone: +81-6-6879-8793, Fax: +81-6-6879-8794

E-mail: murata@cmc.osaka-u.ac.jp

Abstract: Network dimensioning is becoming important in order to provide a stable Quality of Service (QoS) to the customers. However, it is not an easy task because the Internet traffic changes dynamically and frequently. We thus need a new approach for network dimensioning suitable to the Internet. Probably, it would be quite different from the existing well-established method for the telephone network. Actually, our method proposed in this thesis tries to establish a new paradigm of network dimensioning for the new communication era of the Internet.

In this paper, we consider the network dimensioning approach in order to resolve the bottleneck for stable and QoS-rich data communications in the Internet. For that purpose, a possible cause of the bottleneck must be examined not only within the network, but also at the end hosts, which originated in an intrinsic nature of the Internet architecture. In our study, we first propose a method to identify the bottleneck for improving the performance of the end users. Once the network is found to be bottleneck, the network operator should determine how much the network resources is increased to meet the end user's QoS requirements. So, we next focused on the measurement method for utilization of the bottleneck link between end-to-end hosts. To measure the utilization of the bottleneck link accurately, we propose a new measurement method that can eliminate several kinds of measurement errors. We collected the measurement results from the actual operating Internet, and provide an evidence to support an appropriateness of our approach.

Based on the measurements, we finally propose a design framework in order to determine the adequate link capacity for offering a desired QoS level to the end users.

All correspondence should be directed to:

Dr. Shingo Ata

Department of Information and Communication Engineering

Graduate School of Engineering, Osaka City University

3–3–138 Sugimoto, Sumiyoshi-ku, Osaka 558–8585, Japan

E-mail: ata@info.eng.osaka-cu.ac.jp

Phone, Fax: +81–6–6605–2191

1 Introduction

Network dimensioning for the Internet is becoming more important because Internet Service Providers (ISPs) will not be commercially successful unless they provide their customers with a stable Quality of Service (QoS). Network dimensioning, however, is not easy because the network is changing rapidly and frequently.

Several network architectures providing QoS-rich communications have been developed recently. One example is the found in a diff-serv architecture [1], where various bandwidths can be provided for various QoS classes. Another example is the MPLS architecture [2] and IP-over-WDM networks, where the underlying network (an ATM, SONET or WDM network) provides the physical paths needed to connect IP routers in an end-to-end fashion and the bandwidth between edge routers is guaranteed.

There is, however, a fundamental limit in those architectures: the physical (and/or logical) path capacity should be determined a priori by network providers. Network dimensioning plays an essential role in determining the appropriate network capacity. The network providers should therefore first determine QoS metrics for end users, characterize the traffic generated by computer applications, and then estimate the amount of network resources needed to satisfy those QoS metrics. In computer networks such as the Internet, those are very complicated, and their interdependence makes network dimensioning even more difficult. Consider, for example, Web applications. Even if the bandwidth provided by the network is insufficient, an application will still work because the underlying TCP can adjust the packet emission according to the network congestion status. That is, the traffic characteristics by the end hosts are apparently influenced. Furthermore, whether the resultant QoS (the document transfer time in the case of Web browsing) will or will not satisfy the users is not clear. Anyway, a problem is that we have no established QoS metric which satisfies the Web users.

Network dimensioning for a loosely coupled network like the Internet probably needs to be approached in a way different from that used to approach network dimensioning in the conventional telephone network, where the network carrier is responsible for maintaining the network. What is critical here is that QoS metrics such as Web document download times can never be measured by the network providers, but only by the end users. This is because in the Internet the processing of protocols higher than those of layer-four (i.e., the transport layer) is performed at the end host. We think that the first step in establishing a framework for network dimensioning in the Internet is to measure the network characteristics. The results of these measurements can then be used for dimensioning the network resources for the users.

The first thing that needs to be done is to identify the existing bottleneck in meeting the QoS requirements of the users. It can be in the network resources (available bandwidth, a router's packet-forwarding capability, etc.) or at the end hosts in the Internet. Increasing the network bandwidth does not necessarily improve the users' QoS and may be a wasted investment if the bottleneck is at the end hosts. Because the major applications in the current Internet are Web systems utilizing the http and TCP, we propose a bottleneck identification method for TCP-based applications. Possible causes of the bottleneck in these applications are the socket buffer size of the receiver, the available bandwidth of a link, and the packet-processing at the sender. As will be described in Section 3, we can find the cause of the performance bottleneck by carefully examining the characteristics of the TCP connections.

It is also very important to measure the bottleneck link utilization between end hosts. By characterizing the bottleneck link, we can know the QoS level that the end users receive in the current network environment. To measure the utilization of the bottleneck link, we developed a new measurement tool based on the `cprobe`.

If we find the bottleneck to be a link within the network and we estimate its current utilization, we then need to determine how much the bottleneck link capacity needs to be increased in order to meet the QoS requirement of users. We use the TCP throughput as a QoS metric and propose the design framework for determining the link capacity. In determining the link capacity, it is important to consider the characteristics of cross traffic sharing the congested link with the connection of the target user. Taking account of the cross traffic, in Section 4 we describe the method for determining the new link capacity.

The rest of this paper is structured as follows. We found bottlenecks by measuring network characteristics, including round-trip time and packet loss probability, and in Section 2 we propose a bottleneck identification method based on these characteristics. In Section 3 we explain how we evaluated the current status of the bottleneck link by measuring its utilization. Using the measurements described in the previous sections, in Section 4 we explain how our capacity dimensioning method works well. In Section 5 we conclude by briefly summarizing this paper and indicating a direction for future research.

2 Measurement-Based Identification of Performance Bottlenecks

For network dimensioning, we need first determine which factor actually restricts the current performance. If the link capacity is found to be a factor limiting performance, it needs to be increased. In this section we propose a new bottleneck identification method based on the measurement of network characteristics.

2.1 Framework for Bottleneck Identification

The framework of our measurement-based bottleneck identification method is shown in Figure 1. Considering the TCP connection through data (e.g., Web documents) is sent from the destination host (server) to the measurement host (client), we identify potential bottlenecks in the current network configuration and determine which one limits the end-to-end performance. Our identification process is applied to each TCP connection.

We classify the factors that can limit the TCP performance into the following three categories.

1. **Network Conditions**

   The capacity of one or more links is quite a bit smaller than the end users expect. A simple example may be found in the case where the customer is connected by a telephone line. Then the throughput is limited by the speed of a modem (e.g., to 33.6 or 56 Kbps). The most important factor limiting performance, however, is link congestion. It leads to the loss of many packets and reduces TCP throughput significantly. It can be resolved only by increasing the capacity of the bottleneck link.

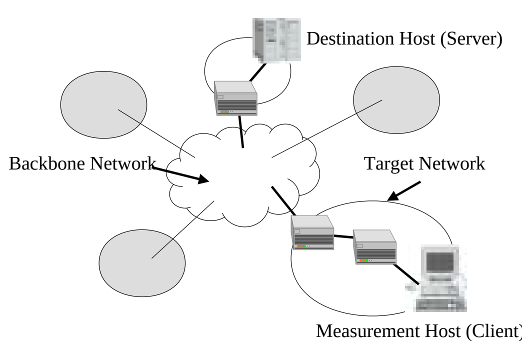

Figure 1: Measurement framework.

2. **Receiver–Side Configurations:**

   To keep the packet sequence from becoming mis-ordered, the receiver host prepares a buffer to store the received packets. The required buffer size in a TCP connection is basically given by the *Bandwidth–Delay Product* (Available Bandwidth $\times$ RTT). The sender is notified of the buffer size of the receiver host by the *Advertised Window* field in the TCP header. This buffer sometimes becomes a bottleneck in an environment with a large *Bandwidth–Delay Product* (e.g., in a network offering high-speed links and/or a large RTT). Once this buffer is found to be a bottleneck, its size can be increased by using a TCP window-scale option [3]. Too large a window, however, leads to the bursty packet emission, which may result in the frequent packet losses. Thus, careful consideration is necessary when the receiver's advertised window size is determined.

3. **Sender–Side Configurations:**

   The content service providers or the operators of WWW and ftp servers sometimes limit the transmission rate of each TCP connection in order to ensure that the resources of the access network and the end hosts are shared fairly among clients. If such a rate control is adopted as the service policy, there is no way to improve the performance obtained by individual users.

   Another server–side bottleneck is due to the packet-processing overhead at the sender host. For example, if we connect to a busy WWW server, the document download rate is very limited because of the processing overload of the sender host. This kind of the bottleneck should be resolved by upgrading the server's power.

The first step of our method is to identify the location of bottlenecks based on the measurement, which will be explained in the next subsection.

2.2 Bottleneck Identification

2.2.1 Measurement Parameters

We use the following equations that characterize the TCP throughput [4].

If the bottleneck is not located at the sender-side configuration, the expected window size of the TCP connection is given by

$$E[W] = \frac{2+b}{3b} + \sqrt{\frac{8(1-p)}{3bp} + \left(\frac{2+b}{3b}\right)^2}, \tag{1}$$

where p and b are respectively the packet loss rate and the number of packets whose arrival is notified by one acknowledgement (ACK) packet. In the original version of TCP the receiver should send an ACK packet for the receipt of each packet, But Clark [5] showed that the overall network load can be reduced without any performance degradation by using applying a *delayed ACK*, in which the receiver sends an ACK packet acknowledging the receipt of multiple (two or more) packets. Delayed ACK is widely used in recent TCP implementations, and its parameter b is typically set at 2. Note that $E[W]$ cannot be estimated if $p = 0$. But when the link bandwidth is a bottleneck, packet losses are likely to occur because of the inherent nature of the TCP mechanism: the TCP congestion control decreases the window size (i.e., the transmission rate) by detecting packet losses. If packet loss never occurs, there is no problem with the current network configuration.

To use Eq. (1) we need to know the packet loss probability along the path. As described before, however, the end host cannot detect the packet loss directly. One approach is to introduce an upper layer protocol for exchanging such information. Such an approach can be found in RTP (Real-Time Protocol) and RTCP (Real-Time Control Protocol) [6], But it limits the applicability of our approach because it requires special-purpose software at server and client hosts. Another approach is to utilize an active measurement tool such as `pchar`, but this requires additional traffic measurement. The simpler approach we have adopted is to use the following approximation. The packet loss rate p is determined by counting the number of *Triple Duplicate ACKs*, N_{TD}, and the total number of packets, N_p. The packet loss rate is then estimated as $p = N_{TD}/N_p$. Since these statistics can be collected during the ordinary network use, no additional resource consumption is introduced.

Once we have estimated the expected window size $E[W]$, we check whether the receiver buffer is sufficient by comparing $E[W]$ and the buffer size W_{max}. In [3], TCP throughput is also given by

$$\begin{aligned} B(p) &= \frac{\frac{1-p}{p} + E[W] + \hat{Q}(E[W])\frac{1}{1-p}}{RTT(\frac{b}{2}E[W]+1) + \hat{Q}(E[W])T_0\frac{f(p)}{1-p}}, && \text{if} E[W] < W_{max} \\ andby &= \frac{\frac{1-p}{p} + W_{max} + \hat{Q}(W_{max})\frac{1}{1-p}}{RTT(\frac{b}{8}W_{max} + \frac{1-p}{pW_{max}} + 2) + \hat{Q}(W_{max})T_0\frac{f(p)}{1-p}} && \text{otherwise,} \end{aligned} \quad (2)$$

where $\hat{Q}$ and $f(p)$ are determined by the following equations:

$$\hat{Q}(w) = \min\left(1, \frac{(1-(1-p)^3)(1+(1-p)^3(1-(1-p)^{w-3}))}{1-(1-p)^w}\right) \quad (3)$$

and

$$f(p) = 1 + p + 2p^2 + 4p^3 + 8p^4 + 16p^5 + 32p^6. \quad (4)$$

If the receiver buffer is the bottleneck (found by using Eq. (2)), TCP throughput is given approximately by

$$B(p) \approx \frac{W_{max}}{RTT}. \quad (5)$$

T_0 is obtained from the retransmission timeout value (RTO) which is necessary to detect the packet at the end host. Since the IP, the protocol underlying the TCP, is connectionless, the TCP sender is not directly notified of packet loss. The TCP therefore uses an acknowledgement-based protocol to inform the sender of packet losses. There are two ways this is done, and one is that the sender host decides thata packet is lost if the sender does not receive its ACK packet within the timeout threshold. This threshold is the RTO and is calculated by using the following equations:

$$\begin{aligned} RTO &= RTT_m + 4D \quad (6) \\ Err &= RTT - RTT_m \\ RTT_m &= RTT_m + 0.125Err \\ D &= D + 0.25(|Err| - D). \end{aligned}$$

Note that the actual RTO value is usually not used as the timeout threshold because of the granularity of the TCP timer. For example, FreeBSD 3.4 has a 500-msec timer, and the value of RTO is rounded off by the unit of 500 msec.

The second way to detect packet loss is to check the sequence numbers of the acknowledgements. When a packet is lost within the network, the sender host receives an ACK packet with the same acknowledgement number as the previous one. Those ACK packets are called *duplicate ACKs*. The sender host, however, cannot determine whether a duplicate ACK is caused by a lost segment or simply by a mis-ordering of packets, and some "margin" for ignoring duplicate ACKs is necessary. The sender host does not actually recognize that a packet has been lost until it receives at least three duplicate ACKs have been received. The sender then retransmits the packet and reduces the congestion window size by a factor of 2.

To use these equations to identify the bottleneck, we need to obtain (1) the packet loss rate p, (2) the maximum window size W_{max} at the receiver host, (3) the number b of packets whose arrival is notified by one ACK packet, (4) the round-trip time RTT, and (5) the packet retransmission timeout T_0. We also need to measure the TCP throughput because the above formula for predicting it is occasionally incorrect. The packet loss probability can be determined using the method described above. Since W_{max} and b are the configurable parameters, they can simply be obtained by the kernel configuration of the operating system. Since the measurement is performed at the receiver-side, it is not easy to get the round-trip time (RTT) from the trace results of TCP headers. We therefore need to obtain RTT values by using measurement tools such as `ping`. After we collect a set of RTTs, we can evaluate the individual RTO values by using Eq. (7) and then calculate the mean value. We finally calculate the measured TCP throughput by observing traced TCP headers and counting the total number of bytes transmitted. The physical capacity of the bottleneck link is helpful for bottleneck identification and can be obtained by using the method described in /refmypaper.

2.2.2 Identification Method

In this subsection we describe our new bottleneck identification method. According to the identification framework described in Subsection 2.1, we examine the receiver-side configuration, sender-side configuration, and network condition. Specifically, we

1. Measure the network parameters: RTT, RTO, packet loss rate, TCP throughput, and the physical capacity

of bottleneck link.

2. Change the receiver buffer size and measure the TCP throughput again. If the throughput is not increased, the bottleneck is likely to be on the sender-side. Alternatively, it can be conjectured by a degree of the difference between the estimated TCP throughput and the measured throughput because Eq. (2) do not consider the sender–side bottlenecks.

3. Check whether the receiver buffer size is sufficient. This can be done by using Eq. (1). If the buffer size is insufficient, increase it. Otherwise the current cause of the limited performance is in the network. We can confirm this by increasing the receiver buffer size. When the bottleneck is the link bandwidth, increasing the size of the buffer simply increases the packet loss rate and RTT.

4. When the receiver buffer is too small, its size should be increased so that desired performance can be obtained. This should be done carefully, however. As shown in Eq. (5), the receiver buffer size and the resultant TCP throughput are linearly related when the receiver buffer is the bottleneck. Anyway,the bottleneck may shift to another part of the network when the receiver buffer size is increased.

2.3 Experimental Results

In our experiments we used a `ping` program to obtain the round-trip times (RTTs) and used `tcpdump` [7] to capture TCP headers. We also used the bandwidth prediction method presented in Ref. 12 to obtain the physical link capacity of each link.

We placed the receiver host (client) at Osaka University and choose five sender hosts (servers): Osaka City University, Sony Communication Network Corporation (`ftp.so-net.ne.jp`), Internet Initiative Japan Inc. (`ftp.iij.ad.jp`), Ryukoku University (`ftp.ryukoku.ac.jp`), and The Institute of Advanced Media Arts and Sciences (`ftp.iamas.ac.jp`). The sender hosts and their abbreviated names are listed in Table 1.

Our measurement and bottleneck identification results are summarized in Table 2, where each row contains nine entries: the sender host, the bandwidth of the bottleneck link, the receiver buffer size, the round-trip time,

Table 1: Sender hosts.

abbreviated name	network	host
ocu	Osaka City University	–
sonet	Sony Communication Network Corporation	`ftp.so-net.ne.jp`
iij	Internet Initiative Japan Inc.	`ftp.iij.ad.jp`
ryu	Ryukoku University	`ftp.ryukoku.ac.jp`
iamas	Institute of Advanced Media Arts and Sciences	`ftp.iamas.ac.jp`

Table 2: Results of measurements and bottleneck identification.

sender	bottleneck	buffer size	RTT (ms)	loss (%)	measure	estimate	buffer	class
`ocu`	1.47 Mbps	8 KB	4.03	0.000187	1.35 Mbps	1.62 Mbps	insufficient	1
		32 KB	17.4	0.000946	1.36 Mbps	2.57 Mbps	sufficient	
		64 KB	21.7	0.00268	1.35 Mbps	1.21 Mbps	sufficient	
`sonet`	19.3 Mbps	8 KB	17.7	0.00007	2.13 Mbps	2.62 Mbps	insufficient	3
		32 KB	23.4	0.00293	9.66 Mbps	10.5 Mbps	insufficient	
		64 KB	23.4	0.00406	15.6 Mbps	20.8 Mbps	insufficient	
`iij`	29.6 Mbps	8 KB	20.8	0.000573	2.83 Mbps	3.03 Mbps	insufficient	2
		64 KB	22.9	0.000288	4.33 Mbps	35.3 Mbps	sufficient	
		128 KB	20.7	0.000867	4.78 Mbps	21.7 Mbps	sufficient	
`ryu`	1.51 Mbps	8 KB	35.8	0	0.76 Mbps	1.74 Mbps	insufficient	2
		32 KB	34.4	0	0.79 Mbps	7.25 Mbps	insufficient	
		64 KB	36.1	0	0.72 Mbps	13.8 Mbps	insufficient	
`iamas`	33.1 Mbps	8 KB	56.7	0.00018	0.85 Mbps	1.16 Mbps	insufficient	2
		32 KB	65.8	0	1.32 Mbps	3.86 Mbps	insufficient	
		64 KB	65.7	0.000361	1.32 Mbps	11.1 Mbps	sufficient	

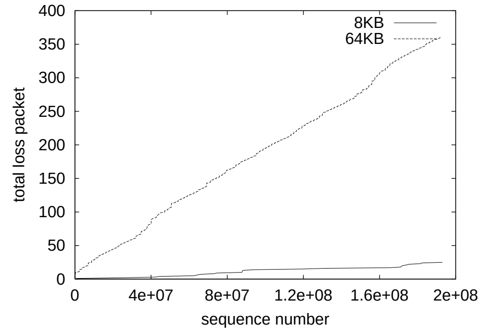

Figure 2: Osaka City University: Number of lost packets.

the packet loss rate, the measured TCP throughput, the estimated TCP throughput, A "buffer" entry specifying whether or not the buffer size was sufficient (which was determined using Eq. (1)), and a number indicating whetherthe bottleneck was due to (1) the network configuration, (2) the sender-side configuration, or (3) the receiver-side configuration.

In the `sonet` case the receiver buffer was the bottleneck because the RTT value was small, the receiver buffer size was evaluated as "insufficient," and the socket buffer size was proportional to the measured TCP throughput. The same results were obtained in `ocu`, `iij`, and `iamas` cases when the socket buffer size was set to 8 KB.

The `ocu` case shows a different result: the bottleneck moved from the buffer to the link bandwidth when the buffer size was changed from 8 KB to 64 KB. The number of packets when the receiver buffer size was set to 8 KB and 64 KB is shown in Figure 2. Note that the horizontal axis of the figure is the sequence number of packets received by the receiver host. As shown in the figure, the number of lost packets was quite small when the buffer size was 8 KB, which means that the bottleneck was not due to the network configuration. Furthermore, the number of lost packets is increased in proportion to time, which is a typical observation when the bottleneck is the link bandwidth.

In the `iij` case the maximum TCP throughput was expected from the measured bottleneck link bandwidth to be about 30 Mbps (see the second column). It is verified by the estimated TCP throughput (the seventh column). The maximum throughput measured, however, was about 4.8 Mbps. If throughput is limited by the network congestion, the packet loss rate would be expected to be high. However, there is no changes in the packet loss rate contrary to `ocu`. Such results are also found in the `ryu` and `iamas` cases. These cases were classified as ones in which the sender-side configuration limits the performance. There are two possible reasons for this sender-side limitation:

- **Insufficient bandwidth of the first link from the sender:**

  If the bandwidth of the first link is smaller than the packet-processing speed at the server, the packet sending rate becomes equal to or less than the bandwidth of the first link. Accordingly, the rest of the packets are waiting for forwarding at the input buffer. Unlike the intermediate routers, the input buffer of the first link seldom overflow, since the sender can adjust its sending rate according to the bandwidth available for the first link. Imagine an Ethernet-like network as an example. The data link layer protocol (the MAC layer in the case of the Ethernet) can control the sending rate at the server. As a result, the packet loss rate and RTTs are small even though the link bandwidth is the bottleneck.

- **Server-limited available bandwidth**

  Recently, some operating systems (such as Linux, FreeBSD, and Windows Servers) support mechanisms allowing the ftp server to limit the sending rate of each user. If such a restriction is used, the server does not send packets at the rate exceeding the predetermined bandwidth. Then the packet loss rate and RTTs are kept low because the packets are seldom queued within the networks.

To discriminate these two causes, we need to measure the link bandwidth of the first link from the server. When the sender host is within the target network, these two bottlenecks could be easily distinguished by information considering the link bandwidth and whether or not the server limits the available bandwidth. If the server is outside the target network, on the other hand, it is difficult to discriminate these two causes. However, if we consider the capacity dimensioning on the target network as just in the current case, this identification is no longer meaningful.

Confidence intervals for the TCP throughputs in the ocu, sonnet, and iij cases are listed in Table 3. The very

Table 3: Confidence intervals for estimated TCP throughput.

server	buffer size	min	mean	max
`ocu`	8 KB	1.64	1.64	1.64
	32 KB	1.40	1.40	1.40
	64 KB	1.97	1.98	1.98
`sonet`	8 KB	3.78	3.78	3.78
	32 KB	12.98	12.98	12.98
	64 KB	23.30	23.35	23.40
`iij`	8 KB	3.30	3.30	3.30
	32 KB	11.33	11.34	11.35
	64 KB	22.83	22.84	22.84

narrow confidence intervals indicate that we collected enough data to be confident of the results.

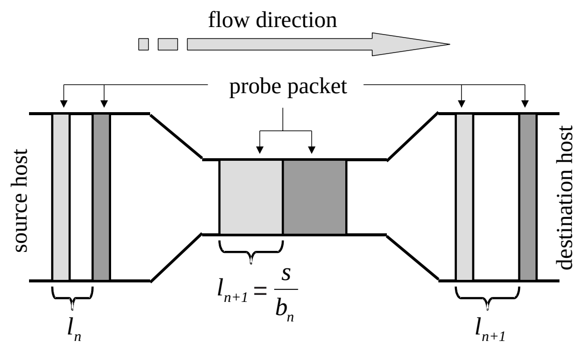

Figure 3: Packet-pairmeasurement method.

3 Measurement of Bottleneck Link Utilization

3.1 Issues in the Measurement of Link Utilization

In this subsection we describe the procedure used to measure the link utilization. It is based on the “Packet Pair” method, which is well known and widely used to measure the capacity or available bandwidth of a bottleneck link [8-10]. In the packet-pair method the measurement host sends back-to-back packets, and the bandwidth is calculated from the intervals between two back-to-back packets. The difference between the arrival times corresponds to the transmission time of the header packet. This difference may be increased according to the value of the link bandwidth but never decreases when the packets pass through links with larger capacities. As a result, the difference between the times two packets arrive at the destination host is equal to the header packet’s transmission time on the bottleneck link. We can also calculate the available bandwidth between the measurement host and destination hosts, b_a, easily by using the relation $b = s/l$, where s is the probe packet size and l is the interval between two packets. Figure 3 illustrates the mechanism of the packet-pair method.

There are, however, some difficulties in using the packet-pair method to measure the utilization of the bottleneck link. One is that the packet-pair approach does not necessarily measure the bandwidth of the actual bottleneck link. It first finds the link with the smallest capacity, and then it finds the link with least available bandwidth separately. The packet-pair approach assumes that those links are the same, but they may not be. A link with a large capacity is congested and it may be the bottleneck link on a path only when that link is congested. For capacity dimensioning, we want to know the available bandwidth of the actual bottleneck link. A

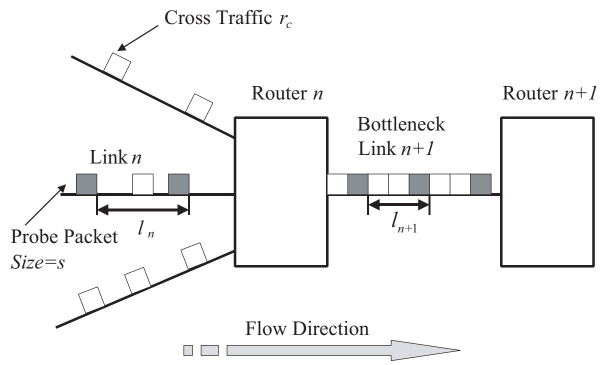

Figure 4: Two back-to-back packets.

second difficulty is that the influence of the measurement packet load should be taken into account but is not considered in the packet-pair approach. Even if the bottleneck link utilization reaches 1 and the available bandwidth b_a becomes 0, the packet-pair approach does not estimate the available bandwidth to be 0. The measurement packets waste some of the bandwidth available on the bottleneck link and the minimum value of a becomes $b/2$, which is apparently incorrect. The new estimation method in the next subsection solves these problems and estimates the bottleneck link utilization accurately.

3.2 Estimation of Bottleneck Link Utilization

The utilization of the bottleneck link cannot be estimated without knowing the capacity of that link, and this can be determined by several measurement tools such as `pathchar` and `pchar` and tools derived from them [11]. An accurate bandwidth estimation method based on `pathchar` was also proposed in [12]. After using one or more of these tools to determine the capacity of the bottleneck link, we estimate the amount of cross traffic passing through the bottleneck link. We can do this by using the packet-pair technique described in the previous subsection.

Figure 4 illustrates how the link utilization can be calculated when the bottleneck is the $n+1$st link from the measurement host. Before two back-to-back packets arrive at the router n, the time difference betweenthe beginnings of their headers is equal to the transmission time on the path from the measurement host to the router n. Let b_n be the physical capacity of the link n and let r_c be the throughput of the cross traffic passing through the link $n+1$. When packets are forwarded in a first-come-first-served (FCFS) fasion at the router, the dispersion

l_n of two packets is given by the difference between the time the last bit of the first packet arrived at router n and the time the last bit of the second packet arrived at the router. When there is cross traffic on the nth link, the probing packets experience additional queueing delays and the dispersion becomes l_{n+1}. If we neglect the packet loss at the router n, the amounts of input and output traffic are same. That is,

$$l_n r_c = l_{n+1} b_n - s. \tag{7}$$

Thus we obtain the following equation for the throughput of the cross traffic:

$$r_c = \frac{l_{n+1} b_n - s}{l_n}. \tag{8}$$

The utilization u of the bottleneck link u_n is thus given by

$$u_n = \frac{r_c}{b_n} = \frac{l_{n+1} b_n - s}{l_n b_n}. \tag{9}$$

The value of u_n can thus be determined by using the methods described in [12] to measure the capacity of nth link, b_n, and using the packet-pair method to measure the back-to-back packet dispersions l_n and l_{n+1}.

3.3 Measurement and Analysis of Packet Dispersions

Theoretically, the utilization u_n is given by Eq. (9). But it, of course, may be inaccurate because of measurement errors. This subsection explains how we minimized these errors in our measurements.

Figure 5 illustrates our measurement environment. To measure the packet dispersions l_n and l_{n+1}, we respectively sent "ICMP ECHO request" packets to the nth and $n+1$st routers. In what follows we will use the term "up-path" for the path from the measurement host to the target router and will use "down-path" for the path in the opposite direction. Assuming that the bottleneck link is on the up-path and that queueing delay does not occur at any routers on the down-path, we measured the packet dispersions by observing the intervals of the packets returned from the routers. In our experiments the measurement host was in Osaka City University, and the target host was in Osaka University. The bandwidth of the bottleneck link, which was the third hop from the measurement host (i.e., $n = 3$), was 6 Mbps. The parameters we changed in the experiments were the number N of back-to-back packets and the packet size s.

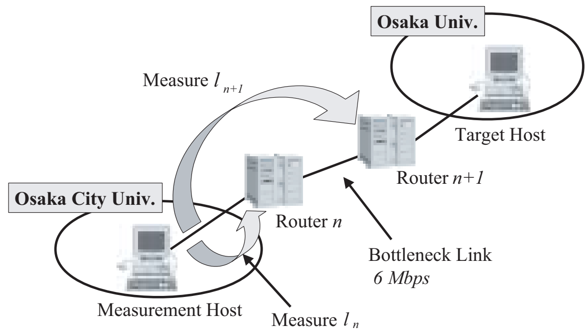

Figure 5: Measurement environment.

We measured the link utilization under various traffic conditions by changing the measurement time. The first measurement was started at 05:20 on Feb 16, 2001 (JST). At that time the load on the bottleneck link (the most congested link) was light ($u_n = 0.159$). To determine the appropriate value of packet size s, we measured packet dispersion while changing the packet size s. The number of probe packets N was set to 15, and the measurement was repeated until a 1000 dispersions were collected. Figure 6 shows the distribution of packet dispersion for probe packet sizes 42 bytes and 1442 bytes. When the size of the probe packets was 42 bytes there was only a slight difference between l_n and l_{n+1}. On the other hand, when the size of the probe packets was 1442 bytes, l_{n+1} became larger than l_n. This difference was caused by an interference of the cross traffic. We mentioned in the previous subsection that the cross traffic causes additional queueing delays increasing the packet dispersion, so the cross traffic load can be determined from the dispersion. The probe packets should thus be as large as possible if the link utilization is to be estimated accurately. If the probe packets are small, queueing may not occur at the bottleneck router because the transmission delay s/b_n becomes much smaller than the packet dispersion l_n at that router. As illustrated in Figure 4, when queueing occurs at the bottleneck router a space in the dispersion of packets on link n is filled with the cross traffic at the link n. When queueing does not occur at the bottleneck router, a space in the dispersion of packets on link n may remain unchanged at the bottleneck link, and then the dispersion of the bottleneck link l_{n+1} becomes larger than the dispersion when queueing occurs. Thus a large l_{n+1} leads to an overestimation of the amount of cross traffic, and consequently to an overestimation of the utilization of the link n.

The packet dispersion obtained in measurements starting at 15:26 on Feb 12, 2001 (JST), when the bottleneck link was heavily loaded ($u_n = 0.776$), are shown in Figure 7. Comparing this figure with Figure 6, we can see

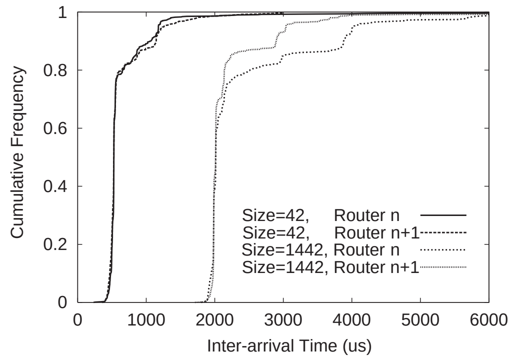

Figure 6: Distribution of packet dispersion when the traffic is light.

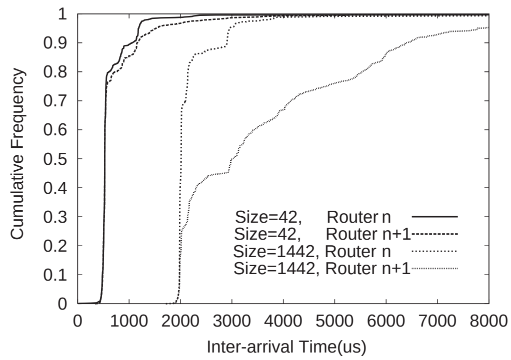

Figure 7: Distribution of packet dispersion when the traffic load is heavy.

that the dispersion of the probe packets passing through the congested bottleneck link l_{n+1} becomes large due to a large amount of the cross traffic. And in Figure 7 we can the influence of the cross traffic more clearly when the probe packet is larger.

Measurements of packet dispersion are more reliable when larger numbers of probe packets are used, and there are two reasons for this. One is that we need to send $M+1$ packets in order to obtain M dispersions, and such an overhead is not negligible when the number of packets is small. The second reason is that a longer sequence of the probe packets is less affected by measurement errors. When we send only two packets, the dispersion l_n is given by

$$\hat{l}_n = (T_1 \pm \delta_1) - (T_2 \pm \delta_2), \tag{10}$$

where T_i and δ_i are respectively the time at which ith packet is received and the error of this time. These errors cannot be reduced by repeating two-packet measurements, but if we send M packets for each measurement we can decrease the error by calculating the mean value of measured dispersions. The mean value of the dispersions

Table 4: Estimation results.

cross traffic throughput (actual utilization)	packet size	number of probe packets	packet dispersion link n (μs)	link $n+1$ (μs)	estimated utilization
0.95 Mbps (15.9%)	1442	15	2134	2320	22.8%
	542	15	1133	1204	45.5%
	42	15	605	613	92.5%
	1442	3	2371	2635	33.8%
1.7 Mbps (27.9%)	1442	15	2145	2431	27.8%
2.3 Mbps (38.5%)	1442	15	2144	2522	32.1%
4.1 Mbps (69.5%)	1442	15	2136	3123	60.3%
4.6 Mbps (77.6%)	1442	15	2128	3570	81.6%

is given by

$$\begin{aligned}\hat{l}_n &= \frac{1}{M-1}\sum_{i=1}^{M-1}((T_i \pm \delta_i) - (T_{i+1} \pm \delta_{i+1})) \\ &= \frac{1}{M-1}((T_1 \pm \delta_1) - (T_M \pm \delta_M)). \end{aligned} \quad (11)$$

Thus, an M-packet sequence can reduce the effect of errors by a factor of about $1/(M-1)$. Note that using too long a sequence of packets to measure dispersions are disadvantageous because it puts too heavy load on the bottleneck router, causing packet loss due to buffer overflow. Accordingly, we must find the optimal number of probe packets to use when measuring packet dispersions. This is one of our future research topics.

After measuring a set of dispersions, we excluded outlying values by discarding the highest and lowest 2.5% of the measured values.

3.4 Experimental Results

Results obtained when measuring the utilization of the bottleneck link are listed in Table 4.

The values in the first column are the cross traffic throughput and, in parentheses, the utilization calculated from the router information. The next two columns are for packet size and the number of packets, and the two columns to the right of those are for the packet intervals of the bottleneck link and the link just past the bottleneck link. The estimated utilization values are in the final column. Our method obviously provides an accurate estimate of the link utilization. Most of our estimated utilizations differ less than 10% from the actual utilizations (values in parentheses in the leftmost column), regardless of the traffic load on the bottleneck link. Our estimation method is enough accurate to be used in the capacity dimensioning described in the next section.

The table also shows the effects of the number of packets and the packet size. Comparing top three lines, we can see that smaller probe packets reduce the accuracy of the estimation. This is because queueing did not occur at the bottleneck router. The effect of the number of packets can be seen by comparing the first and fourth lines in this table, Which show that the estimated results are inaccurate when the number of probe packets is small. The utilization estimated when the number of packets is 3 is about 50% larger than the more accurate one estimated when the number of packets is 15. Note again that the larger number of packets gives a more accurate estimate but wastes the bandwidth of the bottleneck link by filling it with meaningless traffic.

4 Capacity Dimensioning Based on Traffic Measurement

In this section we explain how our measurement-based capacity dimensioning is performed. Our dimensioning method is intended to satisfy the QoS required from the end user's point of view, and the target QoS metric we consider in this section is TCP throughput. We have shown in Section 2 that link capacity is not the only possible performance bottleneck and that increasing the link bandwidth does not necessarily improve performance for end users. Performance might instead be limited by the sender-side and receiver-side configurations. We thus have first to first identify the bottleneck by statistically characterizing the current network and server configurations.

Our goal of the network dimensioning is to increase the end-user's throughput from t to $t = t + \Delta t$, and throughout this section we will call a TCP connection with throughput t a *target connection*. The difficulty of ensuring such a connection is due to the fact that increasing bandwidth of a TCP connection by Δt does not necessarily increase the target user's throughput by that amount because the additional bandwidth is also available to other users. In determining the link capacity, it would be best if the future traffic demands could be accurately determined from the current traffic. But because this is very difficult and probably impossible, we have to obtain the target throughput by repeating the processes explained in the following subsections. We first explain the capacity-dimensioning process in the case where the link capacity is not the bottleneck and then discuss the process in the case where it is.

4.1 Bottlenecks in the Sender-side or Receiver-Side Configurations

As discussed in Section 2, when the bottleneck is the sender-side configuration the throughput can be increased by eliminating the rate control at the servers or increasing the processing power there. On the other hand, when the bottleneck is the socket buffer of the receiver the throughput can be increased by providing more buffer space.

When TCP throughput is limited by the buffer size f on the receiver side, the throughput t is given by the following equation [4]:

$$t = \frac{f}{RTT}. \tag{12}$$

Accordingly, the throughput t' is obtained when

$$f' \geq t' RTT. \tag{13}$$

If the buffer size is too large, however, packets might be sent at a rate exceeding the capacity of the bottleneck link bandwidth. This would reduce throughput because a larger buffer causes the size of the TCP congestion window to fluctuate faster, leading to more frequent packet loss. To avoid this, we need to set an appropriate buffer size based on Eq. (13).

Increasing the buffer size to f' is sometimes insufficient because it can simply shift the bottleneck from the buffer size to the link, but let us suppose here that the buffer is actually the bottleneck. Let C be the bandwidth of the link with the smallest utilization (ρ) along the end-to-end path (ρ is not large because this link is not the current bottleneck). Furthermore, let $A = \rho C - t$ be the bandwidth used by the background traffic (i.e., the traffic connections on connections other than the target connection). The available bandwidth of the link is then equal to $(1 - \rho)C$. Increasing the buffer size to f' is expected to increase the throughput by Δt. Of course, if $\Delta t > (1 - \rho)C$, the target throughput cannot be achieved without increasing the link capacity.

Because the IP routing mechanism is based on the shortest-path-finding algorithm, we can assume that increasing the physical capacity of the bottleneck link does not affect IP routing. And because the link was underutilized before buffer resizing, we can also assume that the total throughput of the background traffic remains A. The updated link capacity C' must then be at least equal to the background traffic bandwidth plus the original link throughput plus the throughput increase. That is,

$$C' \geq A + t + \Delta t. \tag{14}$$

4.2 Bottlenecks in Network Links

When the bottleneck link is almost fully utilized in this case (i.e., when ρ is close to 1), packet loss occurs frequently. In this case, the bottleneck may be caused only by the target connection or may be caused by multiple connections (including the target connection). When the bottleneck is caused only by the target TCP connection the additional bandwidth will be used by the target connection and the required capacity of the bottleneck link can be determined by using (14). When the bottleneck is caused by multiple TCP connections, on the other hand, adding bandwidth will also increase the cross traffic. In discussing this problem here we assume that the bottleneck link is an M/M/1 queueing system. Our framework, however, is not limited to such a system.

When L is the average size of the packets passing through the bottleneck link, the packet arrival rate $\lambda is (A+t)/L$ and the service rate μ is C/L. The utilization ρ of the link capacity can then be written

$$\rho = \frac{\lambda}{\mu} = \frac{A+t}{C}, \tag{15}$$

and we get the following equation for the packet service time T:

$$T = \frac{1}{\mu}\frac{1}{1-\rho}. \tag{16}$$

Increasing the bottleneck link capacity by a factor of α leads to

$$\mu' = \frac{\alpha C}{L} = \alpha\mu. \tag{17}$$

Here we assume that the amount of the cross traffic is still large and the utilization of the bottleneck link is unchanged. Using Eqs. (16) and (17), we obtain the following equation for the new packet service time T':

$$T' = \frac{1}{\mu'}\frac{1}{1-\rho} = \frac{T}{\alpha}. \tag{18}$$

Note that the service time is shorter after the link bandwidth has been upgraded, so the RTT of each connection is also shorter. In [4] TCP throughput is modeled as

$$t = \frac{1}{RTT}\sqrt{\frac{3}{2bp}}. \tag{19}$$

Letting n be the number of intermediate routers in the round-trip end-to-end path, we can calculate the RTT from

$$RTT = \sum_{i=1}^{2n} T_i + S + \sum_{i=1}^{2n+2} D_i, \tag{20}$$

where T_i is the packet service time at the router i, S is the packet processing time at the sender host, and D_i is the propagation delay of the link i. The RTT after the link bandwidth is increased can be calculated from

$$RTT' = T' + \sum_{i=1}^{2n} T_i - T + S + \sum_{i=1}^{2n+2} D_i = T' + T_o, \tag{21}$$

where

$$T_o = \sum_{i=1}^{2n} T_i - T + S + \sum_{i=1}^{2n+2} D_i. \tag{22}$$

After increasing the link capacity when the link bandwidth is the bottleneck, the packet loss rate would become smaller than p, but we assume it to be unchanged. Then, the following condition should be satisfied in order to realize the target throughput t':

$$t' \leq \frac{1}{RTT'}\sqrt{\frac{3}{2bp}}. \tag{23}$$

Usingrelations. (18), (21), and (23), we can determine the new link capacity as

$$\alpha \geq \frac{T}{\frac{1}{t'}\sqrt{\frac{3}{2bp}} - T_o}. \tag{24}$$

Noticeably, Eq. (24) suggests that that the new throughput values depend on the sum of delays except the service time at the bottleneck router T_o. If T_o is small (e.g., if the sender is close to the receiver), the target user can exploit the large portion of the bottleneck link bandwidth by reducing the service time from T to T'. In this case the cross traffic will increase less than the target user's traffic does, and we therefore set the capacity of the bottleneck link at C' to satisfy

$$C' \geq \frac{t'}{t}C. \tag{25}$$

This relation can be derived by setting $T_o = 0$ in relation (24).

When the sender host is far from the receiver host and RTT is very large, the improvement of the throughput of the target connection is quite limited because the sum of delays T_o is still large. Accordingly, it is necessary

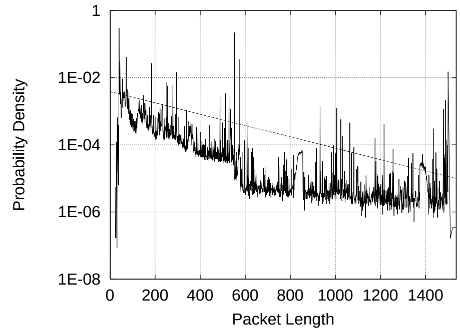

Figure 8: Distribution of packet size at a backbone router.

to adjust the link capacity until the bottlenecks of the background connections move from the current bottleneck link to another link. The necessary capacity is not easily estimated, but our proposed framework can be used to simply get rid of the bottleneck repeatedly. We have assumed that the behavior of the packet forwarding at the bottleneck router is similar to that in an $M/M/1$ queueing system (see relation (24)), but this assumption seems unwarranted. For example, the distribution of packet size is not actually exponential. See Figure 8, where the dotted line shows the exponential distribution with the same mean. In that case, we can use our proposed design framework by utilizing the $M/G/1$ queueing system. This can be done by appropriately changing Eqs. (16), (17), and (18).

Of course we cannot guarantee the target throughput to every user even if the our framework is used. This is a problem inherent in best-effort-based protocols, including those used in the Internet. Our capacity designing process has to be repeated until the throughput performance required by the end users is satisfied.

5 Conclusion

In this paper we have proposed some measurement techniques providing the information needed for capacity-dimensioning in the Internet.

First we proposed a bottleneck identification process based on traffic measurement and characterized the current network performance by measuring the packet loss rate, RTT, RTO, TCP throughput, and socket buffer size of the client host. Our proposed identification method can identify three kinds of the bottlenecks: the receiver-side configuration, sender–side configuration, and the network condition-and experimental results have shown that our method can identify the bottleneck in an end-to-end TCP communication.

We next proposed a method for estimating the utilization of the bottleneck link on the communication path between the end hosts. The characteristic of the packet dispersion which is required for our estimation methods has been discussed. We have explained the relation between the packet dispersion and the number and the size of probe packets. Experimental results have shown that our method can provide an accurate estimate of the link utilization.

Finally, we have proposed a design framework for determining the adequate link capacity on the basis of end-to-end measurement. We have described a capacity-dimensioning process bringing the TCP throughput to the target value. Using an $M/M/1$ queueing system for the router, we presented examples of the capacity-dimensioning process for the following cases:

- The link bandwidth is not a bottleneck.
- The link bandwidth becomes a bottleneck caused by only the target connection.
- The link bandwidth becomes a bottleneck caused by multiple connections sharing the link.

In the first two cases, we can resize the link capacity according to the target bandwidth. On the other hand, the capacity dimensioning should be done carefully because the background traffic sharing the bottleneck link affects the performance obtained by the target user.

As a future topic, the measurement technique and its accuracy improvement for some network parameters like link utilization should be discussed, which is applicable to our network dimensioning method.

References

[1] S. Blake, D. Black, M. Carlson, Z. W. E. Davies, and W. Weiss, "An architecture for differentiated services," *IETF RFC 2475*, December 1998.

[2] E.C.Rosen, A.V.Than, and R.Callon, "Multiprotocol label switching architecture," *IETF draft*, `http://search.ietf.org/internet-drafts/draft-ietf-mpls-arch-06.txt`.

[3] W. R. Stevens, *TCP/IP Illustrated, Volume1: The Protocols*. Addison-Wesley, 1994.

[4] J. Padhye, V. Firoiu, D. Toesley, and J. Kurose, "Modeling TCP throughput: A simple model and its empirical validation," in *Proceedings of ACM SIGCOMM'98*, pp. 303–314, September 1998.

[5] D. D. Clark, "Window and acknowledgement strategy in TCP," *IETF RFC 815*, July 1982.

[6] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A transport protocol for real-time applications," *IETF RFC 1889*, January 1996.

[7] LBNL's Network Research Group, "tcpdump," `ftp://ftp.ee.lbl.gov/tcpdump.tar.Z`.

[8] V. Paxson, "Measurements and analysis of end-to-end Internet dynamics," *Ph.D.Thesis*, *University of California Berkley*, April 1997.

[9] R. L. Carter and M. E. Crovella, "Dynamic server selection using bandwidth probing in wide -area networks," *TR-96-007*, March 1996.

[10] C. Dovrolis, P. Ramanathan, and D. Moore, "What do packet dispersion techniques measure?," in *Proceedings of IEEE INFOCOM 2001*, April 2001.

[11] "Caida measurement tool taxonomy," `http://www.caida.org/tools/`.

[12] K. Matoba, S. Ata, and M. Murata, "Improving accuracy of bandwidth estimation for Internet links by statistical methods," *IEICE Transaction on Communications*, vol. E84-B, pp. 988–995, June 2001.