INVITED PAPER *Special Issue on Internet Technology II*

On a Network Dimensioning Approach for the Internet

Masayuki MURATA†, ***Regular Member***

SUMMARY In this paper, a network dimensioning approach suitable to the Internet is discussed. Differently from the traditional telephone networks, it is difficult to guarantee QoS for end-users even in a statistically sense due to an essential nature of an end-to-end communication architecture in the Internet. We should therefore adopt another approach, based on the traffic measurement. In the approach, the traffic measurement is performed for monitoring the end-to-end QoS. Then, the network adaptively controls the link capacities to meet the user's QoS demands. For this purpose, the underlying network should support such a capability that the link capacities can be flexibly reused. With the WDM network as an underlying network, an example scenario for network provisioning is finally illustrated.

key words: *Internet, network dimensioning, QoS (quality of service), end-to-end communication, active/passive traffic measurement*

1. Introduction

Network dimensioning (more rigorously, capacity dimensioning in this paper's context) is important to provide stable Quality of Service (QoS) to the Internet users. In the telephone network, we have rich historical experiences on characterizing call demands (in Erlang), and an Erlang loss formula has been widely utilized for network dimensioning. The traffic load estimation is not difficult because the ratio of blocked calls is of primary concern for performance monitoring. Then it can be used for capacity dimensioning of the links in operational networks. Fortunately, the call blocking probability is directly related to the user's perceived QoS as well as the network-oriented QoS. This simple relation has led to the success of the stable and reliable operation of 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 even to estimate the traffic demand. It is because the Internet traffic changes dynamically and frequently. Moreover, dominants of the Internet traffic are TCP-based applications having a capability of adapting to network congestion; if the network falls into congestion, each TCP connection defers packet transmission so that the network resources are shared among active and newly arriving connections. Since performance monitoring within the network can only exhibit the packet-level metrics of, e.g., the transmission time and/or loss probability of IP packets at routers, which is insufficient to identify the QoS level perceived by end-users.

Probably, network dimensioning for the distributed and 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. In the Internet, QoS metrics such as Web document download times never be able to be measured by the network providers, but only by the end users, because the processing of protocols higher than the layer four (i.e., TCP of the transport layer) is performed at end-hosts. Controlling the network congestion is also performed at the end-host with dynamically changes of the window size of TCP. We believe that a first step towards establishing the network dimensioning framework is therefore to characterize the performance by the traffic measurement in an end-to-end fashion.

The measured statistics are then used for dimensioning the network resources. As will be presented in Sect. 3, we have theoretical foundations, which can be utilized for this purpose. A network dimensioning approach is next discussed based on the results. Through the approach, network resources that we should increase can be identified. Of course, we should make continuous efforts to keep the satisfactory QoS for the end users; to measure the QoS, to analyze the bottleneck, and to upgrade the bottleneck resource. Our emphasis is that this kind of a spiral approach is especially important in the Internet.

We also present requirements on the underlying network infrastructure in order to support the abovementioned approach. For establishing an effective and meaningful positive feedback loop, we need a flexible bandwidth management mechanism in the underlying network. ATM has such a capability; one example is a dynamic VP bandwidth management method [1]. More recently, MPLS (Multi-Protocol Label Switching) [2] is standardized for IP to be carried over ATM. Thus, we have a possibility of utilizing such functionality via MPLS, which is often referred to as "traffic engineering."

This paper is organized as follows. We first discuss how QoS is supported for data applications in the Internet in Sect. 2. The network dimensioning approach

Manuscript received July 2, 2001.
Manuscript revised September 3, 2001.

†The author is with Cybermedia Center, Osaka University, Toyonaka-shi, 560-0043 Japan.

suitable to the Internet is next discussed in Sect. 3. Section 4 presents an example scenario for our approach applied to the case of IP over MPLS networks. Our concluding remarks are presented in Sect. 5.

2. How QoS is Supported for Data Applications?

2.1 Network Provisioning in Telecommunications

In the telephone network, the Erlang loss formula has been played a central role to provide satisfactory and stable QoS to the users. It is realized by establishing the following feedback loop.

1. Set the target call blocking probability (e.g., 1% during busiest hours).
2. Estimate traffic characteristics (typically the offered load and call holding time).
3. Apply the Erlang loss formula in order to determine the required link capacities (the number of telephone lines) such that the target call blocking probability can be fulfilled.
4. Build the network according to the estimated values.
5. Perform the traffic measurement, and if the target call blocking probability is not satisfied, go back to Step 3 or 2 to adjust link capacities.

A most important point is that call blocking, which can be monitored by the network providers, is directly related to user's perceived QoS in the telephone network. Of course, call blocking experienced by end users is not determined at a single link. However, the end-to-end blocking probability can be well approximated by summing up call blocking probabilities of the links that the call traverses. It is also applicable to the case where the call traverses multiple carriers. It can be treated by SLA (Service Level Agreement).

The above-mentioned approach can be applied to all the reservation-based networks including the Integrated Services (int-serv) architecture [3], which is a standardization for supporting real-time communications in the Internet. In real-time applications, specifying QoS of communication services is rather straightforwarded. It can be well-defined in terms of, e.g., throughput, packet delays and loss probability, although it is not easy to guarantee the latter two metrics in general [4]. Since we have a clear relationship between the allocated bandwidth and quality perceived by the user in terms of, e.g., MOS (Mean Opinion Score) values [5], the bandwidth reservation is sufficient for treating QoS for those applications. Then, the Engset formula, allowing the connections requiring different bandwidths, can be applied in order to dimension the required line capacity.

2.2 Network Provisioning in the Internet

In contrast to the telephone network, network dimensioning for the data applications is not easy. We have several obstacles.

1. Difficulty in defining performance metrics
   An adequate performance metric for determining the required amount of network resources is not known. One may think that the end-to-end delay (e.g., after clicking the button in the Web page until the corresponding Web document displayed) is suitable for the data applications. One realization seems to be that the delay is divided and allocated to each component, and that each component is designed to assure the allocated delay (as a SLA approach as explained before). However, we have many components on the end-to-end communication path; those include
   - physical links affecting propagation delay between sender and receiver,
   - DNS (Domain Name System) for translating the domain name to the IP address,
   - the link, of which transmission capacity decides the packet transmission time,
   - routers deciding packet switching and forwarding times,
   - upper layer-four protocol processing at sender/receiver, and
   - applications.

   Most importantly, we do not have a dominant factor that decides the end-to-end QoS [6]. Thus, it is difficult to be applied especially to the heterogeneous network environments consisting of various kinds of backbone networks, access networks of ISPs (Internet Service Providers), access lines of users to ISP, and server/client computers in the Internet.
2. No way to measure end-to-end metrics
   Even if the performance metric is decided in an end-to-end fashion, the network provider has no means to measure it. Only the user can know his/her performance. It is a quite different point from the traditional telephone network where the network operator can measure the user's QoS by monitoring the numbers of generated/lost calls at the switching node. In the data network, on the other hand, the delay consists of many factors, and the network operator can know only a part of entire delays.
3. Difficulty in predicting the demand
   The Internet traffic is heavily changing and fluctuating. A rapid growth of data traffic makes it more difficult to predict the future traffic demand. The various applications (including the Web-based

services and multimedia applications) have different traffic characteristics. Those facts imply that it is almost impossible to provision the network capacities based on the estimated traffic demands.

4. Complexity in deriving the theoretical foundations As we have emphasized several times, the end-to-end QoS metric is important for the users. However, the protocol hierarchy adopted in the TCP/IP suite makes it complicated to dimension the network based on the theoretical foundations. Namely, the underlying TCP adopts the window-based feedback congestion control in an end-to-end fashion, and we have to take account of it when the end-to-end QoS is identified. See the below; we will discuss this aspect more.

Due to the difficulties mentioned above, QoS in data communication services is probably not to guarantee some performance metrics such as the end-to-end delay or throughput, but to lower the delay at each component as much as possible with *best-effort*. That is, all that we can do is to make efforts for improving each network component separately. We then expect that the entire delay be improved as a result.

Before provisioning, we need to know which component is a limiting factor for the performance improvement. Traffic measurement can be utilized to identify the current cause of the bottleneck in an end-to-end communication. Then, it is used to determine how much resource is invested into the identified bottleneck component. Of course, it results in that the bottleneck point moves to another component, and the next bottleneck component should be found by another traffic measurement. This continuous effort is especially important in order to keep good QoS in data communications.

Do we have any theory determining an amount of the resources to resolve the current bottleneck? A queueing theory has been considered to offer a fundamental theory in the data network during a long time. Its origin can be found in [7], and its usefulness is needless to say in the area. However, the queueing theory only reveals the basic property of a single entity, corresponding to the packet buffer of the router in the case of the Internet. We can find the packet queueing delay and loss probability (for the finite capacitiy of the buffer) by applying the queueing theory. However, the QoS metric of the data network is neither packet delay nor loss probability at the router. The performance at the router is an only component of the entire delay. It is a quite different point from the teletraffic theory (i.e., the Erlang loss formula); the derived call blocking probability is directly related to the user's perceived performance. We have another theory, called a queueing network theory, which treats the network of queues (see, e.g., [8]). However, it does not reflect the dynamic behavior of TCP, which is essentially the window-based feedback congestion control. We thus need another fundamental theory to model and evaluate the data network. One promising approach is based on a control theory that has an ability to explicitly model the feedback loop of the congestion control. See, e.g., [9], [10].

Another simple and rather brute-force approach is that we expect a flexible bandwidth management mechanism to the underlying network. If the link capacity is found to be short, it is increased. Since we always have a possibility of encountering errors in the traffic measurement, an increase of the link capacity may be unnecessary in actual. The underlying network should have a capability of quick adjustments even for the misconfiguration of the link capacities. In the following sections, network provisioning based on this approach is described. Essentially, we abandon the traditional *static* design method, in which the traffic load is assumed to be given a priori. Instead, we utilize a more flexible underlying network suitable to the Internet.

3. Network Provisioning by Traffic Measurement

3.1 A Framework

It is important for the end-users to identify the bottleneck because there are many causes of limiting the performance in an end-to-end communication. If one (or more) links of the network is found to be bottleneck, then the network should take an action of increasing the link capacity.

Fundamentally, we should give up guaranteeing or even predicting QoS for data services. Instead,

- We continuously monitor the end-to-end QoS, and
- If the network resource is short, it is adequately increased.

For the first step, many of current research efforts on traffic measurement are devoted just to acquire the traffic characteristics passively on the link [11]–[14]. For network provisioning, however, it is insufficient. The active measurement for examining end-to-end QoS is important. Furthermore, we need to add the statistical confidence on results for adequately adding the resources, by which the spiral approach for network provisioning is accomplished (Fig. 1). The flexible bandwidth management mechanism in the underlying network is mandatory in this approach. IP over ATM or MPLS can be utilized for this purpose because those networks separate the physical and virtual resources in order to build the logical network. Thus, the logical network has a flexibility to determine the capacities. See the next section for more detail.

3.2 Bottleneck Identification *before* Network Provisioning

Before network provisioning, bottleneck identification

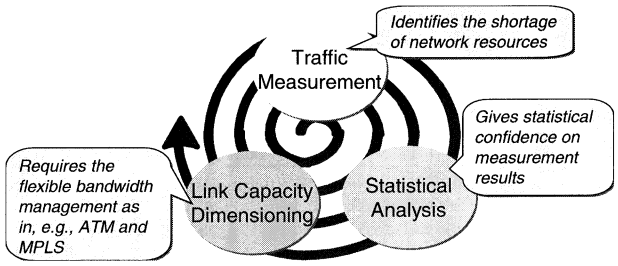

Fig. 1 Spiral approach for network dimensioning.

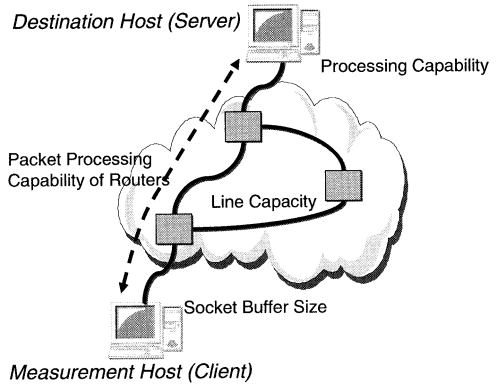

Fig. 2 Measurement environment for bottleneck identification.

is important because there are several causes of the bottleneck in the Internet end-to-end communication mechanism. For example, the Web server can easily become a bottleneck. More essentially, the maximum window size of the TCP connection may limit the performance.

In what follows, we will explain the bottleneck identification based on the traffic measurement. See Fig. 2. Its main goal is to find the limiting factor on the end-to-end performance. 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 which part of the end-to-end communication is a bottleneck in the current network configuration.

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

1. Receiver-Side Configurations: The receiver host prepares a buffer to store the received packets. In the TCP connection, the required buffer size is basically given by the *Bandwidth-Delay Product*, which is determined from the available bandwidth multiplied by RTT (Round Trip Time). The buffer size of the receiver host is notified to the sender by the *Advertised Window* field in the TCP header. The buffer sometimes becomes a bottleneck in the environment with the large *Bandwidth-Delay Product* (e.g., in the network giving 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 after preparing the additional buffer. However, too large a window leads to the bursty packet emission due to the rapid growth of the window size [15], which results in the frequent packet losses. Thus, a careful consideration is necessary in determining the receiver's advertised window size.
2. Sender-Side Configuration: 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 network resource 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 connection processing overhead at the sender host. For example, if we connect to the busy WWW server, a 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 (see, e.g., [16]).
3. Network Configurations: The available bandwidth of one or more links (or router's processing power) is short. A simple example may be found in the case where the customer is connected by a telephone line. The throughput is limited to the modem speed (e.g., 56 kbps). The second and more important reason is due to the link congestion. It leads to many packet losses, and TCP throughput is significantly degraded. It is necessary to increase the capacity of the bottleneck link, which should be resolved by network provisioning.

In order to identify the location of bottlenecks based on the measurement, we can utilize the following equations that characterize the performance of the TCP flow [17]. If the bottleneck is not located at the sender side configuration, the expected window size of the TCP connection is determined by

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

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. However, the mechanism of the delayed ACK [18] is widely used in recent TCP implementations, and its parameter b is typically set at two. Note that if $p = 0$, $E[W]$ cannot be estimated. However, when the link bandwidth is a bottleneck, packet loss is 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

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

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. The end host cannot detect the packet loss directly. One approach is to utilize the active measurement tool such as `pchar` [19], [20] (see the next subsection for detail), but it requires the additional traffic measurement. Another simpler approach is to use the following approximation. The packet loss rate p is determined by checking the sequence number of ACKs as follows. When the packet is lost within the network, the sender host receives the ACK packet with the same acknowledgement number as the previous ACK packet. Those ACKs are called as *Duplicate ACK*s. The sender host cannot determine whether a duplicate ACK is caused by a lost segment or just a mis-ordering of packets, and hence some *margin* is necessary to ignore duplicate ACKs. When three or more duplicate ACKs (called as *Triple Duplicate ACKs*) have been received, the sender host recognizes that the packet has been actually lost. The sender then retransmits the packet and sets the congestion window size to be half. Thus, the packet loss can be recognized 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 normal 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. It is validated by comparing the buffer size W_{max} and the expected window size $E[W]$. In [21], TCP throughput is given as Eq. (2).

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

$$\hat{Q}(w) = \min\left[1, \frac{\left\{\begin{array}{c}(1-(1-p)^3)(1+(1-p)^3\\ \times(1-(1-p)^{w-3}))\end{array}\right\}}{1-(1-p)^w}\right],$$

and

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

T_0 is obtained from the retransmission timeout value (RTO), which is for detecting the packet at the end host [21]. The RTO value is calculated by the following equations:

$$\begin{aligned} RTO &= RTT_m + 4D \\ Err &= RTT - RTT_m \\ RTT_m &= RTT_m + 0.125Err \\ D &= D + 0.25(|Err| - D). \end{aligned} \tag{3}$$

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.

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, it is a good choice to actually measure the TCP throughput for validity check, because the above prediction formula of TCP throughput is occasionally incorrect. The packet loss probability can be determined by 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 result of TCP headers. We may 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. (3) 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.

In summary, the bottleneck can be identified in the following procedures.

1. Traffic Measurements
   Measure the parameters, i.e., RTT, RTO, packet loss rate, TCP throughput, and the physical capacity of bottleneck link, which can be obtained by, e.g., `pathchar` (see the next subsection).
2. Identification of Bottleneck due to the Sender-Side

Table 1 Results on measurement and bottleneck identification.

sender	bottleneck	buffer size	RTT (ms)	loss (%)	measure	estimate	buffer	class
sonet	19.3 Mbps	8 KB	17.7	0.00007	2.13 Mbps	2.62 Mbps	insufficient	receiver-side
		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	
ocu	1.47 Mbps	8 KB	4.03	0.000187	1.35 Mbps	1.62 Mbps	insufficient	network
		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	
iij	29.6 Mbps	8 KB	20.8	0.000573	2.83 Mbps	3.03 Mbps	insufficient	sender-side
		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	

Configuration
Increase the receiver buffer size, and measure the TCP throughput. If the throughput is not improved at some value of the receiver buffer size, the bottleneck is likely to exist at the sender-side. It can be verified by a degree of the difference between the estimated TCP throughput and the measured throughput because Eq. (2) does not consider the sender-side bottlenecks.

3. Identification of Bottleneck due to the Network Configuration
Check whether the receiver buffer size is sufficient or not, which can be verified by Eq. (1). If the buffer size is insufficient, increase it. Otherwise the current cause of the limited performance exists within the network. We can confirm it by actually increasing the receiver buffer size. When the bottleneck is the link capacity, it simply results in that the packet loss rate and RTT values are increased.
4. Identification of Bottleneck due to the Receiver-Side Configuration
When the receiver buffer size is not sufficient, increase the receiver buffer size so that desired performance can be obtained. However, it should be performed carefully. As shown in Eq. (2), the receiver buffer size and the resultant TCP throughput has a linear relation if the receiver buffer size is a cause of the bottleneck. However, as the receiver buffer size is increased, the bottleneck would shift to the other part.

In what follows, we show simple experimental results [22]. In our experiments, we used a `ping` program to obtain the round trip times (RTTs), and `tcpdump` [23] for capturing TCP headers. We placed the receiver host at Osaka University, and chosen three hosts as the sender host (Osaka City University, Sony Communication Network Corporation (`ftp.so-net.ne.jp`) and Internet Initiative Japan Inc. (`ftp.iij.ad.jp`)), which will be referred to as "`ocu`," "`sonet`," and "`iij`," respectively.

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

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` and `iij` cases when the socket buffer size is set to be 8 KB. The `ocu` case shows different results; the bottleneck is moved from the buffer size to the link bandwidth when changing the buffer size from 8 KB to 64 KB. As shown in the table, RTT and the number of lost packets are quite small in the 8 KB buffer case. It means that the bottleneck is not due to the network configuration. Furthermore, RTT and the number of loss packets are increased, which is a typical observation when 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 around 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 change in the packet loss rate; the tendency is different from the `ocu` case. It is classified into the case that the sender-side configuration limits the performance.

3.3 Measurement on Bottleneck Link

Once the link is found to be bottleneck, its capacity and the currently available bandwidth should be obtained. The network provider can directly know it by passive traffic measurements. However, it is sometimes impossible; the bottleneck link may be located outside the provider. Or, the end users may not be able to know it. In those cases, the active traffic measurement is utilized. We have `pathchar` and subsequent `pchar` [19], [20] for measuring latency, capacity, queueing delays and packet loss rate for every link between two hosts. A packet-pair approach [24]–[26] is for measuring the available bandwidth of the designated link. The advantage of those tools is that it is not necessary to

deploy new protocols or any special functions at both of routers and end hosts.

`Pathchar` collects RTTs between source and destination hosts. To measure RTTs, `pathchar` uses one of the ICMP packet, called a *TTL exceeded* message. 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, it decreases the value of the TTL field by one. When the TTL value becomes zero, the router discards the packet and returns the ICMP control packet to the source to inform that the validity of the packet is expired. 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 preset value of the TTL field.

The measured RTT value consists of (1) the sum of queueing delays, q_i, at router i ($1 \le i \le n$), (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 ($1 \le j \le n$). That is, RTT_s, the RTT value for given packet size s, is represented by

$$RTT_s = \sum_{j=1}^{n}\left(\frac{s}{b_j}+\frac{s_{ICMP}}{b_j}\right)+\sum_{i=1}^{n}(q_i+f_i)+2\sum_{j=1}^{n}p_j, \quad (4)$$

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

A typical example for the relation between packet sizes and measured RTTs is shown in Fig. 3. The results were obtained by setting the destination to be `www.gulf.or.jp` from our site. The TTL value was set to 16. 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

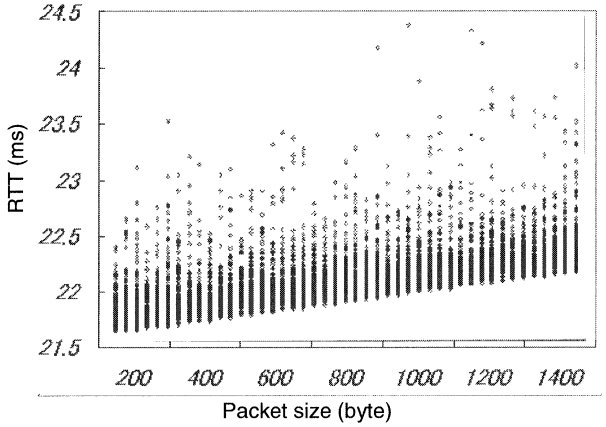

Fig. 3 Distribution of RTT values vs. packet size.

$$minRTT_s = \sum_{j=1}^{n}\frac{s+s_{ICMP}}{b_j}+\sum_{i=1}^{n}f_i+2\sum_{j=1}^{n}p_j. \quad (5)$$

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

$$minRTT_s = s\sum_{j=1}^{n}\frac{1}{b_j}+\alpha. \quad (6)$$

Equation (6) is a linear equation with respect to the packet size s. It is just shown in Fig. 3 if we closely look at the minimum RTT values. By letting the coefficient of the above equation be β_n, we have

$$\beta_n = \sum_{j=1}^{n}\frac{1}{b_j}. \quad (7)$$

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

$$b_j = \frac{1}{\beta_n-\beta_{n-1}}. \quad (8)$$

It is a key idea of `pathchar`.

As indicated above, a difficulty of `pathchar` exists in that the network condition changes frequently in the 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 it wastes 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 in order to calculate β_n, which implies that it assumes errors of minimum RTTs are normally distributed [20]. 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.

Another problem is related to an efficiency of `pathchar`. `Pathchar` sends a fixed number of packets, but the amount of collected data must be changed according to the network condition when measuring the link bandwidth within a reasonable level of accuracy. The authors in [20] 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.

The above problems are addressed in [27];

- By performing a statistical analysis, we add the confidence intervals to the measurement result.
- By applying the nonparametric approach, no assumption on the measurement errors is made on the measurement result.
- By adopting the dynamic control of the measurement intervals, the measurement overhead is minimized. It can be achieved in conjunction with the confidence intervals described above. That is, the measurement is terminated when the desired confidence interval is reached.

In [28], the above procedure is applied to `pchar`. It is a typical example that we need to add the confidence on the result for applying it to the network provisioning method. Note that while the above-mentioned approach is for `pchar` with link capacity estimation, the similar approach can also be applied to other tools.

4. An Example Scenario by Utilizing MPLS

Once the bottleneck is found within a network, the bottleneck link capacity should be increased. It may be realized by adding the physical links. However, it is difficult to estimate an adequate amount of the link capacity that we should increase. Moreover, the traffic in the Internet is heavily fluctuated. It is observed that traffic is stable only during a five-minute observation. Therefore, for a given physical network configuration, the flexible use of the network bandwidth is necessary in the underlying network. That is, a capability of such a flexible bandwidth management of the underlying networks is a key to realizing the spiral approach of the network provisioning in the Internet.

MPLS can offer such a capability. In this section, we explain an example scenario by considering MPLS as an underlying network for network dimensioning. We consider a rather new network architecture, MPλS, where the WDM technology is applied to build MPLS. Note that MPλS is now categorized as a class of GMPLS (Generalized MPLS) [29]. In what follows, we first overview MPλS in order to explain why it can be applied network dimensioning in Sect. 4.1. We then move to our network dimensioning approach in Sect. 4.2.

4.1 A Brief Overview on MPLS

The earliest motivation of MPLS was to simplify wide-area IP backbone network architectures by overlaying IP over new emerging high-speed switching technology. During the mid-90's, the only solution was ATM in which fixed-size packets (called cells) are switched via hardware at each node. Subsequently, many standardization efforts have been devoted to developing packet label switching under the broader MPLS framework, with ATM as a sample underlying technology (see [2] for ATM-based MPLS and MPLS-related terminology). Since the processing of IP packets in the photonic domain is likely not possible for the foreseeable future, MPLS concepts have been further extended to provision lightpath circuit entities, namely MPλS [30], [31].

A key to realizing MPλS is establishing a logical topology, as seen by upper layer protocols, i.e., IP in the current case. The logical topology consists of wavelength paths (called lightpaths), which are configured over the WDM physical network, in order to carry IP packets utilizing the lightpath. Here, the physical network represents an actual network consisting of the optical nodes and optical fiber links inter-connecting nodes. Each optical node contains optical switch devices that can route an input wavelength to an output wavelength. Those include optical cross-connect (OXC) or optical add-drop multiplexer (OADM) nodes. In many cases, these devices can preclude electronic processing, and hence a direct optical connection can be established between two end-point nodes, termed as a lightpath channel [30]. By utilizing the logical topology consisting of lightpaths, even though the physical structure of the WDM networks is fixed, the logical topology now becomes the underlying network to the IP packet layer. In such a network, if the lightpaths are placed between every edge node pairs, i.e., ingress/egress packet LSR (label switch router) nodes (according to MPλS terminology), then no electronic processing is necessary within the network. However, it requires many wavelengths; the order of $O(N^2)$ where N is the number of edge nodes.

Many researchers have studied design methods for logical topology, which also entails a part of the RWA (routing and wavelength assignment) problem. See [32] and references therein. This problem is usually divided into two sub-problems, namely route selection and wavelength assignment. The objective is to maximize the cost function such as wavelength utilization (or minimize the number of required wavelengths on the fiber). The problem is commonly formulated as integer-linear or mixed-integer programming solution, and expectedly, the resulting computational time may be unacceptable for rapid set up. Accordingly, many heuristics have also been proposed in the literature. See [33]. To illustrate a sample application of an optimiza-

tion problem formulation, as used for resolving logical topologies, consider a heuristic algorithm called MLDA (minimum delay logical topology design algorithm) proposed in [34]. MLDA is intended to maximize wavelength utilization and works as follows. First, it places a lightpath connection between two nodes if there is a fiber directly connecting those respective nodes. Then, MLDA attempts to place lightpaths between nodes in the order of descending traffic demands on the shortest-path. Finally, if any free wavelengths still remain, lightpaths are placed randomly utilizing those wavelengths as much as possible.

One problem found in the existing design methods including the above MLDA is that the traffic matrix should be given a priori. Then, the optimization problem is solved exactly or heuristically. However, it is difficult to know the Internet traffic demands in advance. Further, the Internet traffic is heavily changing and fluctuating. Thus, we need a new approach applicable to the Internet, which will be described in the next subsection. MPλS has other problems in carrying the IP traffic; those include a capacity granularity problem, in which the unit of the path capacity between edge nodes is the wavelength capacity. Another problem is concerned with the any-to-any connectivity due to the lack of the number of wavelengths. However, these problems are beyond the scope of this paper. Interested readers refer to, e.g., [35].

4.2 Network Provisioning Using MPλS

If the user's QoS level is found not to be satisfactory by traffic measurement, then lightpaths are newly set up to increase the path bandwidth[†]. Our incremental bandwidth management scheme consists of three phases; an *initial phase*, an *incremental phase*, and a *rearranging phase* [36]. Note that in [36], the authors also consider the backup lightpaths for improving the reliability in IP over WDM networks.

In the initial phase, an IP over MPλS network is built by setting up lightpaths. In this phase, we do not know the traffic demand, but we do need to configure the network anyway. Important is that the estimation on traffic demands is allowed not to be a correct value. The existing design methods for the logical topology can be applied with minor modifications in this phase. For example, MLDA mentioned in the previous subsection may be applied, but the number of wavelengths used for setting up the lightpaths should be minimized so that remaining wavelengths can be utilized for the increasing traffic in the incremental phase.

In our method, reconfiguration of the existing lightpaths is not allowed in the incremental phase. In the incremental phase, the lightpath is added against the new set-up invocations. Those include the request of the user with the lack of QoS level, changes of the traffic demand, or the mis-projection on the traffic demands.

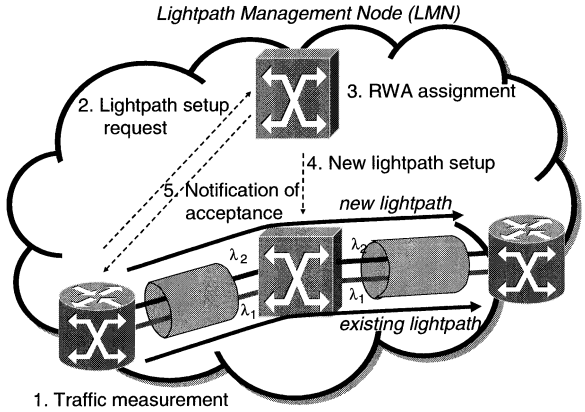

Fig. 4 Logical topology management model in the incremental phase.

In Fig. 4, our logical topology management model is illustrated. In our model, traffic measurement is mandatory. The change of the traffic demand is acquired at the edge-node by monitoring the lightpath utilization. Then, if the utilization of the lightpath exceeds α ($0 < \alpha < 1$), the node requests to a LMN (Lightpath Management Node), which is a special node of managing a logical topology of the WDM network, to set up a new lightpath. It is a simplest form of a measurement-based approach. As described in the previous section, such an approach is insufficient in the data network, and we need an active measurement approach to meet the user-oriented QoS. Anyhow, in our model, we assume that LMN knows the actual traffic demands by the traffic measurement in order to establish the new lightpath. LMN then determines a routing and wavelength assignment for the new lightpath. The new lightpath setup message is issued to the corresponding nodes, so that the configuration of the MPλS network is updated.

An incremental setup of the lightpaths may not lead to the optimal logical topology, i.e., the wavelengths of our logical topology may be less utilized than the one obtained by the static approach. Thus, the new lightpath setup tends to be highly blocked. We therefore need the readjustment phase in which all the lightpaths are reconfigured. The static design method may be applied again in this phase. However, differently from the initial phase, lightpaths are already serving to transport the active traffic. Thus, an influence of the reconfiguration operation should be minimized even if the resulting logical topology would be a local optimum. Note that a global optimal solution tends to require the rearrangement of most lightpaths within the network. Thus, instead of applying the static approach utilizing

[†]In the current case, the unit of the path capacity is a wavelength bandwidth. A fine granularity can be achieved by packet-oriented switching networks such as ATM. See [35].

the global optimal solution, we should configure a new logical topology from the old one step-by-step. One promising method is a branch-exchange method proposed in [37]. The readjustment phase should be performed, e.g., for every month.

One important issue is how the independent requests from the end-users are coordinated for limited physical resources. In the incremental phase, the lightpath setup request is performed on a FIFO basis. When it becomes impossible to accept the new request because of the wavelength shortage, it is simply rejected. It is an alert that the network should be physically upgraded. However, it is not always possible. A fundamental problem is that the capacity granularity of the MPλS networks is coarse, and it is unrealistic that the new lightpath is set up according to the request of the individual user. We should take account of the balance of the wavelength usage to maximize QoS levels of entire users. We need a further research on the fair allocation of the limited resources. If the underlying network has a fine granularity for the capacity, the above problem can be alleviated to some extent. Even in that case, however, we need to tackle the problem for determining the adequate amount of the bandwidth to be increased against QoS requirements of users.

One argument is that all the lightpaths should be provided to the upper layer instead of gradually increasing the lightpaths in the incremental phase. It is because IP has an ability of balancing the traffic so that the available paths are fully utilized. However, it requires halting the operating paths with service interruption. Accordingly, it inhibits the quick response against the user's request, and it is inadequate for providing the high-quality Internet to users.

5. Concluding Remarks

In this paper, we have discussed a network dimensioning approach suitable to the Internet. Due to an essential nature of an end-to-end communication architecture adopted in the Internet, it is difficult to guarantee QoS for end-users even in a statistical sense. We have therefore proposed the new network dimensioning approach based on the traffic measurement. In the approach, the traffic measurement is continuously performed. Then, the network adaptively controls the path capacities to meet the user's QoS requirement. For this purpose, the underlying network should support such functionality. The requirements on the underlying network have also been addressed. We have also presented the example scenario applicable to IP over MPλS. However, it is still rather a conceptual illustration, and we have a lot of research items in this field.

References

[1] S. Shioda and H. Uose, "Virtual path bandwidth control method for ATM networks: Successive modification method," IEICE Trans. Commun., vol.E74, no.12, pp.4061–4068, Dec. 1991.
[2] U. Black, MPLS and label switching networks, Prentice Hall Inc., 2001.
[3] R. Braden, D. Clark, and S. Shenker, "Integrated services in the Internet architecture: An overview," RFC 1633, July 1994.
[4] M. Murata, "Quality of service guarantees in multimedia networks: Approaches and open issues," IEICE Trans., vol.J80-B-I, no.6, pp.296–304, June 1997.
[5] K. Fukuda, N. Wakamiya, M. Murata, and H. Miyahara, "QoS mapping between user's preference and bandwidth control for video transport," Proc. Fifth IFIP International Workshop on Quality of Service, pp.291–302, New York, May 1997.
[6] M. Murata, "Challenges for the next-generation Internet and the role of IP over photonic networks," IEICE Trans. Commun., vol.E83-B, no.10, pp.2153–2165, Oct. 2000.
[7] L. Kleinrock, Queueing systems, volume II: Computer applications, Wiley-Interscience, New York, 1975.
[8] P.G. Harrison and N.M. Patel, Performance modelling of communication networks and computer architectures, Addison-Wesley, 1993.
[9] K. Takagaki, H. Ohsaki, and M. Murata, "Analysis of a window-based flow control mechanism based on TCP Vegas in heterogeneous network environment," Proc. ICC 2001, June 2001.
[10] V. Firoiu and M. Borden, "A study of active queue management for congestion control," Proc. IEEE INFOCOM 2000, March 2000.
[11] V. Paxson, J. Mahdavi, A. Adams, and M. Mathis, "An architecture for large-scale Internet measurement," IEEE Network, vol.36, no.8, pp.48–54, Aug. 1998.
[12] "CAIDA: Cooperative association for Internet data analysis," available at `http://www.caida.org/`.
[13] "IPMA: Internet performance measurement and analysis project," available at `http://www.merit.edu/ipma/`.
[14] "Surveyer home page," available at `http://www.advanced.org/csg-ippm/`.
[15] S. Floyd and V. Jacobson, "On traffic phase effects in packet-switched gateways," Internetworking: Research and Experience, vol.3, pp.397–413, Aug. 1992.
[16] Y. Fujita, M. Murata, and H. Miyahara, "Performance modeling and evaluation of Web systems with Proxy caching," Proc. 16th International Teletraffic Congress, pp.1179–1188, June 1999.
[17] J. Padhye, V. Firoiu, D. Toesley, and J. Kurose, "Modeling TCP throughput: A simple model and its empirical validation," Proc. ACM SIGCOMM'98, pp.303–314, Sept. 1998.
[18] D.D. Clark, "Window and acknowledgement strategy in TCP," IETF RFC 815, July 1982.
[19] V. Jacobson, "pathchar," available at `ftp://ftp.ee.lbl.gov/pathchar/`.
[20] A.B. Downey, "Using pathchar to estimate Internet link characteristics," Proc. ACM SIGCOMM'99, pp.241–250, Aug. 1999.
[21] W.R. Stevens, TCP/IP illustrated, volume1: The protocols, Addison-Wesley, 1994.
[22] K. Matoba, S. Ata, and M. Murata, "Capacity dimensioning based on traffic measurement in the internet," to be presented at IEEE Globecom 2001, Nov. 2001.
[23] LBNL's Network Research Group, "tcpdump," available at `ftp://ftp.ee.lbl.gov/tcpdump.tar.Z`.
[24] V. Paxson, Measurements and analysis of end-to-end Internet dynamics, Ph.D. Thesis, University of California Berkley, April 1997.

[25] R.L. Carter and M.E. Crovella, "Dynamic server selection using bandwidth probing in wide-area networks," Technical Report, Boston University, TR-96-007, March 1996.
[26] C. Dovrolis, P. Ramanathan, and D. Moore, "What do packet dispersion techniques measure?," Proc. IEEE INFOCOM 2001, April 2001.
[27] K. Matoba, "A study on the Internet capacity dimensioning based on traffic measurement and its statistical analysis," Master's Thesis, Graduate School of Engineering Science, Osaka University, Feb. 2001. available at `http://www-ana.nal.ics.es.osaka-u.ac.jp/web01/contents/paper-list.htm`.
[28] K. Matoba, S. Ata, and M. Murata, "Improving accuracy of bandwidth estimation for Internet links by statistical methods," IEICE Trans. Commun., vol.E84-B, no.6, pp.1521–1531, June 2001.
[29] A. Banerjee, J. Drake, J.P. Lang, B. Turner, K. Kompella, and Y. Rekhter, "Generalized multiprotocol label switching: An overview of routing and management enhancements," IEEE Commun. Mag., no.1, pp.144–149, Jan. 2001.
[30] N. Ghani, S. Dixit, and T.-S. Wang, "On IP-over-WDM integration," IEEE Network, vol.38, pp.72–84, March 2000.
[31] D.O. Awduche, Y. Rekhter, J. Drake, and R. Coltun, "Multi–protocol lambda switching: Combining MPLS traffic engineering control with optical crossconnects," IETF Internet Draft, draft-awduche-mpls-te-optical-02.txt.
[32] R. Dutta and G.N. Rouskas, "A survey of virtual topology design algorithms for wavelength routed optical networks," Optical Network Mag., vol.1, Jan. 2000.
[33] H. Zang, J. Jue, and B. Mukherjee, "A review of routing and wavelength assignment approaches for wavelength—routed optical WDM networks," Optical Networks Magazine, vol.1, pp.47–60, Jan. 2000.
[34] R. Ramaswami and K.N. Sivarajan, "Design of logical topologies for wavelength–routed optical networks," IEEE J. Sel. Areas Commun., vol.14, pp.840–851, June 1996.
[35] M. Murata and K. Kitayama, "A perspective on photonic multi-protocol label switching," IEEE Network Magazine, vol.15, no.4, pp.56–63, July/Aug. 2001.
[36] S. Arakawa and M. Murata, "Incremental capacity dimensioning for reliable IP over WDM networks," in Proceeding of OptiComm 2001, Aug. 2001.
[37] J-F.P. Labourdette, G.W. Hart, and A.S. Acampora, "Branch-exchange sequences for reconfiguration of lightwave networks," IEEE Trans. Commun., vol.42, no.10, pp.2822–2832, Oct. 1994.

Masayuki Murata received the M.E. and D.E. degrees in Information and Computer Sciences from Osaka University, Japan, in 1984 and 1988, respectively. In April 1984, he joined Tokyo Research Laboratory, IBM Japan, as a Researcher. From September 1987 to January 1989, he was an Assistant Professor with Computation Center, Osaka University. In February 1989, he moved to the Department of Information and Computer Sciences, Faculty of Engineering Science, Osaka University. From 1992 to 1999, he was an Associate Professor in the Graduate School of Engineering Science, Osaka University, and from April 1999, he has been a Professor of Osaka University. He moved to Advanced Networked Environment Research Division, Cybermedia Center, Osaka University in April 2000. He has more than two hundred papers of international and domestic journals and conferences. His research interests include computer communication networks, performance modeling and evaluation. He is a member of IEEE, ACM, The Internet Society and IPSJ.