A Study on the Internet Capacity Dimensioning Based on Traffic Measurement and its Statistical Analysis

Kazumine Matoba

Department of Informatics and Mathematical Science

Graduate School of Engineering Science

Osaka University

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 thesis, we first consider a new estimation method of the end-to-end traffic characteristics in the Internet, which is necessary to determine the capacity of the current link. We improve an existing measurement tool called `pathchar`, which measures the capacity of all links between end hosts. The measurement method of `pathchar` is innovative, but it still has many problems. If unexpectedly large errors are included or if route alternation is present during the measurement, the obtained estimation is much far from the actual one. We then propose a new method to eliminate those errors in estimating the bandwidth. To increase the reliability on the estimation, the confidence interval for the estimated bandwidth is important. For this purpose, two approaches, parametric and nonparametric approaches, are investigated in order to add the confidence intervals. Another important issue is the method for controlling the measurement period. If the link is stable, small measurement data is sufficient, while if the data is not sufficient, many measurements is necessary to obtain an accurate and reliable estimation. We propose a measurement method to adaptively control the number of measurement data sets. By our approach, a reliable bandwidth estimation can be established.

We next move to 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. 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 present the methodology to determine the link capacity.

Keywords

Bandwidth Measurement

Pathchar

Bottleneck

TCP (Transmission Control Protocol)

Network Dimensioning

Contents

List of Figures

List of Tables

1 Introduction

Network dimensioning for the Internet is becoming more important to provide a stable Quality of Service (QoS) to the customers of ISP (Internet Service Providers), and without it ISP would not be commercially successful, since QoS provided by ISP is now recognized as one of important measures in a competitive market. However, network dimensioning is not an easy task because the network is changing rapidly and frequently.

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

However, there is a fundamental limit in those architectures: the physical (and/or logical) path capacity should be determined a priori by network providers. Network dimensioning takes an essential role in determining the appropriate network capacity. For that purpose, the network providers should first determine QoS metrics for end users. Traffic generated by computer applications should then be characterized. Finally, the appropriate network resources satisfying those QoS metrics should be estimated. In the computer networks such as the Internet, those are very complicated, and an interdependence among them makes it more difficult. For just one simple example, let us consider Web applications. Even if the bandwidth provided by the network is insufficient, the application still works since 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, it is not clear whether the resultant QoS (the document transfer time in the case of Web browsing) may or may not satisfy the users. Anyway, a problem is that we have no established QoS metric which satisfies the Web users.

In the telephone network, a historical experience on estimation of call demand (in Erlang) and an Erlang loss formula has been successfully used for network dimensioning. The traffic estimation is not difficult because the number of blocked calls is of primary concern for performance monitoring. Also, the Erlang loss formula is robust in the sense that it does not require the information on the distribution of service times (connection holding times), but only the average. Then it can be used for capacity planning of the links in operational networks. Fortunately, the call blocking probability is user's perceived QoS metric as well as the network-

oriented QoS metric. This simple feedback loop of network dimensioning has led to the success of the stable and reliable telephone networks.

In the Internet offering data communication service, on the other hand, such an approach cannot be adopted since it is difficult to know or to at least estimate the traffic demand in advance, mainly due to the following two reasons. The one is that the Internet is growing drastically and therefore it cannot forecast future demands of user traffic. The other is due to the inherent characteristics of the Internet traffic. The situation is very different from telephone traffic, which is characterized by arrival rates of calls and holding times. A dominant of the Internet traffic is TCP-based application having a capability of adapting to network congestion; i.e., if the network falls into congestion, it defers transmission such that TCP connection is not *blocked*. Thus, the packet-level metrics of, e.g., the transmission time and/or loss probabilities at the routers are quite insufficient to identify the performance provided by the operating network. It suggests that the network monitoring only at the node and/or link is not sufficient for capacity planning.

Probably, network dimensioning for the loosely-coupled network just like the Internet needs a different approach from the conventional telephone network, where the network carrier has a responsibility of maintaining the network. Most important is that QoS metrics such as Web document download times can never be measured by the network providers, but only by the end users, because in the Internet, the processing of protocols higher than the layer-four (i.e., the transport layer) is performed at the end host. We believe that a first step for establishing a framework of network dimensioning in the Internet is to measure the network characteristics. It is then used for dimensioning the network resources for the users.

Recently, various tools have been developed to measure the traffic characteristics on the Internet. See, e.g., [3]. Essentially, there are two kinds of measurement tools; *passive* and *active* measurement tools. Using passive measurement tools like OC3MON [4] and MRTG [5], packets passing through the monitoring point are captured, and analyzed by inspecting packet headers (and sometimes, packet contents). Obtained statistics are very accurate in principle. As the name indicates, the passive monitoring tools do not consume network resources. However, its fundamental drawback is that the behavior of the end host, which takes an important role in data communications as described before, cannot be known from the obtained statistics. The QoS provided to end users, which must be characterized by an end-to-end fashion, is also not known.

On the other hand, active measurement tools have an ability to resolve the above problems since it is used at the end host (or, at the edge routers). Here, the "end user" includes ISP that wants to know the characteristics of the networks provided by other ISP's. In the active measurement tools, the packets are injected to the network and collects the returned information. An apparent drawback of the active measurement tools is that it consumes the network resources unless packet probing is adequately controlled. Another problem is that an accurate and reliable estimation becomes difficult due to the dynamic nature of the Internet.

As an example of the active measurement tools is `Pathchar` [6, 7], which is recently developed in order to measure latency, bandwidth, queueing delays and packet loss rate for every link between two end hosts. Its advantage is that it is not necessary to deploy new protocols with any special functions at both of routers and end hosts. `Pathchar` collects RTTs with various sizes of packets and estimates the link bandwidth according to the relation of RTTs and packet sizes. However, `pathchar` has several problems. It requires a large amount of statistics by throwing the large number of packets into the network, in order to improve the bandwidth estimation. It leads to an intrinsic problem that the increased traffic may cause congestion and an estimated value is biased by `pathchar` itself.

In this thesis, instead of pursuing the accuracy of the approach adopted by `pathchar`, we take another approach: we try to add a confidence in the estimation results. A recent version of `pathchar`, which is now called `pchar`, gives a confidence interval for the slope (by which the bandwidth estimation is derived), but it is insufficient for the user to rely on the obtained results. We investigate the calculation method to determine the confidence intervals for the estimated bandwidth. The control method for measurement time is also proposed to limit the unnecessary probes injected into the network.

The second step towards network dimensioning is to identify the existing bottleneck for meeting QoS requirements of the users. As we have emphasized in the above, the bottleneck exists not only at the network resources (including the bandwidth capacity, router's packet forwarding capability, etc.), but also at the end hosts in the Internet. It is apparent that increasing the network bandwidth does not necessarily lead to the QoS improvement. It may only result in the meaningless bandwidth investment if the bottleneck is located at the end hosts. In the current Internet where the major application is Web systems utilizing the http and TCP, we propose a bottleneck identification method for TCP-based applications. In TCP, possible causes of the bottleneck include a socket buffer size of the receiver, an available bandwidth of the link, and the packet processing at the sender. Actu-

ally, by carefully examining the characteristics of TCP connections, we can find the cause of the performance bottleneck as will be described in Section 3.

Once the bottleneck is found within the network, and the current link capacity is known by our improved bandwidth estimation method, we next determine how much the bottleneck link capacity should be increased to meet the QoS requirement of users. For the case of TCP connections, we consider 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. By taking account of the cross traffic, we will describe the determination method for the new link capacity in Section 4.

The rest of this thesis is structured as follows. Section 2 investigates how to improve the bandwidth estimation of `pathchar`. In order to find the bottleneck, we measured the network characteristics including RTT (Round Trip Time) and packet loss probability. Based on these characteristics, our bottleneck identification method is proposed in Section 3. Based on the measurement of the previous sections, we explain how our capacity dimensioning method works well in Section 4. Finally, we conclude our research in Section 5.

2 Improving Bandwidth Estimation

An accurate and reliable estimation of the bandwidth of links on the end-to-end path is a first step towards network dimensioning. There are many researches about end-to-end bandwidth measurement [8-11]. `Pathchar` [6, 7] is one of such tools to measure latency, bandwidth, queueing delays and packet loss rate for every link between two hosts. The advantage of `pathchar` is that it is not necessary to deploy new protocols or any special functions at both of routers and end hosts. `Pathchar` collects RTTs (Round Trip Time) with various sizes of packets and estimates the link bandwidth according to the relation of RTTs and packet sizes.

However, `pathchar` still has many problems: `pathchar` needs a large amount of statistics to improve the bandwidth estimation, which is obtained by throwing the large number of probe packets into the network. However, it has an intrinsic problem that the increased traffic may cause congestion and an estimated value may be biased by `pathchar` itself. Instead of pursuing the accuracy of the approach taken by `pathchar`, we take another approach to add a confidence in the estimation. A recent version of `pathchar`, which is now called as `pchar` [8], gives a confidence interval for the slope (by which the bandwidth estimation is derived), but it is insufficient for the user to rely on the obtained results. In this thesis, we investigate the calculation method to determine the confidence intervals for the estimated bandwidth. The control method for measurement time is also proposed to limit the unnecessary probes injected into the network.

2.1 A Brief Description on `Pathchar` and its Problems

2.1.1 A Brief Description on `Pathchar`

In this subsection, we summarize a bandwidth estimation method taken in `pathchar`. For more details, refer to [7].

`Pathchar` first collects RTTs between source and destination hosts. To measure RTTs, `pathchar` uses one of the ICMP packet, called a *TTL exceeded* message, which is also used in `traceroute` [12]. An IP packet has a TTL (Time To Live) field in the header. It shows the limit of the hop count that the packet can traverse. Before the router forwards the packet to the next hop, the value of the TTL field is decreased by one. When the TTL value becomes zero, the router discards the packet and returns the ICMP control packet to the

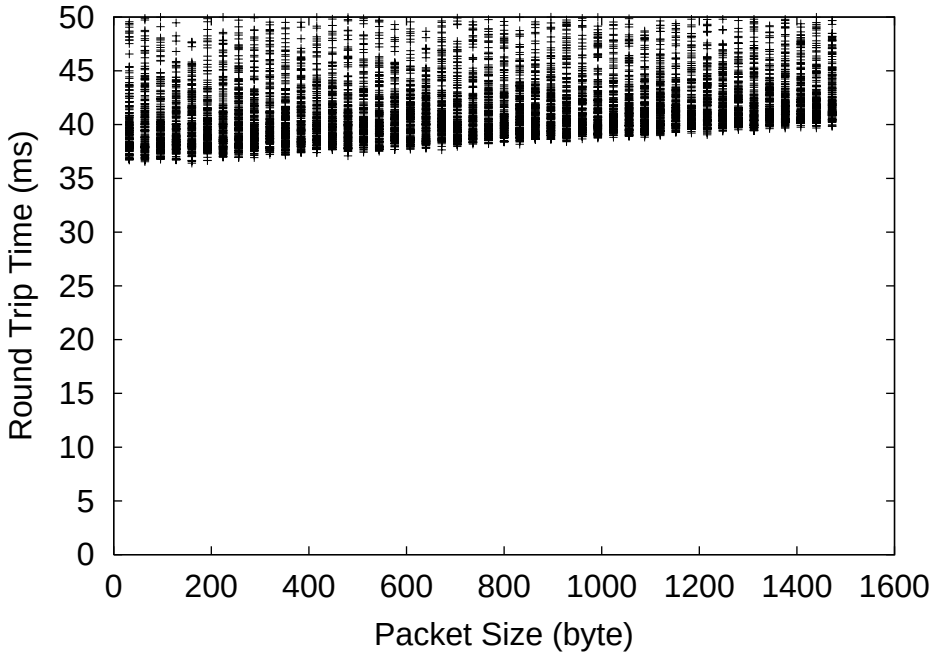

Figure 1: Distribution of RTT Values vs. Packet Size

source to inform that the validity of the packet is expired. This mechanism is necessary in order to avoid a loop of packet forwarding due to, e.g, some misbehaviors of the router. When the packet is sent with the value of the TTL field to be n, the ICMP control packet must be returned from nth hop router. The RTT value between the source and nth router on the path can then be measured by the source. `Pathchar` collects RTTs between the source and every intermediate router by changing the value of the TTL field.

The measured RTT value consists of (1) the sum of queueing delays, q_i, at router i $(1 \leq i \leq n)$, (2) the sum of transmission times to transmit the packet by the intermediate routers, (3) the sum of forwarding times f_i that router i processes the packet, and (4) the sum of propagation delays p_j of link j $(1 \leq j \leq n)$. That is, RTT_s, the RTT value for given packet size s, is represented by

$$RTT_s = \sum_{j=1}^{n} \left(\frac{s}{b_j} + \frac{s_{ICMP}}{b_j} \right) + \sum_{i=1}^{n} (q_i + f_i) + 2 \sum_{j=1}^{n} p_j, \quad (1)$$

where s_{ICMP} is a size of an ICMP error message and b_j is the bandwidth of link j.

A typical example for the relation between packet sizes and measured RTTs is shown in Figure 1. The results are obtained by setting the destination to be `www.gulf.or.jp` from our site. The TTL value was set to 16. It was obtained on Dec 18, 1999 12:54 JST. The figure shows that the RTT values were widely spread even for the fixed packet size. It is because the queueing delay at the router changes frequently by the network condition.

However, it is likely that several packets do not experience the queueing delays at any router by increasing the trials. Such a case actually appears in the figure as a minimum value of RTTs for each packet size. The minimum RTT for given packet size s, denoted by $minRTT_s$, is thus obtained by

$$minRTT_s = \sum_{j=1}^{n} \frac{s + s_{ICMP}}{b_j} + \sum_{i=1}^{n} f_i + 2\sum_{j=1}^{n} p_j. \tag{2}$$

Note that the packet size of the ICMP error message s_{ICMP} is fixed (56 bytes). Then, by collecting terms not related to the packet size and denoting it by α, the above equation can be rewritten as

$$minRTT_s = s\sum_{j=1}^{n} \frac{1}{b_j} + \alpha. \tag{3}$$

Eq.(3) is a linear equation with respect to the packet size s. It is just shown in Figure 1 if we look at the minimum RTT values. By letting the coefficient of the above equation be β_n, we have

$$\beta_n = \sum_{j=1}^{n} \frac{1}{b_j}. \tag{4}$$

Conversely, if we have β_{n-1} and β_n, we can obtain the bandwidth of link j as

$$b_j = \frac{1}{\beta_n - \beta_{n-1}}. \tag{5}$$

It is a key idea of `pathchar`.

As indicated above, a difficulty of `pathchar` exists in that, the network condition changes frequently in real networks such as the current Internet, and it is not easy to obtain proper minimum RTTs. Thus, `pathchar` needs to send many packets with the same size; it is a weak point of `pathchar` since those waste a large amount of link bandwidth to get a minimum RTT. Even after many RTTs are collected, some measurement errors must be contained. `Pathchar` solves this problem by a linear least square approximation.

2.1.2 Problems of `Pathchar`

The approach of the bandwidth estimation taken by `pathchar` is innovative, but it still has several problems as described below.

Reliability on Obtained Estimation

First, we cannot know whether the estimated bandwidth obtained by `pathchar` is reliable or not. `Pathchar` uses the linear least square fitting to calculate β_n, which implies that it assumes errors of minimum RTTs are

normally distributed [7]. However, we have no means to confirm whether errors follow a normal distribution or not. From this reason, it is necessary to consider another approach that can lead to bandwidth estimation independently from the error distribution. Such an approach is often called as a nonparametric approach. The nonparametric approach is already developed in `pchar` [8], an updated version of `pathchar`. While in `pchar`, the user can choose the parametric or the nonparametric method for estimation, it does not offer any criterion to decide which approach is better.

Efficiency of Measurements

The second problem is the efficiency of `pathchar`. `Pathchar` sends a fixed number of packets, but the amount of collected data must be changed according to the network condition to measure the link bandwidth within a reasonable level of accuracy. The authors in [7] then propose an *adaptive* data collection method to improve the efficiency of `pathchar`. They have shown that the required number of packets in `pathchar` can much be reduced if `pathchar` is equipped with an ability to send a different number of packets for each link estimation. In their proposal, the number of transmitted packets is decided by observing whether the even-odd range of bandwidth is converged or not. However, the range is not based on the reliability on the result and the method does not guarantee an accuracy in a *statistical* sense.

Exceptional Errors of RTTs

The third problem is that various kinds of errors are mixedly contained in minimum values of RTT. Nevertheless, `pathchar` assumes that the error of the minimum RTT is originated from the measurement noise only, and assumes the normal distribution for measurement errors. Basically, `pathchar` relies on the fact that the queueing delays at the intermediate routers can be removed by gathering a number of measurements since one or more packets must fortunately encounter no queueing delay by increasing the number of measurements. If the number of measurements is insufficient, the queueing delay may be involved. However, it may be able to be viewed as a Gaussian noise.

The problem is that we encounter the errors which cannot be explained by the Gaussian noise. One example is shown in Figure 2, which was obtained at Dec 21 08:39, 1999 JST by setting the destination as `www.try-net.or.jp` and the TTL value as 13. Several small values were observed during the measurement as shown in Figure 2. We need to introduce some method to remove such errors before the bandwidth estimation is

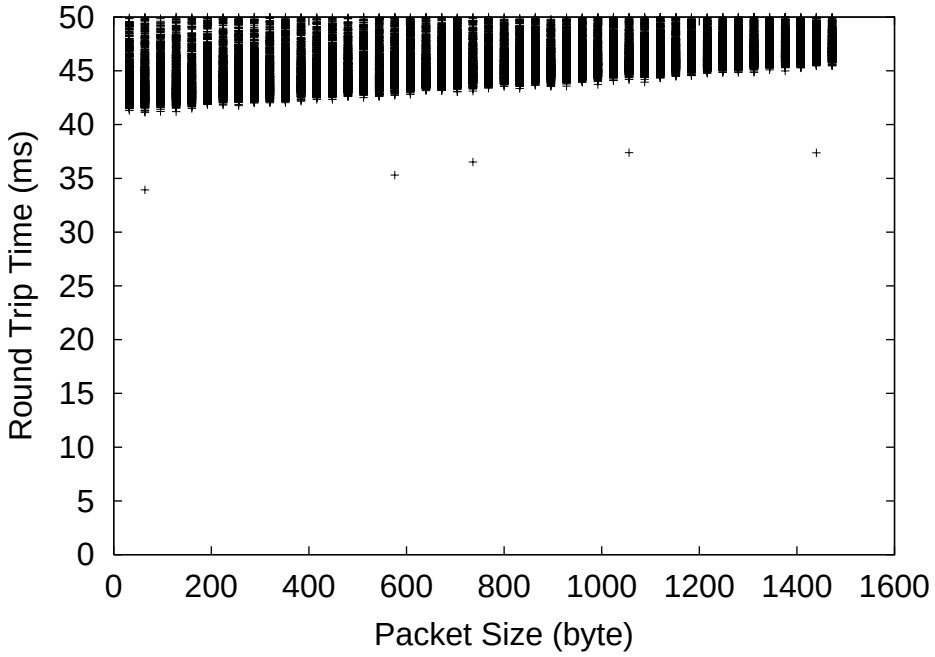

Figure 2: A Sample of Errors not Following a Normal Distribution

performed. For this purpose, we will apply a weighted least square fitting method as to be explained in Subsection 2.2.1.

A second example was obtained by route alternation. To get the bandwidth estimation, all probes should be relayed on the same path. If `pathchar` detects the route changes by checking the field of the source IP address in the returned ICMP packet, it simply discards the returned packet. The problem is that it cannot eliminate the case where the source IP addresses of the returned ICMP packets are same, but the relayed paths are different. Such a case may happen due to load balancing at routers [13]. In fact, we obtained such a measurement which is presented in Figure 3. It was observed at 8-th link destined for `www.kyotoinet.or.jp` at Dec 10 12:29,1999 JST. Figure 3 clearly shows that there exist two (or maybe more) paths during the measurement. To remove such an effect, we need to select the proper subgroup of RTTs for accurate estimation, which will be explained in Subsection 2.2.2

2.2 Accuracy and Reliability Improvements for Bandwidth Estimation

As we have discussed in the previous section, we need to solve several problems for obtaining accurate and reliable bandwidth estimation. For this purpose, we first examine two estimation methods; parametric and

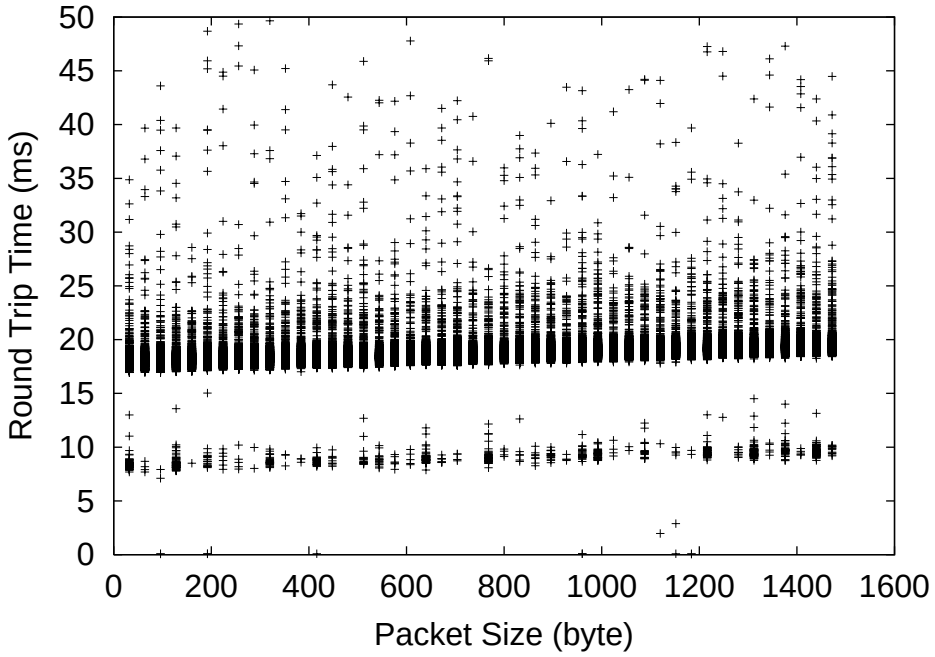

Figure 3: A Sample having Two Groups of RTTs

nonparametric approaches. The approach to obtain the confidence interval is also described in order to increase the reliability on estimation. Those are presented in Subsection 2.2.1. Our clustering method to pick up proper RTTs from two or more groups of RTTs is then presented in Subsection 2.2.2. An adaptive mechanism to control the measurement period is finally presented in Subsection 2.2.3. Our experimental results based on those methods are shown in the next section.

2.2.1 Accurate and Reliable Slope Estimation Methods

As having been described in the previous section, using the linear least square fitting method in `pathchar` implies that errors follow a normal distribution. Thus, unexpectedly large errors (as shown in Figure 2) significantly affect the accuracy of the estimated value. To eliminate such an negative influence, we introduce two estimation methods instead of the linear least square fitting method. One is an M-estimation method with a Tukey's biweight function [14], which is a sort of the parametric approach. It is robust to produce results with uniformly high efficiency. Because it presumes that almost all data is reliable and only some data includes unexpectedly large errors, the result is robust even if the large errors are contained as in our case. The other is a nonparametric linear least square fitting method which does not assume any distribution on measurement errors.

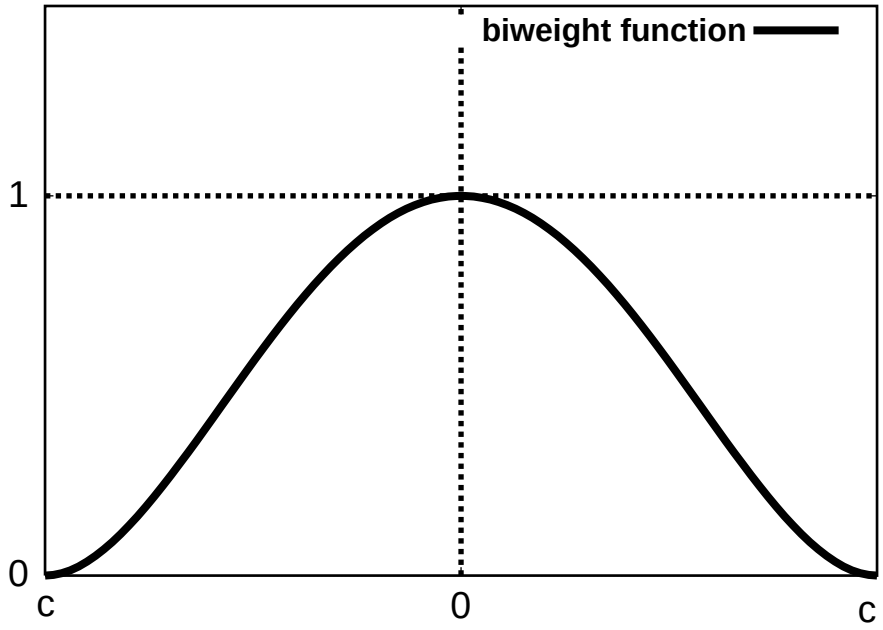

Figure 4: The Biweight Function

In what follows, we will describe two methods in turn.

M-estimation Method

In this subsection, we describe the weighted least square fitting method. With this method, the influence of the large error can be limited. Note that this method is applicable when the number of large errors is rare but not negligible. Otherwise, we need to use a nonparametric approach which is independent of an error distribution. The latter approach is presented in the next subsection.

The M-estimation method is an extension of a maximum likelihood estimation method. In the M-estimation method, the weighted least square fitting is iterated to calculate an appropriate weight. There are some variations in the M-estimation method, and we apply the Tukey's biweight function which is considered to be one of the best estimation methods [14]. In the Tukey's biweight function, a weight function is chosen as shown in Figure 4. It is apparent from the figure that the Tukey's biweight function is robust against the unexpectedly large errors if those occur infrequently. Let the number of kinds of the packet size be m. After we collect a minimum value of RTT for each packet size, we can estimate the slope according to the following procedure. Note that the slope means a coefficient β_n for router n (see Eq. (4)). In the following equations, we omit n for brevity. We label m kinds of packet size as $x_i (1 \leq x_i \leq m)$ and denote the minimum RTT for the packet size x_i as y_i.

1. The straight line is expressed by $y = \alpha + \beta x$, where x is a packet size and y is an ideal minimum RTT. We set initial values of a vertical intercept α and a slope β with the least square fitting method;

$$\alpha = \bar{y} - \beta x, \qquad \beta = \frac{\sum_{i=1}^{m} x_i y_i - m\bar{x}\bar{y}}{\sum_{i=1}^{m} {x_i}^2 - m\bar{x}^2}, \tag{6}$$

where $\bar{x}$ and $\bar{y}$ shows the mean of x_i and y_i.

2. Calculate the difference $|v_i|$ between the minimum RTT and the point on the straight line at the packet size x_i, i.e.,

$$|v_i| = |y_i - \beta x_i - \alpha|. \tag{7}$$

3. By obtaining the median of differences, the standard size of an error s is calculated as

$$s = median\{|v_i|\}. \tag{8}$$

4. By using the biweight function, we set a weight adjustment factor ω_i^{adj} for each difference;

$${\omega_i}^{adj} = \begin{cases} \left[1 - (\frac{v_i}{c\,s})^2\right]^2 & \text{if } |v_i| < c\,s, \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

where c is a constant value used as an index for making the total weight to be zero.

5. Let ω_i denote the weight of RTTs, which is given by

$$\omega_i = \frac{m{\omega_i}^{adj}}{\sum_{i=1}^{m} {\omega_i}^{adj}}. \tag{10}$$

We then estimate new values of α and β with the weighted least square fitting.

$$\alpha = \frac{\sum_{i=1}^{m} \omega_i\, y_i}{m}, \qquad \beta = \frac{\sum_{i=1}^{m} \omega_i\, x_i\, y_i}{\sum_{i=1}^{m} \omega_i\, x_i^2}. \tag{11}$$

6. After k iterations, we adopt α and β as solutions.

The parameter c in Eq. (9) controls a boundary for the errors contained in measured RTT values to be neglected. Through our experiments, we found that $c = 3$ and the number of iterations $k = 5$ are sufficient. Note that slopes of straight lines are always converged in our experimental results when we use above parameter values.

We then calculate a confidence interval with the M-estimation method. We introduce the following assumptions;

- For given packet size x, the random variable of the minimum RTT, Y follows the normal distribution, whose mean and variance are given by $\alpha + \beta x$ and σ^2, respectively.

- The measurements for each packet size are mutually independent.

The above assumptions imply that the set of slopes follows the normal distribution with mean β and variance σ_B^2, which are obtained from

$$\beta = \frac{\sum_{j=1}^{m}(x_j - \bar{x})(Y_j - \bar{Y})}{\sum_{j=1}^{m}(x_j - \bar{x})^2}, \tag{12}$$

$$\sigma_B^2 = \frac{\sigma^2}{\sum_{j=1}^{m}(x_j - \bar{x})^2}, \tag{13}$$

where σ is the standard error of the RTTs from the estimated line. It can be estimated from the measurement data as

$$\hat{\sigma}^2 = \frac{1}{m}\sum_{j=1}^{m}(y_j - \alpha - \beta x_j)^2. \tag{14}$$

From Eq. (12), we can calculate the values of the slope for links $n-1$ and n as $\hat{\beta}_{n-1}$ and $\hat{\beta}_n$, respectively. The variances for links $n-1$ and n are also estimated as $\hat{\sigma}_{n-1}^2$ and $\hat{\sigma}_n^2$ from Eq. (13), respectively.

Once those values are determined, we next calculate the confidence interval as follows. We can estimate the mean and variance for the difference of slopes as

$$\beta_u = \hat{\beta}_n - \hat{\beta}_{n-1}, \sigma_u^2 = \hat{\sigma}_n^2 - \hat{\sigma}_{n-1}^2. \tag{15}$$

The value of $1/\beta_u$ just gives an estimated bandwidth for link n, and β_u must follow t-distribution with $2m-4$ degrees of freedom. Thus, we first obtain interval k as

$$k = \frac{c\,\sigma_u}{\sqrt{2m-4}}, \tag{16}$$

where c is a 97.5% value of the t-distribution if we want 95% confidence interval. Then we have the confidence interval for the estimated bandwidth $1/\beta_u$ as

$$\frac{1}{(\beta_u + k)} \leq \frac{1}{\beta_u} \leq \frac{1}{(\beta_u - k)}. \tag{17}$$

We have a reliable estimation by adding the confidence intervals as described above. However, it was assumed that measurement errors follow the normal distribution after few very large errors are excluded by the biweight function. That is, when the number of large errors increases, this approach no longer gives a valuable

estimation. In the next subsection, we will present a nonparametric estimation method which does not require any assumption on the error distribution.

Nonparametric Estimation Method

In the nonparametric approach, we do not need any assumption on the error distribution. Let m be the number of obtained measurement data set for each packet size as before. The slope estimation can be obtained by the following procedure.

1. By choosing the every combination of two minimum values of RTTs, and calculate the slope. That is, we have $m(m-1)/2$ slopes by this step.

2. We sort a set of obtained slopes and adopt its median as the proper slope.

A Kendall's τ method [15] is known as one such a way of finding the confidence interval in the nonparametric method. However, it cannot be directly applied to the current problem since it is necessary to calculate the difference of two slopes. Alternatively, we use a Wilcoxon's method [16], which is based on the difference between medians of two data sets **S** and **T**.

In the current context, we use two sets of slopes obtained from the measurements for links $n-1$ and n, which are denoted as **S** and **T**, respectively. By letting the numbers of elements of **S** and **T** be $|S|$ and $|T|$, respectively, we label elements of two sets **S** and **T** as $s(j)$ and $t(i)$ $(1 \leq i \leq |T|, 1 \leq j \leq |S|)$. The bandwidth estimation and its confidence interval are then obtained as follows.

1. Calculate the set of differences $t(i) - s(j)$ $(1 \leq i \leq |T|, 1 \leq j \leq |S|)$. Let us denote the obtained set of the differences as **U**.

2. Sort the set **U** in an ascending order.

3. Let $u(i)(1 \leq i \leq |S| \times |T|)$ denote ith element of sorted set **U**. The confidence interval is then given by

$$u\left(\frac{|T|(2|S|+|T|+1)}{2}+1-a\right) \leq \beta_u \leq u\left(a-\frac{|T|(|T|+1)}{2}\right). \tag{18}$$

   If we want 95% confidence interval, parameter a should be determined such that the probability $P(\sum u(i) \geq a)$ is equal to 0.975. When the numbers of measured data $|S|$ and $|T|$ are large, it is known that $\sum u(i)$

follows the normal distribution with mean $|T|(|S|+|T|+1)/2$ and variance $|S||T|(|S|+|T|+1)/12$. Thus, we can approximate a as;

$$a = \frac{|T|(|S|+|T|+1)}{2} + \frac{1}{2} + 1.96\sqrt{\frac{|S||T|(|S|+|T|+1)}{12}}. \tag{19}$$

We still have a problem in the above procedure. Our final goal is to control the measurement time so that the measurement is finished when the confidence interval of the bandwidth estimation is within a prespecified value. For that purpose, on-line calculation is necessary. However, the above procedure requires much computational time. Suppose that we gather RTT values with 46 kinds of packet sizes as in `pathchar`. The number of slopes obtained for each link becomes 1035, and therefore the number of elements of **U** is beyond 1,000,000. It is too large for the method described above.

We therefore use another method based on a Kendall's rank correlation coefficient [15]. We obtain $m(m-1)/2$ slopes from m trials for each link, and therefore the number of elements $|S|$ and $|T|$ becomes $m(m-1)/2$. We therefore use the following procedure to estimate the confidence intervals.

1. Sort **S** and **T**, and obtain the set **U'**, the element of calculated by

$$u'(i) = s(i) - t(i)(1 \leq i \leq \frac{m(m-1)}{2}). \tag{20}$$

2. The confidence interval of **U'** is then determined by the following equation.

$$u'\left(\frac{\frac{m(m-1)}{2} - C}{2}\right) \leq \beta_{u'} \leq u'\left(\frac{\frac{m(m-1)}{2} + C}{2}\right), \tag{21}$$

where C is the Kendall's rank correlation coefficient. If K is 97.5% value of the standard normal distribution, we obtain 95% confidence interval by using

$$C = K\sqrt{\frac{m(m-1)(2m-5)}{18}}. \tag{22}$$

The on-line calculation procedure and stopping rule for the RTT measurement will be described in Subsection 2.2.3.

2.2.2 Removal of Unnecessary RTT Values

As having been shown in Figure 3, it is necessary to pick up proper RTTs when the distribution of RTT consists of several groups of RTTs. It is caused by the route alternation that `pathchar` can never detect. To divide data

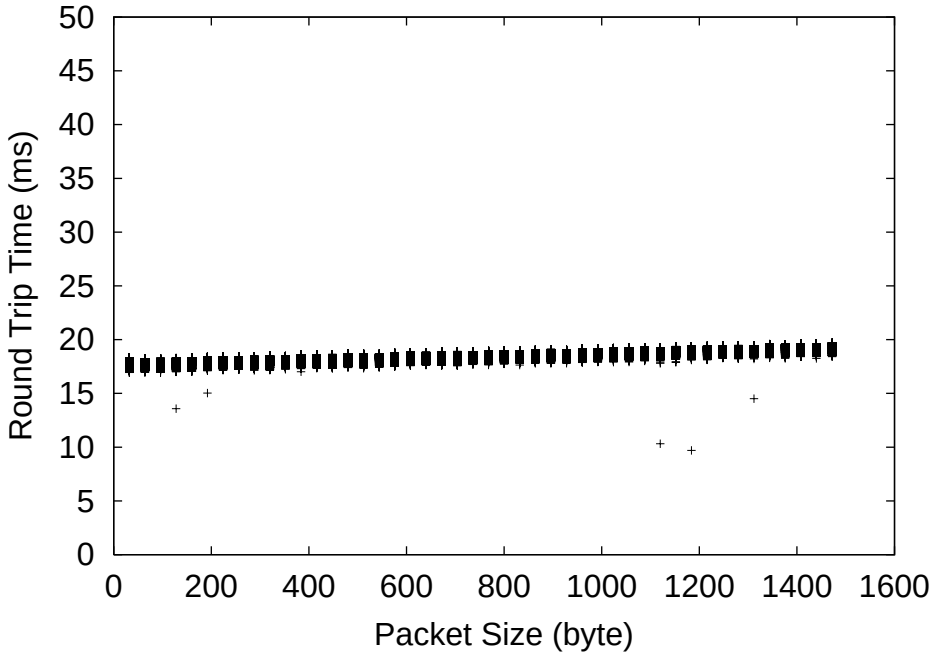

Figure 5: Result of Clustering

into several groups, we use the clustering method [17]. After we obtain the measurement data, we first abandon the upper z% of measured RTTs since those does not help estimating the link bandwidth. In the experiments in Section 4, we will set $z = 30$ which was found to be a reasonable value for our purpose. Then, we divide them into several clusters. We assume that the cluster having the largest number of measured elements contains the actual minimum RTT. If route alternation does not occur during the measurement, it is not necessary to apply the clustering. We can know it if divided clusters are very close with each other. Figure 5 plots the result of the clustering using the data shown in Figure 3. Note that we divided the gathered data into three clusters. The figure shows that we can extract the clusters of RTTs properly. A weak point of this procedure is that it takes much time for clustering and therefore we cannot repeat clustering for every packet arrival. From this reason, we perform clustering after the measurement of RTTs will have finished in the experiment.

2.2.3 An Adaptive Mechanism to Control the Measurement Period

To control the measurement period of the bandwidth estimation, our adaptive mechanism is based on an iterative procedure which the number of probes is progressively increased until the confidential interval of the estimated bandwidth becomes less than the prescribed value. In this mechanism, one problem is how to update the mini-

mum RTTs for each iteration. After measuring additional RTTs, there are two candidates for the minimum RTT; one is the smallest value of additional RTTs, and another is the current minimum RTT. Usually, it is desirable that the minimum RTT is updated by the smaller value of these two candidates. However, this approach sometimes leads to the inaccurate bandwidth estimation when the smallest value of measurements is caused by the exceptional large error or route alternations. Even if our estimation approach can remove such errors, it requires an additional number of iterations which directly leads to increase the processing overhead. From this reason, it is necessary to eliminate the influence of errors as many as possible before the estimation. We then introduce the following update process of the minimum RTT, which we will refer to as *"Minimum RTT Update Procedure"*.

Let the smallest value of measured RTTs and the current minimum RTT be y_i' and y_i, respectively. We first compare y_i' with y_i. If $y_i' \geq y_i$, we keep y_i as the minimum RTT. If not, we next calculate the difference $d' = |y_i' - Y|$ and $d = |y_i - Y|$, where Y is the estimated value of RTT on the line (e.g., $Y = \beta x_i + \alpha$ from the given packet size x_i). By comparing d' with d, we replace the minimum RTT with y_i' when the difference d' is smaller than d or 30% larger than d. Without this approach, lots of accurate minimum RTTs are discarded and then many additional probe packets are required to get the same accuracy as result of latter approach.

More specifically, the following procedure is performed during the RTT measurement. In describing the procedure below, we suppose that the bandwidth estimation for link $(n-1)$ has already been finished.

1. For estimating the bandwidth of link n, we first send a fixed number of packets. For example, we send 10 packets in our experiments presented in the next section. Then, RTTs are collected for 46 kinds of the packet size (from 40 bytes to 1,500 bytes). Namely, the source sends $10 \times 46 = 460$ packets in the initial measurement.

2. For taking account of route alternation, we check the source address of the ICMP packets as in `pathchar`. We take router n, the address of which appears most in the ICMP packets.

3. We then estimate the initial value of bandwidth and its confidence interval of nth link by either our parametric or nonparametric methods; (see Subsection 2.2.1).

4. To get the accurate bandwidth estimation and confidence interval, we iterate following procedure.

   (a) We send an additional set of probes (e.g., 10 packets for each packet size) to get new RTTs for router

n.

(b) We update the minimum RTT by using the *Minimum RTT Update Procedure*.

(c) Because the accuracy of the bandwidth estimation for link n depends not only on the precision in the slope estimation of link n, but also on the one of link $(n-1)$, we need to send additional packets to router $(n-1)$ when the source sends more packets to router n. Note that these additional measurements are not necessary for the bandwidth estimation for link $(n-1)$.

(d) By using our estimation approach (described in Subsection 3.1), we update the estimated bandwidth and its confidential interval. The iteration terminates if the confidence interval of the estimated bandwidth becomes less than the prescribed value (e.g., 10% of the mean value).

5. After the iteration terminates, we finally verify whether RTTs have reasonable values. A most important task at this step is to apply the clustering technique. If the measurement is correct, the cluster having the largest number of elements must contain the measured minimum RTTs. If RTTs are not proper because of the route alternation, we retry the measurement process by going back to Step 4.

2.3 Experimental Results and Discussions

2.3.1 Removing Irregular RTT Values due to Exceptionally Large Errors

We first show experimental results for the case where RTT values apparently do not follow the normal distribution because of some large errors. The example was shown in Figure 2. Figure 6 plots only minimum values of RTTs against the packet size from Figure 2. As shown in this figure, the variation of minimum RTTs exhibits far from the linear relation. Figure 7 compares results of the slope estimations by `pathchar`, `pchar`, and our methods (the M-estimation and nonparametric methods). Straight lines of `pathchar` and `pchar` are inaccurate due to exceptionally large errors whose packet sizes are 288, 960, 1376, and 1440 bytes. On the other hand, our approach can filter out such errors.

Table 1 shows the estimated values. In the table, two cases of the bandwidth estimation are shown; 13-th link from 202.231.198.2 destined for 210.142.124.1 (corresponding to Figure 2) and 13-th link from 202.232.8.66 destined for 210.141.224.162. The capacities of those links were known a priori as 1.5 Mbps and 45 Mbps. For each of two links, we show the estimated bandwidth obtained by all methods. Confidence intervals of 95% are

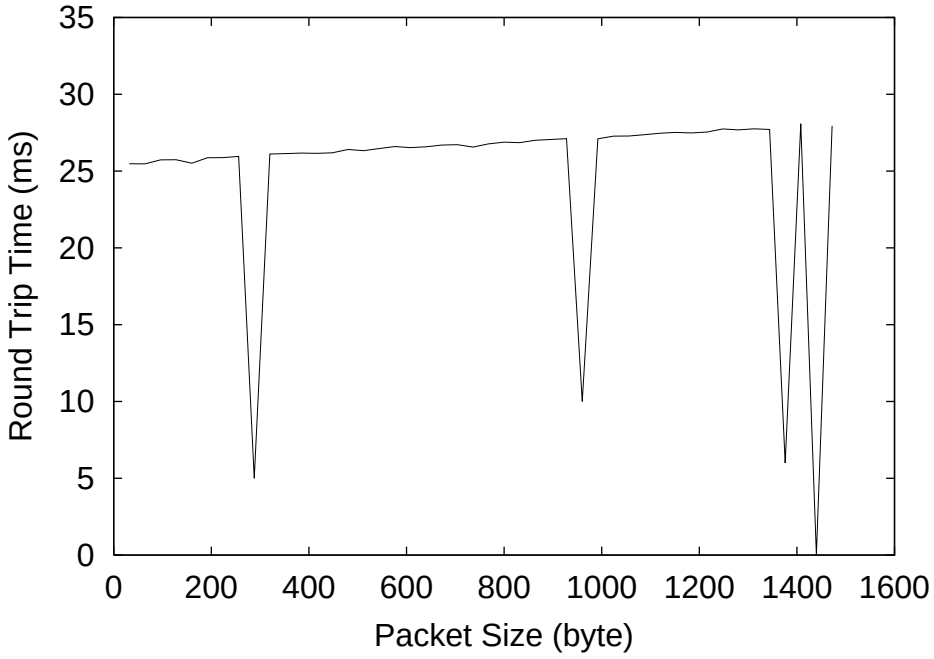

Figure 6: Minimum RTTs Including Irregular Values

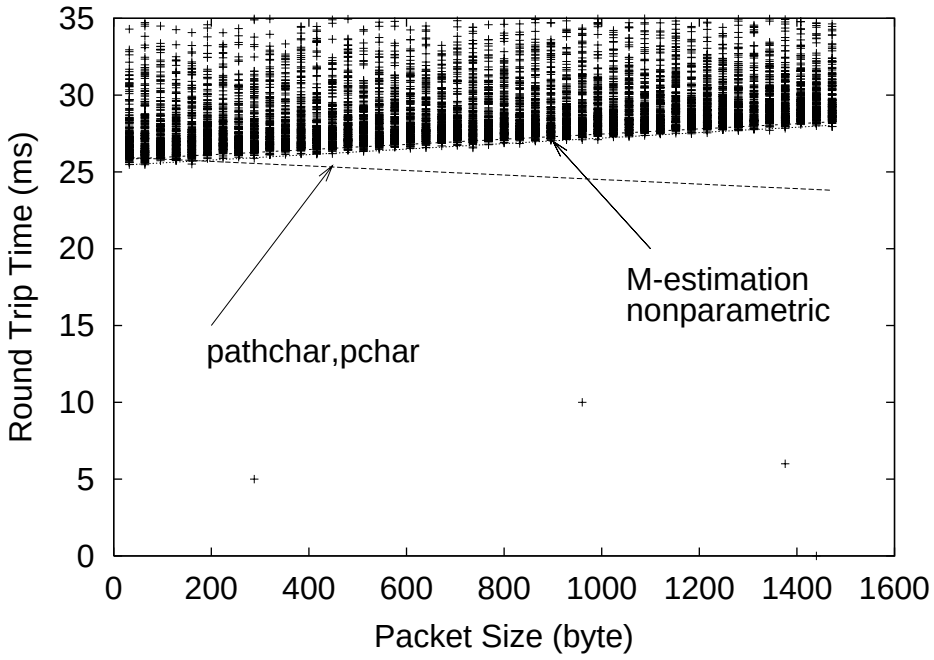

Figure 7: Estimated Lines Including Irregular Values

Table 1: Bandwidth Estimation and Confidence Intervals for the Measurement data with Irregular Values

bandwidth	method	estimated results (Mbps)	the # of probe packet
1.5 Mbps	Pathchar	0.87	200
	M-estimation	$1.34 \leq 1.35 \leq 1.36$	10
	Wilcoxon	$1.32 \leq 1.33 \leq 1.36$	20
	Kendall	$1.30 \leq 1.33 \leq 1.37$	20
45 Mbps	Pathchar	86.65	200
	M-estimation	$44.06 \leq 46.58 \leq 49.42$	200
	Wilcoxon	$42.99 \leq 53.44 \leq 66.54$	200
	Kendall	$52.22 \leq 53.44 \leq 54.69$	200

also shown in our methods. As shown in the table, results obtained by `pathchar` and `pchar` are far from the actual bandwidth, while our methods can give very close values. The difference of the actual bandwidth and the estimated bandwidth is due to the overhead of the underlying network. In the table, the numbers of packets transmitted for each packet size are also shown. In our methods, the very small number of packets were sufficient to obtain the accurate results for 1.5 Mbps link. For 45 Mbps link, on the other hand, 200 packets were necessary, which is same as `pathchar`. It is due to the fact that as the link bandwidth becomes large, the accurate estimation becomes difficult, which has already been pointed out in [7].

In the case of 1.5 Mbps link, we cannot observe differences among our three methods, the M-estimation, Wilcoxon's and Kendall's methods. In the case of 45 Mbps link, the M-estimation method seems to be best. However we cannot decide the best one here because we found many cases that the other method gives the best result, as will be presented in the below.

2.3.2 Clustering RTT Values against Route Alternation

If the distribution of RTT values consists of several groups due to route alternation, it is apparent that the approach to cut off the exceptionally large errors mentioned above is not sufficient. See Figure 8, where we plot minimum values of RTTs against the packet size. The RTT values fluctuate to a large extent. Of course, it misleads us about the estimation, and the estimation obtained by `pathchar` and `pchar` are meaningless. Then,

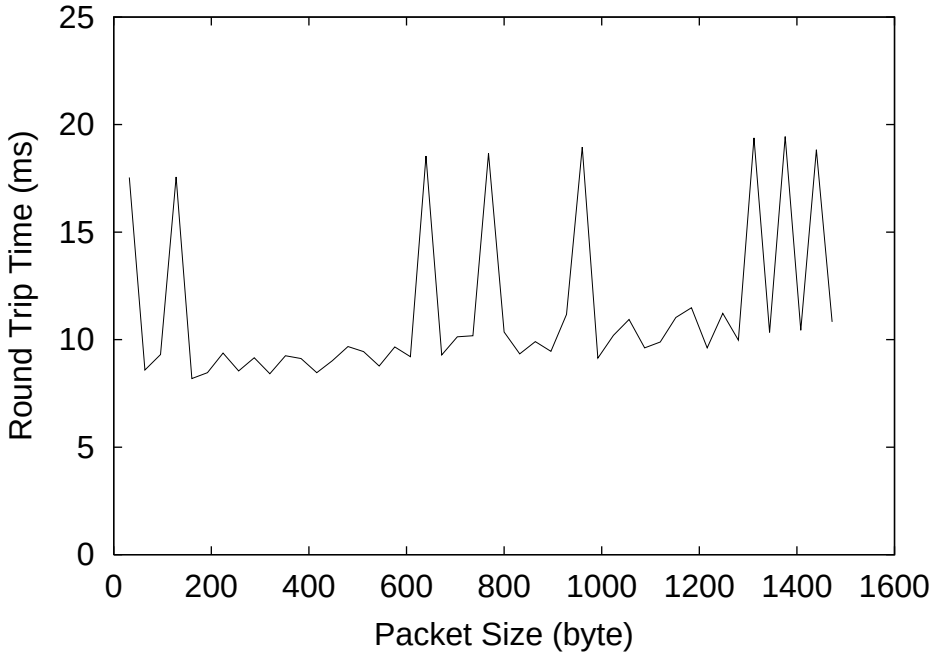

Figure 8: Minimum RTT for Mixed Groups

our clustering approach presented in Subsection 2.2.2 becomes necessary to exclude the RTT values obtained by an *exceptional* route.

Table 2 shows the estimation results. In the table, the cases of 10 Mbps and 12 Mbps links are shown. Those are located at 8-th link from 150.100.59.2 towards 202.219.160.22 and 15-th link from 210.157.131.158 towards 210.224.236.1. The numbers of packets transmitted in each method are also shown in the table. From this table, our estimations show the reasonable values while `pathchar` and `pchar` lead to even a negative value. The reason becomes clear when we look at the slope estimation plotted in Figure 9. Since the estimated values of `pathchar` and `pchar` is small, the resultant estimation on the bandwidth of the target link takes a negative value. On the other hand, our methods can estimate the slope adequately.

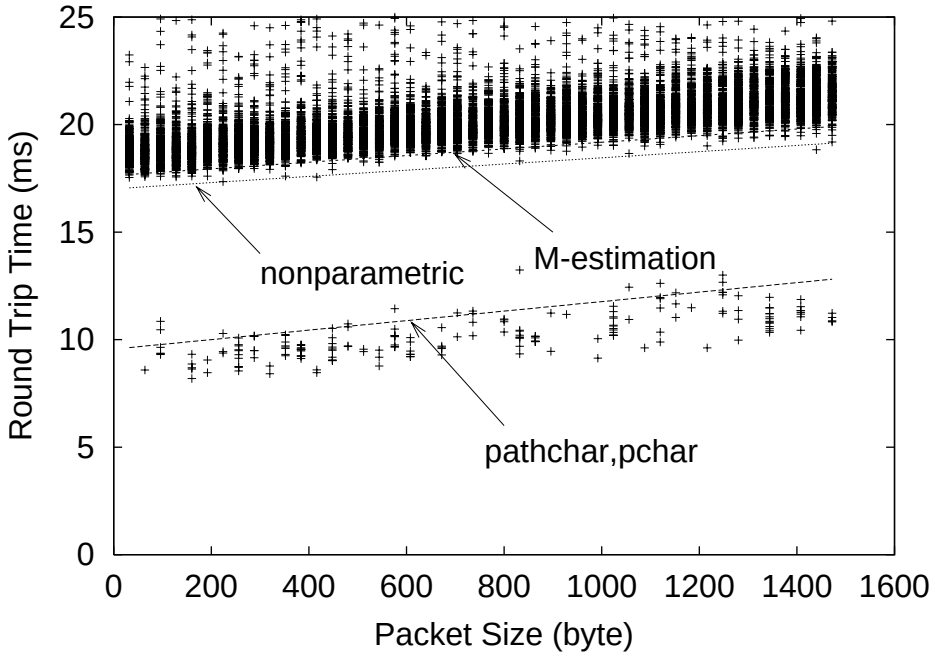

Figure 9: Estimated Slopes in the case of Mixed RTT Values

Table 2: Bandwidth Estimation and Confidence Intervals for the case of Mixed RTTs

bandwidth	method	estimated results (Mbps)	the # of probe packet
10 Mbps	Pathchar	-22.6	200
	M-estimation	$10.07 \leq 12.40 \leq 16.11$	200
	Wilcoxon	$16.59 \leq 16.95 \leq 24.07$	200
	Kendall	$14.24 \leq 16.95 \leq 25.29$	200
12 Mbps	Pathchar	8.25	200
	M-estimation	$9.79 \leq 9.94 \leq 10.09$	20
	Wilcoxon	$13.3 \leq 13.8 \leq 14.4$	90
	Kendall	$13.6 \leq 13.8 \leq 14.1$	90

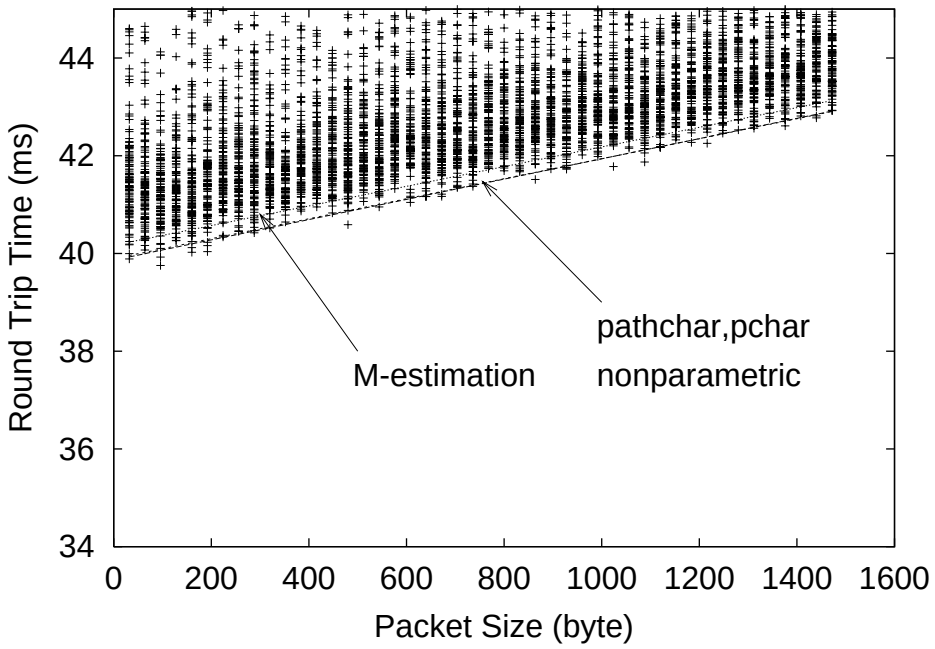

Figure 10: RTT Values and Estimated Slopes

2.3.3 Controlling the Measurement Period Adaptively

We next show how our adaptive control of the measurement period works. Differently from previous cases, we pick up the cases where `pathchar` can also show the reasonable results in this subsection. Figure 10 compares estimated slopes of minimum RTTs among `pathchar`, `pchar` and our methods. As shown in the figure, there is no remarkable difference among all estimation methods. Table 3 also shows the same tendency; estimated values of link bandwidths are quite close with each other. These results suggest that the error contained in the minimum values of RTT can well be modeled by a normal distribution in usual cases if the amount of measurement data is sufficiently large.

However, our estimation approaches have two advantages over `pathchar` (and `pchar`). First, our method can control the number of probes adaptively. As shown in Table 3, the measurement terminates with a less number of probes in our method except the case of 6 Mbps link. Table 4 summarizes the required number of probes to obtain the 95% confidence intervals where minimum and maximum values are within 5% difference from the mean value. Note that symbol '*' in the table shows that the result does not reach within the prescribed confidence interval by that number of probes. For several links, the number of probes for each packet size is less than 200. On the other hand, the number of transmitted probes by `pathchar` was always 200; it implies that

Table 3: Bandwidth Estimations and Confidence Intervals by Measurement Period Control

bandwidth	method	estimated results (Mbps)	the # of probe packet
6 Mbps	Pathchar	5.75	200
	M-estimation	$6.48 \leq 6.60 \leq 6.72$	200
	Wilcoxon	$5.65 \leq 5.87 \leq 5.92$	200
	Kendall	$5.67 \leq 5.87 \leq 5.94$	200
1.5 Mbps	Pathchar	1.46	200
	M-estimation	$1.37 \leq 1.40 \leq 1.43$	20
	Wilcoxon	$1.43 \leq 1.45 \leq 1.47$	110
	Kendall	$1.42 \leq 1.45 \leq 1.48$	110
12 Mbps	Pathchar	10.6	200
	M-estimation	$- \leq 10.5 \leq -$	200
	Wilcoxon	$10.40 \leq 11.34 \leq 12.28$	50
	Kendall	$11.04 \leq 11.34 \leq 11.59$	50

Table 4: Variations on the Required Number of Probes per Packet Size

bandwidth	link location	M-estimation	Nonparametric
10 Mbps	10 th	10	630
10 Mbps	12 th	*1127	220
12 Mbps	15 th	10	10
12 Mbps	12 th	30	80
45 Mbps	13 th	370	*1007
100 Mbps	16 th	427	979
100 Mbps	9 th	*1080	*1080

`pathchar` wastes the network bandwidth by unnecessarily transmitting packets. For other cases, the numbers of probes are larger than `pathchar`, but we can expect that the resultant estimated values become more reliable than the values obtained by `pathchar`.

A second advantage of our methods is that we can obtain unified degrees of confidence on all links. On the other hand, the accuracy of estimation by `pathchar` is varied, and more importantly, there is no means to know about reliability on the estimated values.

Between parametric and nonparametric approaches, the required number of probes by the nonparametric approach is larger than that of the parametric approach. It is natural since the nonparametric approach does not assume any distribution on errors. Then, it needs a larger number of probes for reliable estimation. The large number of probes was necessary for the second link in the table in spite of 10 Mbps link. It is because the utilization of that link was high. It verifies that our method can adaptively increase the number of probes according to the link congestion.

2.3.4 On-line Estimation of Confidence Intervals

We last discuss on the derivation methods of confidence intervals in our methods. As having been described in Section 2.2.1, the method based on Kendall's rank correlation coefficient is approximate in obtaining the confidence interval, and it must be less accurate than the one based on Wilcoxon's method. However, differences between Kendall's and Wilcoxon's methods were within 5% of the link bandwidth as having been shown in Tables 1, 2, and 3. If we collect m kinds of the packet size, the calculation time by Wilcoxon's method becomes $O(m^4)$, while $O(m^2)$ in Kendall's method. Therefore, Kendall's method is useful for the on-line estimation of confidence intervals.

In our experiments, the M-estimation method sometimes failed to determine the confidence interval, which was shown in the last example of Table 3. It is caused by assuming that the variance of slopes σ_n^2 for link n is larger than σ_{n-1}^2 for link $(n-1)$. See Eq. (15). That assumption is valid if we can measure RTTs of routers $n-1$ and n by the same packet. However, because it is impossible, RTTs of routers $n-1$ and n must be measured separately, and the above assumption does not hold.

As having been presented in the tables, the assumption that the measurement errors follow the normal dis-

tribution seems to be often valid. However, it can only be known examining the links, bandwidth of which is a priori known.

3 Measurement Based Identification Method of Performance Bottlenecks

In the previous section, we have investigated on the estimation method of link capacities along the end-to-end path. However, there exist many factors limiting the end-to-end performance. For network dimensioning, we first need to determine which factor actually restricts the current performance. If the link capacity is found to be a limiting factor of the performance, then it be increased. In this section, we propose a new bottleneck identification method based on the measurement of network characteristics. Our proposal on the network dimensioning is then described in the next section.

3.1 Framework for Bottleneck Identification

Figure 11 shows our framework of a measurement-based bottleneck identification method. As shown in the figure, we consider the TCP connection through which the user receives data (e.g., Web documents) from the destination host (server) to the measurement host (client). Then, we want to identify what is a bottleneck in the current network configuration, and what restricts the end-to-end performance. The main goal of our study is to find the limiting factor on the end-to-end performance, and our identification process is intended to be applied to each TCP connection.

There exist many possible causes of limiting the TCP performance, but we can classify those in the following

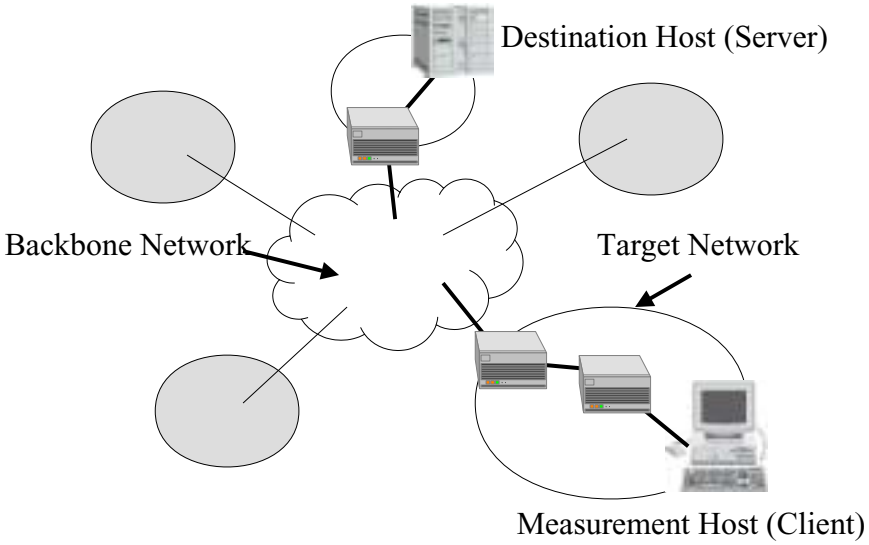

Figure 11: Measurement Environment

three categories.

1. **Network Configurations:**

   The capacity of one or more links is quite small than the one that the end users expect. A simple example may be found in the case where the customer is connected by a telephony line. The throughput is limited to the speed of a modem (e.g., 33.6 Kbps or 56 Kbps). The second and most important reason for performance limitation is due to the link congestion. It leads to many packet losses, and TCP throughput decreases significantly. To resolve it, it is necessary to increase the capacity of the bottleneck link, which is the main target of our bottleneck identification method.

   However, even if the throughput is much lower than the one expected by the user, the performance may be limited by other reasons. In such a case, the capacity increase does not help performance improvement. Those reasons are due to the poor performance of the end hosts, which will be described in the below.

2. **Receiver–Side Configurations:**

   In order to avoid mis-ordering of the packet sequence, the receiver host prepares a buffer to store the received packets. In the TCP connection, the required size of buffer is basically given by the *Bandwidth–Delay Product*, which is determined from Available Bandwidth $\times$ RTT. The buffer size of the receiver host is notified to the sender via the *Advertised Window* field in the TCP header. The buffer size sometimes becomes a bottleneck in the environment with the large *Bandwidth–Delay Product* (e.g., the network offering high-speed link(s) and/or the large RTT). Once it is found to be a bottleneck, it can be solved by a window scale option of TCP [18] after preparing the additional buffer. However, too large window leads to the bursty packet emission due to the rapid growth of the window size, which may result in the frequent packet losses. Thus, a 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 a transmission rate of each TCP connection, in order to share the resources of the access network and the end hosts fairly among clients. If such a rate control is adopted as the service policy, there is no means to improve the performance of individual users.

Another cause of the server–side bottleneck is due to the packet processing overhead at the sender host. For example, if we connect to the 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 solved 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.

3.2 Bottleneck Identification Method

3.2.1 Measurement Parameters for Bottleneck Identifications

Our target of the bottleneck identification process is TCP connections. We use the following equations that characterize the throughput of the TCP flow [19].

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

$$E[W] = \frac{2+b}{3b} + \sqrt{\frac{8(1-p)}{3bp} + \left(\frac{2+b}{3b}\right)^2}, \tag{23}$$

where p and b are the packet loss rate and the number of arrival packets notified by one acknowledgement (ACK) packet, respectively. In the original version of TCP, the receiver should send an ACK packet for each receipt of the packet. Authors in [20], however, have shown that the overall network load can be reduced without any performance degradation by applying a *delayed ACK*, in which the receiver sends an ACK packet for the acknowledgement of multiple (two or more) received packets. The mechanism of the delayed ACK is widely used in recent TCP implementations, and its parameter b is typically set to be 2. Note that if $p = 0$, $E[W]$ cannot be estimated. However, when the link bandwidth is a bottleneck, packet losses are likely to occur because of an inherent nature of the TCP mechanism; the congestion control mechanism of TCP decreases the window size (i.e., the transmission rate) by detecting packet losses. If the packet loss never occurs, there is no problem in the current network configuration.

In the above equation, we need to know the packet loss probability along the path. As described before, the end host cannot detect the packet loss directly. One approach is to introduce the upper layer protocol for

exchanging such an information. Such an approach can be found in RTP (Real Time Protocol) and RTCP (Real Time Control Protocol) [21]. However, it limits the applicability of our approach because it requires a special-purpose software at server and client hosts. Another approach is to utilize the active measurement tool such as `pchar`. It requires the additional traffic measurement. Another simpler approach that 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 ordinally network usage, no additional resource consumption is introduced.

Once we have an estimated value of $E[W]$, we next check whether the receiver buffer is sufficient or not by comparing the buffer size W_{max} and the expected window size $E[W]$. In [18], TCP throughput is also given:

$$\begin{aligned} B(p) &= \frac{\frac{1-p}{p} + E[W] + \hat{Q}(E[W])\frac{1}{1-p}}{RTT(\frac{b}{2}E[W]+1) + \hat{Q}(E[W])T_0\frac{f(p)}{1-p}}, && \text{if } E[W] < W_{max}, \\ &= \frac{\frac{1-p}{p} + W_{max} + \hat{Q}(W_{max})\frac{1}{1-p}}{RTT(\frac{b}{8}W_{max} + \frac{1-p}{pW_{max}} + 2) + \hat{Q}(W_{max})T_0\frac{f(p)}{1-p}}, && \text{otherwise}, \end{aligned} \tag{24}$$

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

$$\hat{Q}(w) = \min\left(1, \frac{(1-(1-p)^3)(1+(1-p)^3(1-(1-p)^{w-3}))}{1-(1-p)^w}\right), \tag{25}$$

and

$$f(p) = 1 + p + 2p^2 + 4p^3 + 8p^4 + 16p^5 + 32p^6. \tag{26}$$

If the receiver buffer size is a cause of the bottleneck (found by Eq. (24)), TCP throughput is approximately given as

$$B(p) \approx \frac{W_{max}}{RTT}. \tag{27}$$

T_0 is obtained from the retransmission timeout value (RTO) which is necessary to detect the packet at the end host. Since IP, the underlying protocol of TCP, is connectionless, the TCP sender is not directly notified of the packet loss from the network, and therefore TCP uses an acknowledgement–based protocol to know the packet losses. In TCP, there are two methods to detect the packet loss. The first one is that the sender host decides a packet loss in case where the sender does not receive its ACK packet within the timeout threshold. This threshold is called RTO, and it is calculated by the following equations:

$$RTO = RTT_m + 4D \tag{28}$$

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

Note that the RTO value is not used in a usual case for 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 method to detect the packet loss is performed by checking the sequence number of the acknowledgements. When the packet is lost within the network, the sender host receives the ACK packet with the same acknowledgement number as the previous one. Those ACKs are called as *Duplicate ACK*s. However, the sender host cannot determine whether a duplicate ACK is caused by a lost segment or just a mis-ordering of packets, and some "margin" is necessary to ignore duplicate ACKs. Actually, when three or more duplicate ACKs (called as *Triple Duplicate ACKs*) have been received, the sender host recognizes that the packet has been lost. The sender then retransmits the packet and sets the congestion window size to be half.

To identify the bottleneck based on the above equations, we need to obtain (1) packet loss rate p, (2) maximum window size at the receiver host W_{max}, (3) number of packets notified by one ACK packet b, (4) a round trip time RTT, and (5) a packet retransmission timeout T_0. In addition, we need to measure the TCP throughput, because the above prediction formula of TCP throughput is occasionally incorrect. The packet loss probability can be determined from the method described above. Since W_{max} and b are the configurable parameters, those 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 active measurement tools such as `ping`. After we collect a set of RTTs, we can evaluate the RTO value from Eq. (29) one by one, and get the mean value of RTOs. We finally calculate the measured TCP throughput by observing traced TCP headers and counting the total bytes transmitted. The physical capacity of the bottleneck link is helpful for the bottleneck identification, which can be obtained by our proposed method described in the previous section.

3.2.2 Bottleneck Identification Method

In this subsection, we propose the bottleneck identification method. According to the identification described in Subsection 3.1, we examine the receiver–side, sender–side, and the network configuration in turn to determine the bottleneck.

1. Measure the network parameters, i.e., RTT, RTO, packet loss rate, TCP throughput, and physical capacity of bottleneck link.

2. Change the receiver buffer size if it is a current cause of the bottleneck, and again measure the TCP throughput. If the throughput is not improved, the bottleneck is likely to exist at 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. (24) do not consider the sender–side bottlenecks.

3. Check whether the receiver buffer size is sufficient or not. It can be verified by Eq. (23). If the buffer size is insufficient, increase it. Otherwise the current cause of the limited performance exists in the network. We can confirm it by actually increasing the receiver buffer size. When the bottleneck is the link bandwidth, it simply results in that the packet loss rate and RTT is increased.

4. When the receiver buffer size is not sufficient, the receiver buffer size should be increased so that desired performance can be obtained. However, it should be performed carefully. As shown in Eq. (27), the receiver buffer size and the resultant TCP throughput has a linear relation if the receiver buffer size is a cause of the bottleneck. Anyway, as the receiver buffer size is increased, the bottleneck may shift to another part.

3.3 Experimental Results

In this section, we show the experimental results based on our bottleneck identification method. In our experiments, we use a `ping` program to obtain the round trip times (RTTs), and `tcpdump` [22] for capturing TCP headers. Furthermore, we adopt our bandwidth prediction method presented in Section 2 to obtain the physical link capacity of each links.

As shown in Figure 11, we placed the receiver host (client) at Osaka University, and choose five hosts (Osaka

Table 5: List of Sender Hosts

abbreviation 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`

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`)) as the sender host (server). Table 5 shows the list of sender hosts.

Table 6 summarizes our measurement and bottleneck identification results. Each row in the table consists of nine fields; the sender host, the bandwidth of the bottleneck link, the receiver buffer size, the round trip time, the packet loss rate, measured TCP throughput, estimated TCP throughput. The eighth field ("buffer") shows whether the buffer size is sufficient or not, which was determined according to Eq. (23). The last column of the table is the result of bottleneck identification; the bottleneck is due to 1) network configuration, 2) sender–side configuration, or 3) receiver–side configuration.

In the `sonet` case, the receiver buffer size was the bottleneck since 1) the RTT value is small, 2) the receiver buffer size is evaluated as "insufficient," and 3) the socket buffer size is in proportion to the measured TCP throughput. The same results were obtained in `ocu`, `iij`, and `iamas` cases when the socket buffer size is set to be 8 KB.

The `ocu` case shows a different result; the bottleneck is moved from the buffer size to the link bandwidth when changing the buffer size from 8 KB to 64 KB. The total number of lost packets is shown in Figure 12, where the receiver buffer size is set to be 8 KB and 64 KB. 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 is quite small in the 8 KB buffer case, which means that the bottleneck is not due to the network configuration. Furthermore, the number of lost packets is increased in proportion to time, which is a typical observation when

Table 6: Results on Measurement 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	

the bottleneck is the link bandwidth.

In the `iij` case, it is expected that the maximum TCP throughput is 30 Mbps from the measurement of the bottleneck link bandwidth (see the second column). It is verified by the estimated TCP throughput (the seventh column). However, the maximum throughput by the measurement was about 4.8 Mbps. If throughput is limited by the network congestion, it is expected that the packet loss rate should be high. However, there is no changes in the packet loss rate contrary to `ocu`. Such results are also found in `ryu` and `iamas`. These cases are classified into the case that the sender–side configuration limits the performance. We have two possible causes for this result.

- **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

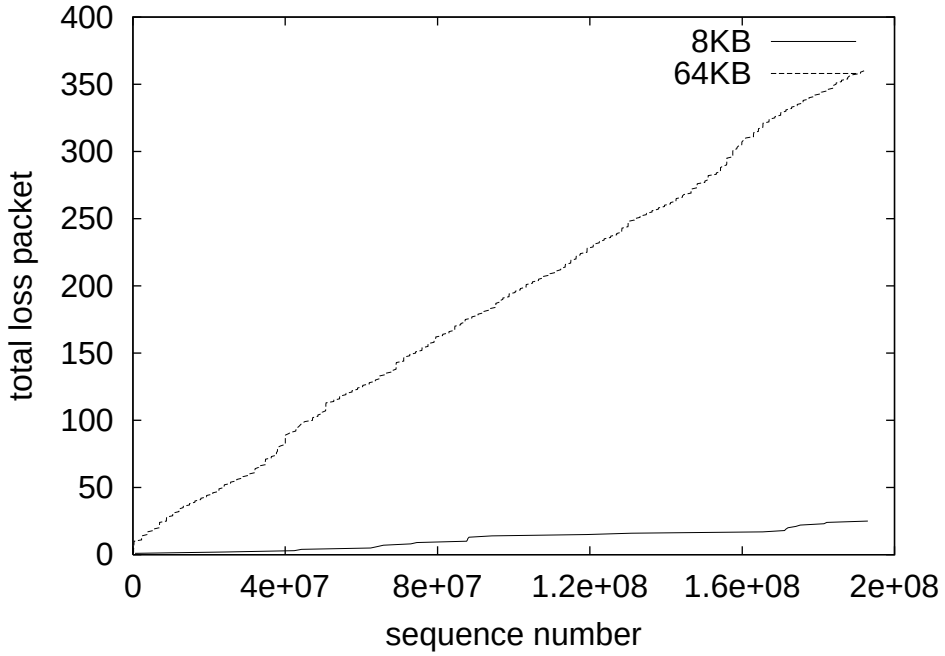

Figure 12: Osaka City University: The Total Number of Lost Packets

sending rate becomes equal to or slower than the bandwidth of the first link. Accordingly, rest of packets are waiting for forwarding at the input buffer. Contrary to intermediate routers, the input buffer of the first link is seldom overflowed, since the sender can control its sending rate based on the available bandwidth for the first link. Imagine the 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 still small, in spite that the link bandwidth is the bottleneck.

- **Available bandwidth limitation caused by a server:**

  Recently, some operating systems support the mechanism for the ftp server to limit the sending rate of each user. Those include Linux, FreeBSD, and WindowsNT. If such a restriction is used, the server does not send packets at the rate larger than the predetermined bandwidth. Then, the packet loss rate and RTTs are kept low values because the packets are seldom queued within the networks.

In order to discriminate the above two causes, it is necessary to measure the link bandwidth of the first link from the server. When the sender host is located within the target network, these two bottlenecks could be easily distinguished to acquire information such as a link bandwidth and an existence of available bandwidth limitations. If the server is located outside the target network, on the other hand, it would be difficult to discriminate

Table 7: Confidence Intervals

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

these two causes clearly. However, if we consider the capacity dimensioning on the target network as just in the current case, this identification is no longer meaningful.

Finally, in order to investigate the accuracy of our measurement results, we present a confidence interval of the estimated TCP throughput in Table 7. We can observe that the width of all confidence intervals is very narrow, which indicates that the sufficient quantity of data was collected to put a confidence on results.

4 Capacity Dimensioning Based on Traffic Measurements

In this section, we will explain how our capacity dimensioning is performed. It is based on traffic measurements described in previous sections. The aim of our capacity dimensioning method is to satisfy QoS from the end user's point of view. The target QoS metric that we will consider in this section is TCP throughput. We have shown in Section 3 that the link capacity is not an only cause of the performance bottleneck; increasing the link bandwidth does not necessarily lead to the performance improvement for the end users. Instead, the sender or receiver side configurations might limit the performance. Thus, we have to first identify the bottleneck via the statistical characterization of the current network and server configurations.

Our goal of the network dimensioning is to increase the end-user's throughput from t to $t' = t + \Delta t$. For that purpose, we want to determine the next action. Throughout this section, we will call the TCP connection for improving its throughput from t to t' as a *target connection*. Its difficulty is due to the fact that increasing the bandwidth by Δt does not necessarily lead to achieve our objective. That is, the additional capacity may not be occupied by the target user willing to increase his throughput, since the other users may also enjoy the increased capacity due to the capability of bandwidth sharing of TCP.

In determining the link capacity, it is better to have the future traffic demand on the basis of the current traffic. However, since it is is very difficult and probably impossible, we have to realize the target throughput by repeating the process mentioned in the below. We first explain the process of the capacity dimensioning in the case where the bottleneck is not the link bandwidth in Subsection 4.1. In Subsection 4.2, the issue of capacity planning is discussed for the case where the link capacity is actually the bottleneck.

4.1 Cases of Bottlenecks in Sender or Receiver Side Configuration

We consider the first case that the link bandwidth is not a bottleneck. Even if the link capacity along the communication path is increased, the throughput improvement cannot be expected since the bottleneck exists in the other part of the end-to-end communication path. When the sender-side configuration is the bottleneck, the throughput is increased by e.g., eliminating the rate control or upgrading the processor power at the servers, as we have discussed in the previous section. On the other hand, if the socket buffer size of the receiver is the bottleneck, the user can increase its throughput by preparing a sufficient amount of the buffer size.

If the buffer size f at the receiver side is the bottleneck, the TCP throughput is given by [19]

$$B = \frac{f}{RTT}. \tag{29}$$

Accordingly, in order to obtain the throughput t', we prepare the buffer size fulfilling the following inequality.

$$f' \geq t' RTT. \tag{30}$$

If the buffer size is too large, however, packets might be sent at a higher rate than the bottleneck link bandwidth, and it would result in throughput degradation. It is because the larger buffer incurs the faster fluctuation of the congestion window size of TCP, and leads to the frequent packet loss occurrences. In order to avoid this, it is necessary to set an appropriate buffer size based on Eq. (30).

Increasing the buffer size to f' solely is sometimes insufficient since the bottleneck may be moved to the link bandwidth. Let us suppose that the buffer size is actually the bottleneck. Let C be the bandwidth of the link that has the smallest value of the utilization (ρ) along the end-to-end path. ρ is not large since it is not the current bottleneck. Furthermore, let $A = \rho C - t$ denote the bandwidth used by the background traffic (i.e., connections except the target connection). We can then obtain the available bandwidth of the link by $(1-\rho)C$. By increasing the buffer size to f', the throughput is expected to increase by Δt. Of course, if $\Delta t > (1-\rho)C$, the target throughput cannot be achieved. In such a case, we next have to consider the increase of the link capacity.

Because the routing mechanism of IP 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. Furthermore, we can also assume that the total throughput of the background traffic keeps A because the link was underutilized before buffer resizing. Then, we have the updated link capacity C' from

$$C' \geq A + t + \Delta t. \tag{31}$$

4.2 Cases of Bottlenecks in Network Conditions

We next discuss the case in which the link bandwidth is the bottleneck. The bottleneck link must be almost fully utilized in this case, i.e., ρ nearly reaches 1. Then, the packet loss of TCP connections occurs frequently. In this case, we can consider the following two situations;

1. The bottleneck is caused by the target connection only.

2. The bottleneck is caused by multiple TCP connections (including the target connection).

If the bottleneck is caused by only the target TCP connection, the additional bandwidth will be occupied by the target connection. Thus we can determine the required capacity of the bottleneck link from (31).

If the bottleneck is caused by multiple TCP connections, on the other hand, a careful consideration is necessary: even if the link capacity is increased, we cannot obtain the target throughput because the amount of the cross traffic is also increased. In what follows, we discuss how to attack this problem. For this purpose, we assume that the bottleneck link is simply assumed by an M/M/1 queueing system, while our framework below is not limited to it.

Let L denote the average size of the packets passing through the bottleneck link. The packet arrival rate is then defined by $\lambda = (A + t)/L$, and the service rate is given by $\mu = C/L$. The utilization of the link capacity ρ can be written as

$$\rho = \frac{\lambda}{\mu} = \frac{A + t}{C}. \tag{32}$$

Therefore, we get the packet service time T as

$$T = \frac{1}{\mu}\frac{1}{1 - \rho}. \tag{33}$$

Increasing the bottleneck link capacity by α times leads to

$$\mu' = \frac{\alpha C}{L} = \alpha\mu. \tag{34}$$

Here, we assume that the amount of the cross traffic is still large, and the utilization of the bottleneck link is unchanged. Using Eqs. (33) and (34), the new packet service time T' is obtained as

$$T' = \frac{1}{\mu'}\frac{1}{1 - \rho} = \frac{T}{\alpha}. \tag{35}$$

Note that the service time is decreased after the link bandwidth has been upgraded, and then the RTT of each connection is also decreased.

In [19], TCP throughput is modeled as

$$t = \frac{1}{RTT}\sqrt{\frac{3}{2bp}}. \tag{36}$$

By letting n be the number of intermediate routers in the round-trip end-to-end path, the RTT value is given by

$$RTT = \sum_{i=1}^{2n} T_i + S + \sum_{i=1}^{2n+2} D_i, \tag{37}$$

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. We can get RTT after increasing link bandwidth by

$$RTT' = T' + \sum_{i=1}^{2n} T_i - T + S + \sum_{i=1}^{2n+2} D_i = T' + T_o, \tag{38}$$

where

$$T_o = \sum_{i=1}^{2n} T_i - T + S + \sum_{i=1}^{2n+2} D_i \tag{39}$$

After increasing the link capacity when the link bandwidth is 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':

$$t' \leq \frac{1}{RTT'}\sqrt{\frac{3}{2bp}}. \tag{40}$$

Using Eqs. (35), (38), and (40), we can determine the new link capacity as

$$\alpha \geq \frac{T}{\frac{1}{t'}\sqrt{\frac{3}{2bp}} - T_o}. \tag{41}$$

Noticeably, Eq. (41) 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 user can occupy the large portion of the bottleneck link bandwidth by reducing the service time from T to T'. In this case, the increase rate of cross traffic is smaller than that of the user's traffic, and therefore, we set the capacity of the bottleneck link at C' satisfying

$$C' \geq \frac{t'}{t}C. \tag{42}$$

The above equation can be derived by setting $T_o = 0$ in Eq. (41).

We next consider the case where the sender host is located far from the receiver host and RTT is very large. In such a case, 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 to re-design the link capacity until the bottlenecks of the background connections change from the current bottleneck link to another link. It is hard to estimate the above capacity directly, but we can repeatedly get rid of the bottleneck based on our proposed framework.

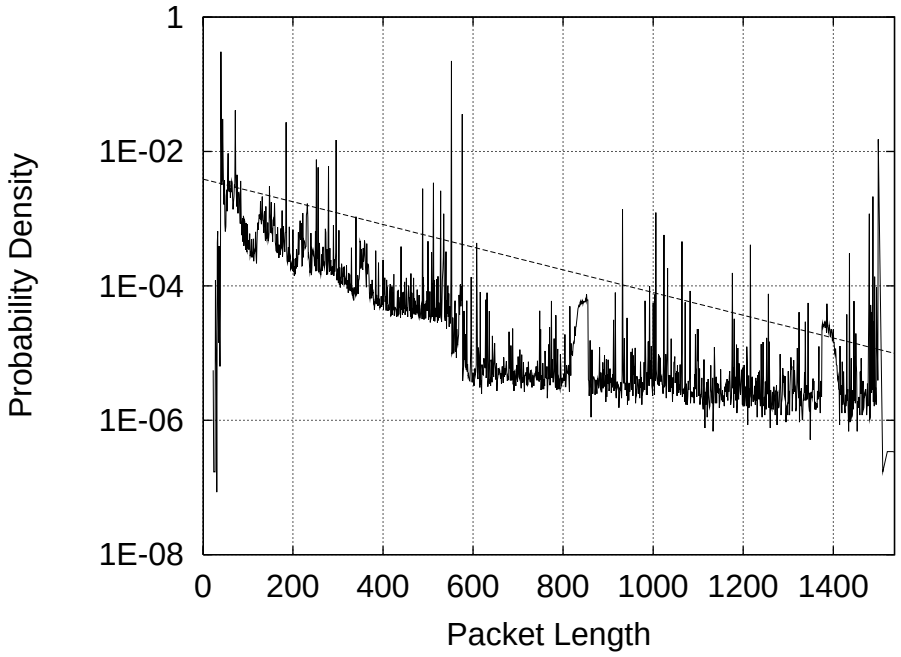

Figure 13: Distribution of Packet Size at Backbone Router

We have assumed that the behavior of the packet forwarding at the bottleneck router follows an $M/M/1$ queueing system (see Eq. (41)). It seems to be insufficient. For example, the packet size does not exhibits an exponentional distribution. See Figure 13 where the dotted line shows the exponential distribution with same mean. In that case, we can apply the above-mentioned design framework by utilizing the $M/G/1$ queueing system, by appropriately changing Eqs. (33), (34), and (35).

Of course, we cannot guarantee the target throughput to every user even if the our framework is applied. This is an inherent problem of the best-effort based protocol including the Internet. We have to repeat our capacity designing process until the throughput performance is satisfied of the end users.

5 Concluding Remarks

In this paper, we have proposed some measurement techniques for establishing the framework of capacity dimensioning in the Internet.

First, we have explained the bandwidth estimation method based on `pathchar`, and proposed two bandwidth estimation methods. From experimental results, we have shown that our methods can produce the robust estimations. Our findings are as follows;

1. `Pathchar` cannot estimate the bandwidth adequately due to two kinds of unexpected errors; a few but very large errors and route alternation. Those pose that measurement errors do not follow some probability distributions such as a normal distribution.

2. We can eliminate exceptionally large errors by utilizing either M-estimation or nonparametric least square fitting methods.

3. By clustering the measured RTTs and selecting an appropriate cluster, errors introduced by route alternation can be avoided.

4. By obtaining the confidence interval, a measurement period can be controlled, which makes it possible to avoid bandwidth waste caused by unnecessary probes in some cases. If the link is congested, on the other hand, more probes are transmitted according to our method. Then accurate and, more importantly, reliable estimation becomes possible.

5. Between parametric and nonparametric approaches, the latter is adequate for reliable bandwidth estimation, but it requires more measurement time. The parametric approach (i.e., the M-estimation method) is better in the measurement and computational time. Perhaps, the choice between them depends on the link condition. If the link load is not high, the obtained measurement data is stable. Then, the assumption that the measurement errors follow the normal distribution would be reasonable. Otherwise, the nonparametric approach presented in this paper is necessary.

We have next proposed the bottleneck identification process based on the traffic measurement. We have investigated the characterization method of 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 detect the three kinds of the bottlenecks: the network configuration, receiver–side configuration, and sender–side configuration. Experimental results have shown that our identification process can work well to identify the bottleneck in an end-to-end TCP communication.

Finally, we have proposed a design framework in order to determine the adequate link capacity based on the end-to-end measurement. We have described a capacity dimensioning process using the to achieve the TCP throughput to the target value. Using an M/M/1 queueing system for the router, an example capacity dimensioning process is presented for the cases;

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

For the first two cases in the above, we can resize the link capacity based on the amount of target bandwidth. On the other hand, the capacity dimensioning is carefully performed since the background traffic sharing the bottleneck link affects the performance of 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.

Acknowledgments

I would like to express my sincere and deep gratitude to my advisor, Professor Masayuki Murata of Osaka University, who has guided me through this work. His continuous advice and encouragement is acknowledged and greatly appreciated. Thanks for leading me into the area of computer networks.

Special thanks go to Research Associate Shingo Ata of Osaka City University, for his valuable comments and suggestions. Without his support, this thesis would not have been possible.

I would like to express my gratitude to Professor Hideo Miyahara, Assistant Professor Naoki Wakamiya, Research Associate Hiroyuki Ohsaki, Research Associate Go Hasegawa, and Research Associate Shn'ichi Arakawa of Osaka University, for their comments and suggestions on the thesis proposal.

Finally, I thank many friends and colleagues in the Department of Informatics and Mathematical Science of Osaka University for their generous help, enlightening and valuable suggestions.

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

[4] J. Apisdorf, k claffy, K. Thompson, and R. Wilder, "OC3MON: flexible, affordable, high performance statistics collection," in *Proceedings of INET'97*, June 1997.

[5] T. Oetiker, "MRTG: Multi router traffic grapher," `http://ee-staff.ethz.ch/~oetiker/webtools/mrtg/mrtg.html`.

[6] V. Jacobson, "pathchar," `ftp://ftp.ee.lbl.gov/pathchar/`.

[7] Allen B. Downey, "Using pathchar to estimate Internet link characteristics," in *Proceedings of ACM SIGCOMM*, pp. 241–250, August 1999.

[8] B. A. Mah, "pchar: A tool for measuring Internet path characteristics," `http://www.ca.sandia.gov/~bmah/Software/pchar`.

[9] K. Lai and M. Baker, "Measuring link bandwidth using a deterministic model of packet delay," in *Proceedings of ACM SIGCOMM 2000*, August 2000.

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

[11] K. Matoba, S. Ata, and M. Murata, "Improving accuracy of bandwidth estimation for Internet links by statistical methods," *13th ITC Specialist Seminar*, pp. 29.1–29.10, September 2000.

[12] V. Jacobson, "traceroute," `ftp://ftp.ee.lbl.gov/traceroute.tar.Z`.

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

[14] J. W. Tukey, "Introduction to today's data analysis," in *Proceedings of the Conference on Critical Evaluation of Chemical and Physical Structural Information*, pp. 3–14, 1974.

[15] M. G. Kendall, *Rank Correlation Methods*. Griffin, 1970.

[16] G. E. Noether, “Some simple distribution–free confidencd intervals for the center of a symmetric distribution,” *J. Am. Statist. Assoc.,68*, pp. 716–719, 1973.

[17] J. A. Hartigan, *Clustering algorithms*. John Wiley & Sons, 1975.

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

[19] 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.

[20] D. D. Clark, “Window and acknowledgement strategy in TCP,” *IETF RFC 815*, July 1982.

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

[22] LBNL’s Network Research Group, “tcpdump,” `ftp://ftp.ee.lbl.gov/tcpdump.tar.Z`.