

# A study on a receiver-based management scheme of access link resources for QoS-controllable TCP connections

Kazuhiro Azuma<sup>\*,†</sup>, Go Hasegawa and Masayuki Murata

*Graduate School of Information Science and Technology, Osaka University, 1-3 Machikaneyama,  
Toyonaka, Osaka 560-8531, Japan*

## SUMMARY

Although the bandwidth of access networks is rapidly increasing with the latest techniques such as DSL and FTTH, the access link bandwidth remains a bottleneck, especially when users activate multiple network applications simultaneously. Furthermore, since the throughput of a standard TCP connection is dependent on various network parameters, including round-trip time and packet loss ratio, the access link bandwidth is not shared among the network applications according to the user's demands. In this thesis, we present a new management scheme of access link resources for effective utilization of the access link bandwidth and control of the TCP connection's throughput. Our proposed scheme adjusts the total amount of the receive socket buffer assigned to TCP connections to avoid congestion at the access network, and assigns it to each TCP connection according to characteristics in consideration of QoS. The control objectives of our scheme are (1) to protect short-lived TCP connections from the bandwidth occupation by long-lived TCP connections, and (2) to differentiate the throughput of the long-lived TCP connections according to the upper-layer application's demands. One of the results obtained from the simulation experiments is that our proposed scheme can reduce the delay of short-lived document transfer perceived by the receiver host by up to about 90%, while a high utilization of access link bandwidth is maintained. Copyright © 2006 John Wiley & Sons, Ltd.

**KEY WORDS:** TCP (transmission control protocol); access network; socket buffer; QoS (quality of service); resource management

## 1. INTRODUCTION

The rapid increase in Internet users has been the impetus for the performance of backbone networks in solving network congestion posed by increasing network traffic. However, little work has been done in the area of improving the performance of Internet servers despite the projected shift in the performance bottleneck from backbone networks to endhosts or access networks. For example, busy web servers must have many simultaneous HTTP sessions, and server throughput degrades when effective resource management is not considered, even with

---

\*Correspondence to: Kazuhiro Azuma, Graduate School of Information Science and Technology, Osaka University, 1-3 Machikaneyama, Toyonaka, Osaka 560-8531, Japan.

†E-mail: k-azuma@nal.ics.es.osaka-u.ac.jp

*Received 1 March 2004*

*Revised 1 June 2005*

*Accepted 1 July 2005*

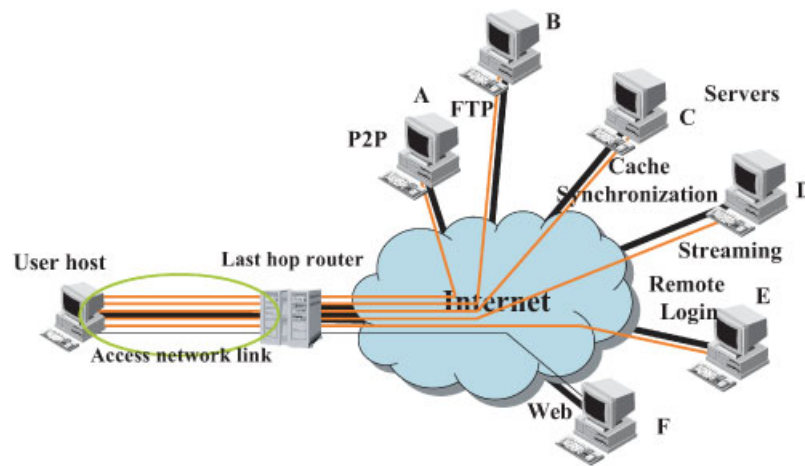


Figure 1. Bottleneck at access network.

large network capacity. Furthermore, web proxy servers [1] must also accommodate a large number of TCP connections, since they are usually prepared by Internet Service Providers (ISPs) for their customers. In our previous work, therefore, we have proposed a TCP connection resource management scheme at endhosts to solve those problems and confirmed its effectiveness through simulation and implementation experiments [2].

On the other hand, the bandwidth of access networks is also increasing rapidly with the latest techniques, such as Digital Subscriber Line (DSL) and Fiber to the Home (FTTH). However, the access link bandwidth remains a performance bottleneck, especially when users activate multiple network applications simultaneously, as shown in Figure 1. In this figure, six TCP connections are established between a user host which becomes a TCP receiver host, and the hosts, A, B, C, D, E and F, which correspond to TCP sender hosts. Each of those connections corresponds to upper-layer applications such as P2P and FTP. For example, when the access link bandwidth is 4 Mbps, which is the typical value on the current Internet in Japan [3], 667 kbps is assigned to each TCP connection when the access link bandwidth is fairly shared. However, since the throughput of the standard TCP connections is affected by various network parameters, including round-trip time (RTT) and packet loss ratio, the access link bandwidth is not shared equally among the network applications. For the same reason, we cannot expect a differentiated throughput for all TCP connections according to the user's demands and the application characteristics. For example, in Figure 1, we cannot intentionally increase the throughput of the TCP connection for P2P and FTP data transmission, and restrict that for the cache synchronization operation which should be done in the background.

Another problem we focus on in this thesis is the performance unfairness between short-lived and long-lived TCP connections. When the access network link is congested and some incoming packets are discarded, the performance of the short-lived connections degrades seriously, compared with that of the long-lived connections [4, 5]. This problem significantly affects the user's perceived performance such as web document transfer delay when they activate long-lived network applications simultaneously.

Therefore, in this thesis, we present a new access link resources management scheme for the effective utilization of the access link bandwidth and the control of the performance of TCP connections. Our proposed scheme *virtually* adjusts the amount of the receive socket buffer for all TCP connections in order to avoid congestion at the access link [6, 7], and assigns it to each TCP connection according to its characteristics and the user's demands for the applications. All TCP connections at the endhost are categorized into two types, which are short-lived connections and long-lived connections. As mentioned above, since the data transfer time in short-lived connections increases greatly when a packet loss occurs, it is necessary to prioritize the short-lived connections, that is, to try not to discard the short-lived connection's packets at the access link. For long-lived connections, on the other hand, it is important to assign the access link resources according to the applications' characteristics and the user's demands, as mentioned above. Thus, the objective of our proposed scheme is to prioritize short-lived TCP connections and differentiate the throughput of long-lived TCP connections, while keeping the utilization of the access link.

The access link resource management scheme proposed in this thesis is implemented in a TCP receiver host, which corresponds to the user host in Figure 1. There are two major reasons for this choice. One is that in the congestion control mechanism of standard TCP [8, 9], a sender host cannot exactly estimate the congestion level of the access link near a receiver host because of the congestion control being performed by the sender host. Another reason is that we cannot control the behaviour of TCP sender hosts to differentiate their throughputs because each TCP connection lives independently of the other connections. That is, the best way is for the receiver host to control the utilization of the access network resources. We also note that our proposed scheme does not modify the congestion control mechanism of TCP, and network protocol structures.

We also consider the situation where a bottleneck exists not at the access network link, but at the endhost resources. We therefore combine in this thesis the access link resource management scheme, and the endhost resource management scheme we have previously proposed [2] into an integrated system. We can apply the integrated system to the various kinds of networks including P2P networks [10–12], where performance bottlenecks exist at various points because of its network dynamics.

The rest of this thesis is organized as follows: In Section 2, we mention some related works on access link resource management and fairness issues between short-lived and long-lived connections. In Section 3, we propose a new access link resource management scheme and confirm its effectiveness by detailing the results we obtained in the simulation experiments in Section 4. Finally, we present our concluding remarks in Section 5.

## 2. RELATED WORK

### 2.1. Receiver-based approaches for access link resource management

There is some research on the management of the access link resources in the previous work [13–15]. The objectives of Reference [13] are to improve the response time of interactive network applications and to keep a high throughput of bulk data transfer. The authors in Reference [13] realize that by assigning a receive socket buffer to each TCP connection in consideration of the bandwidth delay product of the access link and the size of the output buffer of the last-hop

router, which is connected with the access link (see Figure 1). However, since the receive socket buffer for the long-lived connections are limited to only one packet when short-lived connections exist in the network, the throughput of the long-lived connections may be dramatically reduced. Another shortcoming of Reference [13] is that it requires exact knowledge of the available buffer size and bandwidth of the access link from the last-hop router to the endhost.

The authors of Reference [14] propose another sharing method of the access link bandwidth. They consider the packet receiving rate at the user host as the access link bandwidth, and set the receive socket buffer size for TCP connections to differentiate their throughput values by using pre-defined parameters such as priority, minimal rate and weight. Since the main objective of Reference [14] is to improve the performance of long-lived TCP connections providing streaming services over TCP, they do not make any consideration to short-lived TCP connections. This may bring about waste of the access link resources because the long-lived and short-lived connections are equally treated at the TCP receiver and they are assigned the same amount of receive socket buffer.

The authors of Reference [15] proposed delaying TCP ACK packets at the receiver hosts in order to reduce the queuing delay at the last-hop router. However, because of the problems such as the precision of a kernel timer, the mechanism is considered difficult to be implemented.

In Section 4, we compare the performance of the schemes in References [13, 14] with our proposed schemes, and confirm the above-mentioned characteristics.

## *2.2. RTT-based approaches for congestion management*

Our schemes include RTT-based approach for the congestion detection. Therefore, we introduce the work [16] as one of the RTT-based approaches for the congestion management. The authors of Reference [16] proposed the RTT-based approach for the congestion management, which is named TCP Vegas. TCP Vegas dynamically increases or decreases its congestion window size according to observed RTTs of sending packets, whereas TCP Reno only increases its congestion window size until packet loss is detected. That is, TCP Vegas is implemented in sender hosts. Consequently this shows that TCP Vegas is not suitable for our purpose. Moreover, TCP Vegas only performs the congestion control of one TCP connection and cannot take consideration of all the TCP connections managed by a certain host. This also shows that TCP Vegas is not suitable for our purpose.

## *2.3. Performance improvement of short-lived TCP connections*

The approaches that improve the performance of short-lived TCP connections to improve the fairness against long-lived connections can be found in References [4, 17–19]. These schemes generally prioritize short-lived TCP connections by additional mechanisms at the network routers. These kinds of approaches have essential difficulties, especially in implementation issues. One is that most of these mechanisms require the cooperation between edge routers and core routers in the network. Another problem is that these mechanisms assume the availability of Active Queue Management (AQM) mechanisms such as Random Early Detection (RED) [20] and Class-Based Queuing (CBQ) [21], which cannot always be used in the current Internet. Also, there are some problems about not the implementation but the algorithm of these mechanisms. One is that it is essentially difficult to set up optimally some parameters of these mechanisms. For example, in Reference [19], the discussions about the optimal value for some parameters are

not made. Another is that there is no AQM mechanism which realizes the fairness between the short-lived TCP connections and the long-lived TCP connections with the realistic cost.

#### 2.4. Endhost resource management scheme

In Reference [2], we have proposed the endhost resource management scheme. The proposed scheme has the following two mechanisms: control of send/receive socket buffers, and control of persistent TCP connections. As mentioned in Section 1, the objective of this scheme is to improve the performance of an endhost such as a busy web/web proxy server. However, since access link resources may become a bottleneck in the current Internet, the endhost resource management scheme may not be enough to improve the Quality of Service (QoS) perceived by the receiver host. Consequently, we need the additional scheme to manage the access link resources, which is the focus of this thesis.

Therefore, in this thesis, we propose a new scheme for access link resource management that solves the problems in the above approaches. Our proposed scheme prioritizes short-lived connections and differentiates the throughput of long-lived connections in consideration of the network applications' QoS, without degrading the utilization of the access link bandwidth.

### 3. OUR APPROACH AND ALGORITHM

Our proposed scheme can be divided into two mechanisms: adjusting the amount of the receive socket buffer for all TCP connections and assigning it to each TCP connection.

#### 3.1. Adjusting the amount of receive socket buffer for all TCP connections

As mentioned in Section 1, this mechanism is for controlling the arrival rate of packets at the access link and avoiding congestion there. Since the network congestion level changes dynamically, we adjust the amount of the receive socket buffer for all TCP connections at regular intervals. In detail, our proposed scheme periodically measures the RTTs of all TCP connections at the receiver host, and adjusts the amount of the receive socket buffer for all TCP connections according to the measured results as follows and shown in Figure 2:

- When the RTTs of all TCP connections do not increase, we determine that the access link resources are still sufficient and increase the amount of the receive socket buffer for all TCP connections.
- When the RTTs of all TCP connections increase, we decrease the amount of the receive socket buffer for all TCP connections, since it is likely that the congestion occurs at the access link.
- Otherwise, we do not change the amount of the receive socket buffer for all TCP connections.

Also, our proposed scheme estimates the bottleneck link by this approach. That is to say, our proposed scheme can determine whether a bottleneck is the access link or if it is elsewhere for the following reason: When the RTTs of a few TCP connections increase as mentioned in the first case, we can consider that these connections are affected by the congestion at the link through which only these connections pass. That is to say, this congestion is considered to occur at the link other than the access link. Therefore, if the RTTs of a few TCP connections at the

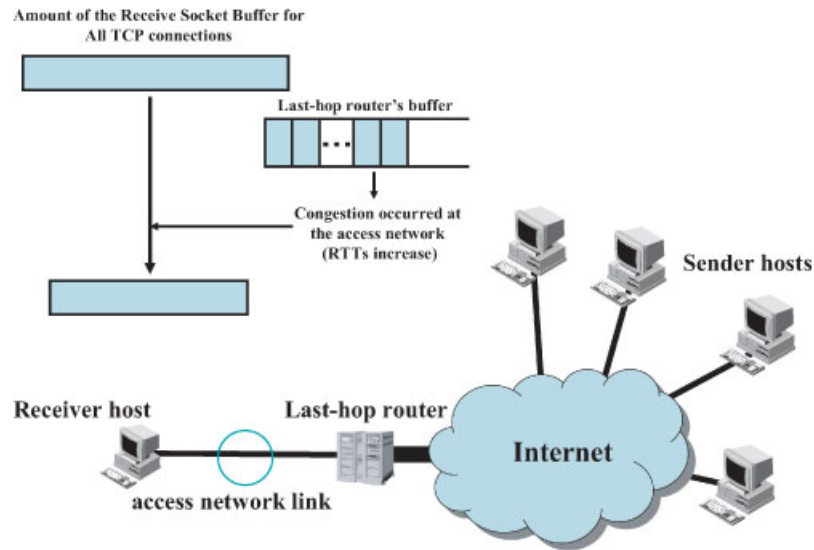


Figure 2. Adjustment of the amount of the receive socket buffer.

receiver host increase, our proposed scheme determines that the bandwidth of the link other than the access link is a bottleneck. On the other hand, when the RTT of most TCP connections increases, we can consider that these connections are affected by the congestion occurring at the identical link, which corresponds to the access link in this case. Consequently, if the RTTs of all TCP connections increase, our proposed scheme determines the access link bandwidth is a bottleneck.

It is important that the amount of the receive socket buffer for all TCP connections be limited to the value determined above, even when the system has sufficient memory capacity and larger memory space can be assigned for the receive socket buffer. This is because if the receive socket buffer size for each TCP connection is too large, the packet transmission rate of the connection unnecessarily increases, which causes the congestion of the access link.

We also note that the meaning of *virtually* adjusting the amount of the receive socket buffer for all TCP connections is to adjust the advertised window size, which reports the current available size of the receive socket buffer to the TCP sender [8], instead of increasing/decreasing the actual size of the receive socket buffer.

### 3.2. Assigning receive socket buffer to TCP connections

Before we assign a receive socket buffer to each TCP connection, we categorize all TCP connections into short-lived or long-lived. This is because the objectives of our scheme are to prioritize short-lived TCP connections, to differentiate the throughput of long-lived TCP connections in consideration of the applications' QoS, and to keep the utilization of the access link. However, the TCP receiver cannot know whether a TCP connection is short-lived or long-lived, since the data size transferred by the TCP connection is not informed in advance. Therefore, in our proposed scheme, we use a threshold-based approach. That is, we use a

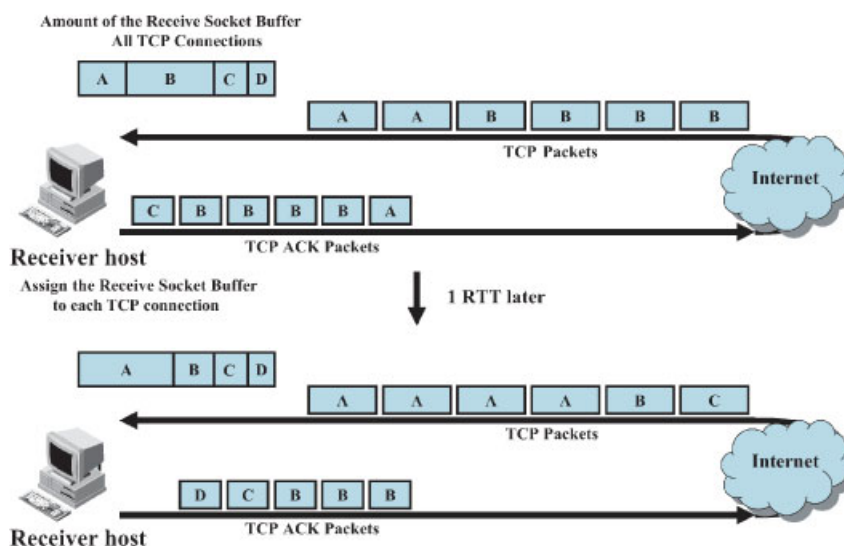


Figure 3. Assignment of receive socket buffer.

threshold value for the receive socket buffer of each TCP connection and categorize the connection by whether the assigned receive buffer size exceeds the threshold value or not. Since all TCP connections are initially categorized as short-lived in this approach, these states are expressed as 'initial state' instead of 'short-lived' and as 'persistent state' instead of 'long-lived' in our scheme. The threshold value is set to the receive socket buffer size, in case we consider all of the TCP connections currently at the receiver host to be in a persistent state.

Then, the receive socket buffer size assigned to each TCP connection is determined as shown in Figure 3. We first assign the receive socket buffer to initial connections preferentially, and then to persistent connections.

(1) For initial TCP connections

We assign the receive socket buffer for initial connections to improve the arrival packet rate from the initial connections at the receiver host. At the same time, our proposed scheme tries not to unnecessarily reduce the throughput of persistent connections when prioritizing initial connections. Therefore, the receive socket buffer size assigned to each initial connection is determined in consideration of the increase algorithm of the congestion window size in TCP's slow start phase.

(1a) When the amount of the receive socket buffer for all TCP connections is sufficient: In this case, the receive socket buffer required by all initial connections can be assigned. Since an initial connection is likely to be in the slow start phase, we focus on the increase algorithm of the congestion window size in the slow start phase to avoid degrading the throughput of persistent connections. That is, the assigned size to the initial connection  $i$  is determined according to the number of RTTs from the beginning of the connection, which is described as  $t_i$ . Consequently, the receive socket buffer size required by connection  $i$  in this case becomes  $2 \cdot 2^{t_i}$  packets.

(1b) When the amount of the receive socket buffer for all TCP connections is insufficient: In this case, the receive socket buffer required by all initial connections cannot be assigned. Therefore, the receive socket buffer is distributed to all initial connections proportionally to the difference

between  $2 \cdot 2^{t_i}$  and the threshold value mentioned above. This is originated by the consideration that it is necessary to prioritize TCP connections just after beginning their data transmission, since they are likely to have small window sizes.

Here, we define the amount of the receive socket buffer for all TCP connections as  $B$ , the number of initial connections as  $N_{is}$ , and the threshold value as  $\text{threshold}_i$ . Then, the receive buffer size assigned to initial connection  $i$ ,  $R_i$ , can be described by the following equations:

$$R_i = \begin{cases} 2 \cdot 2^{t_i} \\ B \cdot \frac{(\text{threshold}_i - 2 \cdot 2^{t_i})}{\sum_j^{N_{is}} (\text{threshold}_j - 2 \cdot 2^{t_j})} \end{cases} \quad (1a, b)$$

(2) *For persistent TCP connections*

It is important to consider each network application's characteristic and user's demands for persistent connections to effectively utilize the access link resources. We assume that each persistent TCP connection has a priority value pre-defined according to the user's demands and the application characteristics.

(2a) *When the amount of the receive socket buffer for all TCP connections is sufficient:* Since the amount of the receive socket buffer for all TCP connections is larger than that required by all initial connections, the remainder is assigned to the persistent connections according to their priority values and RTTs.

(2b) *When the amount of the receive socket buffer for all TCP connections is insufficient:* In this case, we cannot assign enough size of the receive socket buffer for the persistent connections. However, it is necessary to assign at least the receive socket buffer of 1 *mss* for each connection to avoid the TCP's silly window syndrome [22, 23].

Here, we define the amount of the receive socket buffer for all initial connections as  $T_{is}$ , the number of persistent connections as  $N_{ps}$ , the RTT of TCP connection  $i$  as  $r_{tt_i}$ , and the priority value of each TCP connection as  $p_i$ . Then, the receive socket buffer size assigned to each persistent connection  $i$ ,  $R_i$ , is described by the following equations:

$$R_i = \begin{cases} (B - T_{is}) \cdot \frac{p_i \cdot r_{tt_i}}{\sum_j^{N_{ps}} (p_j \cdot r_{tt_j})} \\ 1 \text{ mss}_i \end{cases} \quad (2a, b)$$

Also, these equations show our definition of the fairness among the long-lived TCP connections. We consider that what a user cares is not the network applications' RTT but the network applications' response time. Therefore, we define the fairness in this thesis as the access link bandwidth shared equally by the long-lived TCP connections regardless these RTT.

#### 4. SIMULATION EXPERIMENTS

In this section, we compare our proposed scheme with TCP Reno version, the scheme proposed in Reference [13] and the scheme proposed in Reference [14] through simulation experiments with ns-2 [24] and evaluate the effectiveness of our proposed scheme. In this section, we denote the scheme proposed in Reference [13] *Spring* and that in Reference [14] *Mehra*. We enhance the TCP Reno version and realize our proposed scheme. However, since our proposed scheme

controls the receive socket buffer in one receiver host, our proposed scheme is effective with all TCP variants.

#### 4.1. Simulation scenarios

We consider four scenarios in the simulation experiments, in which we change the network parameters and traffic pattern. In this subsection, we explain the details of each scenario and its objectives.

We explain each scenario's objectives and each situation which we assumed as follows:

- Scenario-1 (The Ideal Case) assumes the most general case. Today, Internet users connect via ADSL whose up/down link bandwidth is about 500 kbps/4 Mbps. Moreover, while they enjoy perusing some web sites, they make use of some bulk file transfer services as network update without realizing them. Since we consider the situation where the bottleneck exists at the access network link, we set up the backbone network parameters so that they may not become the bottleneck.
- Scenario-2 (Large Differences in RTTs and Small Buffers) assumes the case that the receiver host utilizes a very busy ISP and the buffer size assigned to the receiver host's access link at the last-hop router is very small. Also, in this case, when the receiver host accesses the very popular contents server provided by the ISP, we assume the delay between the receiver host and the server will be very large. In this scenario, we evaluate the characteristics of our proposed scheme in a situation where a bottleneck is not the access link bandwidth but the buffer size at the last-hop router. That is, if we focus only on the router buffer size in the control mechanism, the under-utilization of the access link may occur especially when the TCP connections have large RTTs. On the other hand, if we focus primarily on the RTTs of the TCP connections, the buffer at the last-hop router may overflow.
- Scenario-3 (A Large Number of Long-Lived Connections) assumes that P2P application simultaneously downloads a large file from multiple nodes. That is, in this scenario, although many long-lived TCP connections are established between a receiver host and many P2P nodes, a user enjoys perusing a web site. We want to confirm the effect of the number of long-lived connections. Generally, it affects the congestion level of the access link and the frequency of buffer overflow.
- Scenario-4 (Existence of UDP Streaming at the Access Link) assumes UDP streaming services, for example Voice over IP (VoIP), which utilizes UDP packets to transport voice. In this scenario, we confirm the changes of the bandwidth available for TCP connections at the receiver host as this UDP traffic may affect the performance of the proposed scheme, Spring and Mehra. The rate of this UDP traffic is changed as follows: 1 Mbps in 50–150 s and 500–600 s, 2 Mbps in 150–250 s and 600–700 s, and 3 Mbps in 250–350 s and 700–800 s.
- Scenario-5 (Difference of Long-Lived Connections) assumes the download of some files from two or more servers for the network update. We consider that it is possible to perform the network update efficiently, if the priority according to the order of updating each file is set to the download of one from each server. In this scenario, we confirm the effectiveness of differentiating the throughput of the long-lived TCP connections according to the upper-layer application's demands. In this scenario, we do not evaluate the

effectiveness of Spring, because Spring is not a scheme in the consideration of the priority of network applications according to the user's demands.

In Scenario-1, Scenario-2, Scenario-3 and Scenario-4, our proposed scheme (and Spring and Mehra for comparison) is implemented at the receiver host. The flow of these simulations is as follows. The bulk data transfers are performed from the sender hosts A–C (or F in Scenario-3) to the receiver host (long-lived connections) so that the access link bandwidth be fully utilized. At the same time, 100 short-lived connections, each of which transfers 30 KBytes data, start being activated from the sender host D (or G in Scenario-3) at 100 s with random intervals (5 s average). Also, in Scenario-5, our proposed scheme (and Mehra for comparison) is implemented at the receiver host. The three bulk data transfers are performed from the sender host A to the receiver host (long-lived connections). At the same time, 2 long-lived connections, each of which transfer the bulk data for 30 s, start being activated from the sender host B at 25 and 45 s. The interval to check RTTs of TCP connections in our proposed scheme is 5 s and we set the parameters as in Spring and Mehra as summarized in Table II, except that the parameters of Mehra (priority, minimal rate and weight) is not set. Note that the parameters in Table II are the values recommended in the papers [13, 14] and there is almost no difference in parameter selection caused by the changes in network topology and/or simulation.

Figures 4–8 show the simulation models of these scenarios. And, Table I shows some parameters in these simulation models. The fundamental structure of these simulation models consist of some sender hosts and one receiver host. And, some parameters between the sender hosts and the receiver host are shown in Table I.

We use this scenario to evaluate the fundamental characteristics of the proposed scheme, Spring, and Mehra. Note that in the simulation in this section, all of the schemes try to provide equal throughput for long-lived TCP connections.

## 4.2. Simulation results

4.2.1. *Scenario-1: The ideal case.* Figures 9 and 10 show the cumulative relative frequencies (CDFs) of connection establishment time and data transfer time for short-lived connections, the

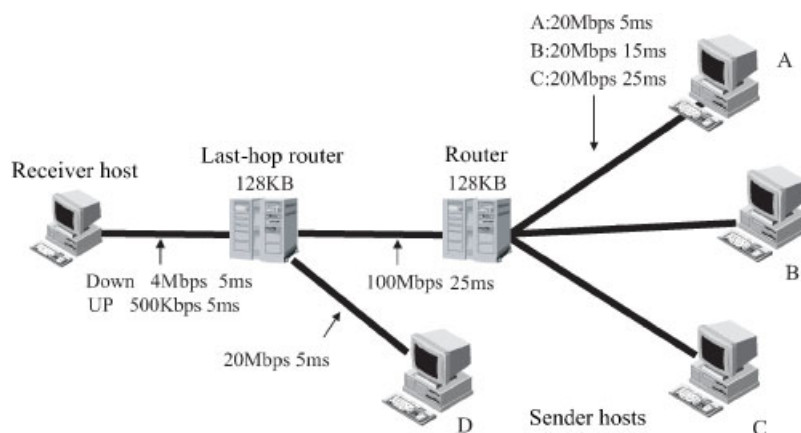


Figure 4. Simulation topology of Scenario-1.

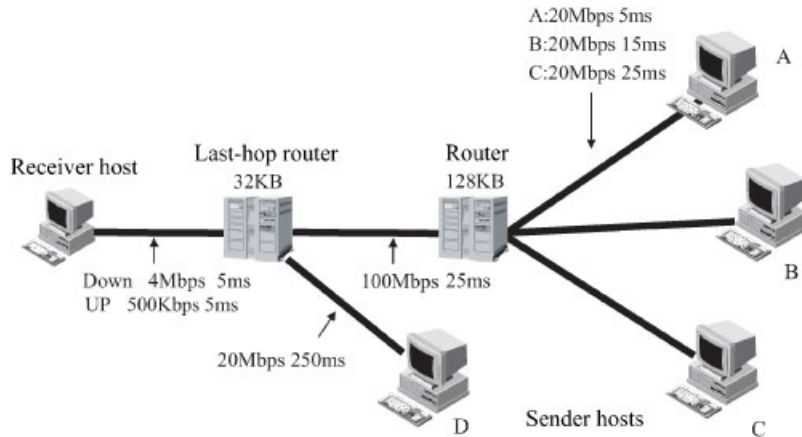


Figure 5. Simulation topology of Scenario-2.

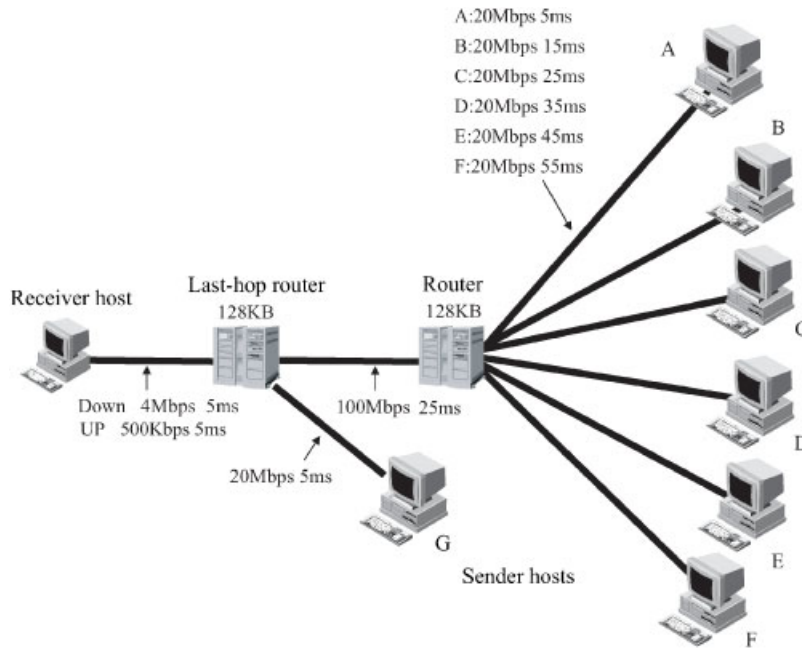


Figure 6. Simulation topology of Scenario-3.

change of utilization of the access link during 500 s simulation time, and the change of the average queue length of the router buffer. In these figures, our proposed scheme is shown by the solid line and labelled as ‘proposed,’ the scheme of Reference [13] by the one point broken line and as ‘Spring,’ the scheme of Reference [14] by two point broken line and as ‘Mehra,’ and TCP Reno by the one point chain line and as ‘traditional.’ From Figures 9(a) and (b), we can

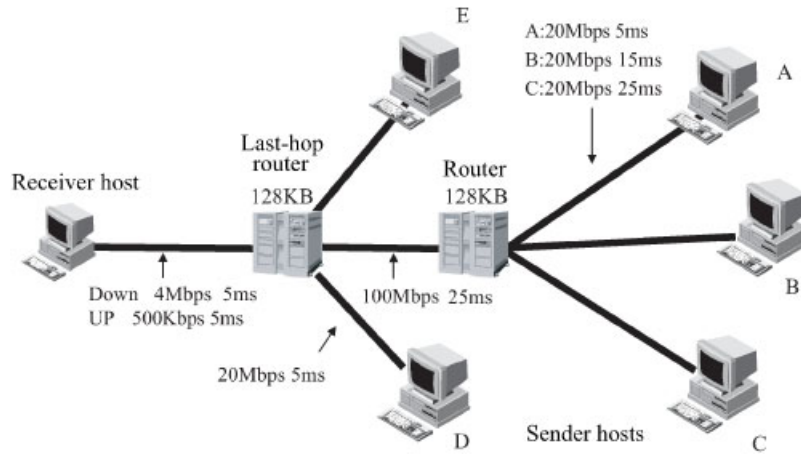


Figure 7. Simulation topology of Scenario-4.

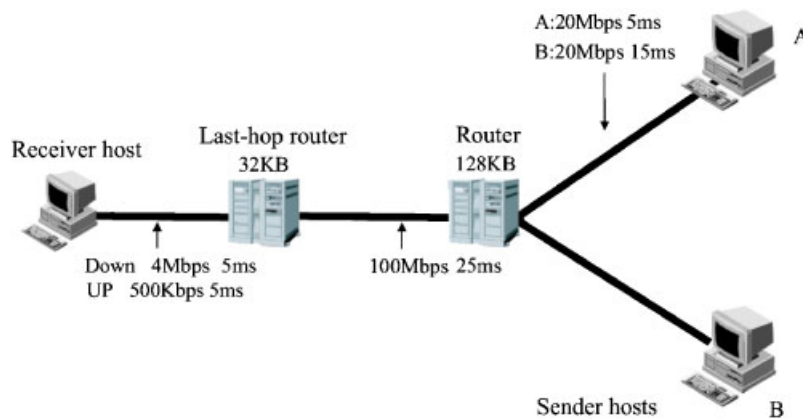


Figure 8. Simulation topology of Scenario-5.

observe that the traditional scheme, which has no special control, shows the longest connection establishment and data transfer times for short-lived connections. This is because the traditional scheme cannot exactly estimate the access link resources, and many packet losses occur at the buffer of the last-hop router due to congestion at the access link. Although the traditional scheme shows a high enough utilization of the access link as shown in Figure 10(a), most of the bandwidth of the access link is occupied by the long-lived connections, while that of the short-lived connections is very low.

From Figures 9(a) and (b), Mehra shows the shortest connection establishment and data transfer times for short-lived connections, but the lowest utilization of the access link from Figure 10(a). Since Mehra tries to assign the same bandwidth for short-lived and long-lived connections, meaning that the access link bandwidth (4 Mbps) is equally shared among three

Table I. Parameters in simulation models.

	Scenario-1	Scenario-2	Scenario-3	Scenario-4	Scenario-5
Last-hop router's buffer size	128 KB	32 KB	128 KB	128 KB	128 KB
Propagation delay A	35 ms	35 ms	35 ms	35 ms	35 ms
Propagation delay B	45 ms	45 ms	45 ms	45 ms	45 ms
Propagation delay C	55 ms	55 ms	55 ms	55 ms	55 ms
Propagation delay D	10 ms	255 ms	65 ms	10 ms	
Propagation delay E			75 ms		
Propagation delay F			85 ms		
Propagation delay G			10 ms		
Access link bandwidth (down)			4 Mbps		
Access link bandwidth (up)			500 kbps		
Packet size			1500 bytes		
Priority of Connection A	0	0	0	0	5
Priority of Connection B	0	0	0	0	10
Priority of Connection C	0	0	0	0	20
Priority of Connection D	0	0	0	0	100
Priority of Connection E	0	0	0	0	200

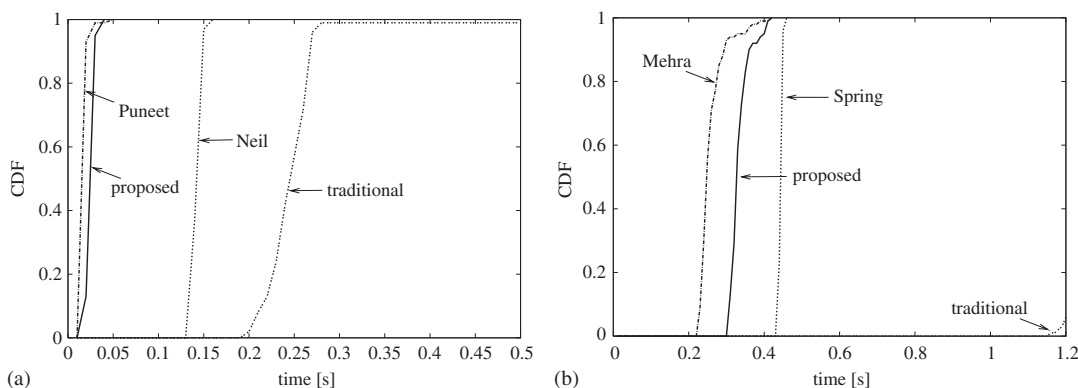


Figure 9. Simulation results of Scenario-1 (1-1): (a) short-lived connections establishment time; and (b) data transfer time of short-lived connections.

long-lived connections and one short-lived connection in this case. Consequently, the access link bandwidth becomes under-utilized, since the bandwidth assigned to the short-lived connections cannot be fully utilized. The near-zero average queue length of Mehra in Figure 10(b) also confirms the under-utilization of the access link.

From Figure 10(b), Spring shows that the average queue length at the last-hop router is relatively long. This is because Spring assigns a receive socket buffer to each TCP connection so that half of the router buffer is utilized. This results in an increase in the connection establishment and data transfer times for short-lived connections, as shown in Figures 9(a) and (b). Note that since the access link bandwidth is not so large, the queuing delay at the last-hop router cannot be ignored. On the other hand, our proposed scheme shows that the average

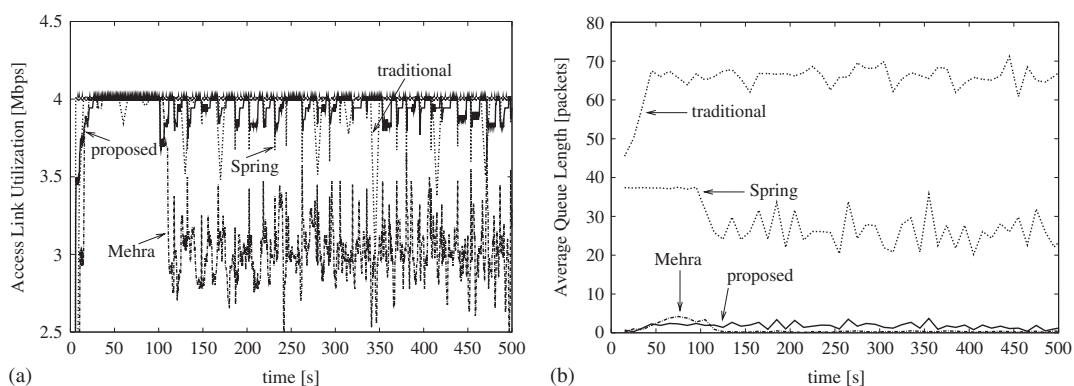


Figure 10. Simulation results of Scenario-1 (1-2): (a) access link utilization; and (b) average queue length.

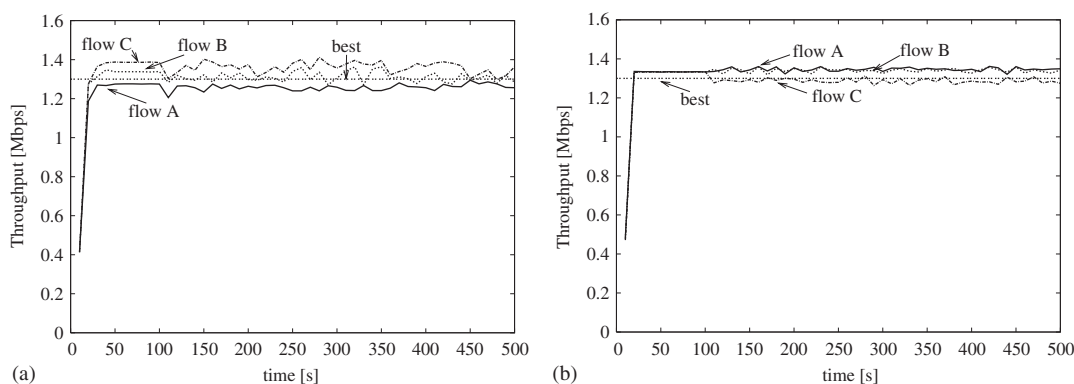


Figure 11. Simulation results of Scenario-1 (2-1): (a) proposed scheme; and (b) spring.

queue length at the last-hop router is small and the connection establishment and data transfer times for short-lived connections are also small, while a high utilization of the access link bandwidth is maintained.

Figures 11 and 12 show the change of the throughput of the long-lived connections in the simulation time. In this figure, we label the throughput of the connection from the sender host A as 'flow A,' that from the sender host B as 'flow B,' and that from the sender host C as 'flow C,' respectively. The label of 'best' represents the throughput value when the access link bandwidth is shared most fairly and effectively. From these figures, our proposed scheme, Spring and Mehra show positive fairness among the long-lived connections, compared with the traditional scheme. However, Mehra shows the lowest throughput and the largest fluctuation of the throughput. This is because Mehra repeats the adjustment to make all TCP connections share the access link bandwidth equally. On the other hand, as we can see from Figures 11(a) and (b), our proposed scheme and Spring show almost the same throughputs as the 'best' case.

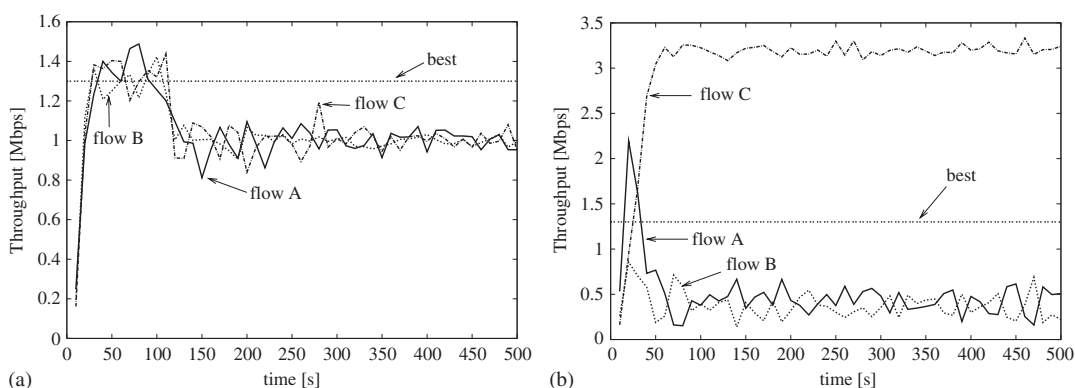


Figure 12. Simulation results of Scenario-1 (2-2): (a) Mehra; and (b) traditional scheme.

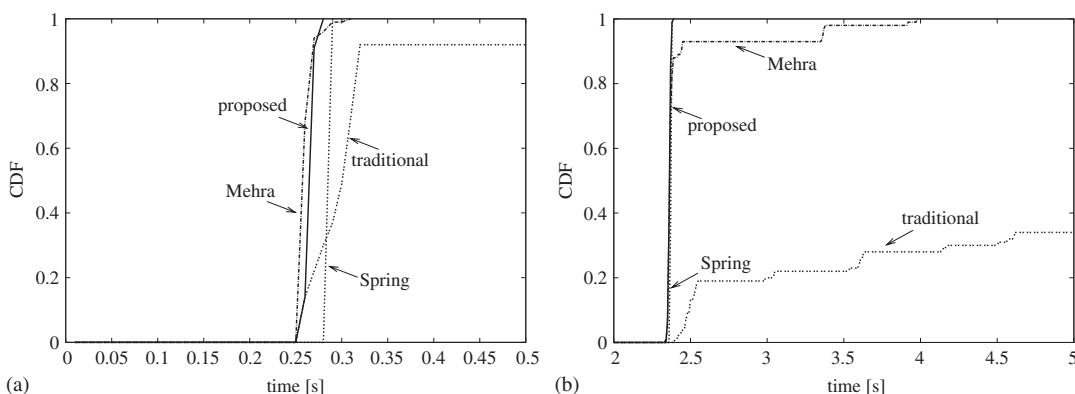


Figure 13. Simulation results of Scenario-2(1): (a) short-lived connections establishment time; and (b) data transfer time of short-lived connections.

*4.2.2. Scenario-2: Large differences in RTTs and small buffers.* Figures 13 and 14 show the simulation results of Scenario-2, as shown in Figures 9 and 10 for Scenario-1. From Figure 14(a), Spring shows a decrease in the utilization of the access link. This is because Spring limits the receive socket buffer for the long-lived connections to only one packet when short-lived connections exist in the network. That is to say, in this scenario, since the propagation delays of short-lived connections becomes larger than Scenario-1, the short-lived connections occupy the receive socket buffer for a longer time. Consequently, the period in which Spring limits the receive socket buffer for the long-lived connections also becomes long, which results in the degradation of the access link utilization. On the other hand, our proposed scheme shows the higher utilization of the access link than that of Spring and Mehra, while maintaining the small connection establishment and data transfer times for short-lived connection. This is because our scheme determines the receive socket buffer size for each short-lived connection in consideration of the increase algorithm of the congestion window size in the TCP's slow start phase. From Figure 13(b), we observe that the difference between the data transfer time for short-lived connections of our proposed scheme and that of Spring is less than that in

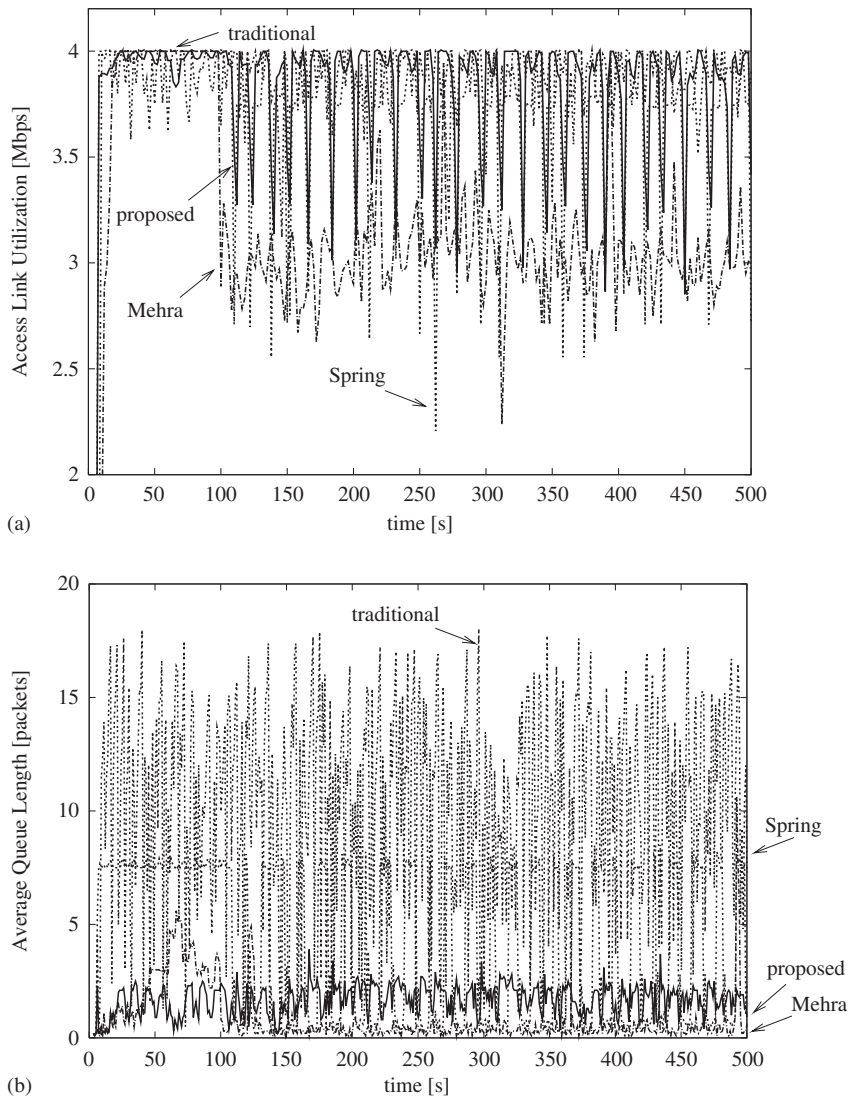


Figure 14. Simulation results of Scenario-2(2): (a) access link utilization; and (b) average queue length.

Scenario-1. This is because the queuing delays at the last-hop router are much shorter than the propagation delays for the short-lived connections, and the data transfer time for short-lived connections is less affected by the queuing delay.

4.2.3. *Scenario-3: A large number of long-lived connections.* Figures 15 and 16 show the CDFs of the connection establishment and data transfer times for short-lived connections, the fairness index [25] among the throughput of the six long-lived connections, and the change of the

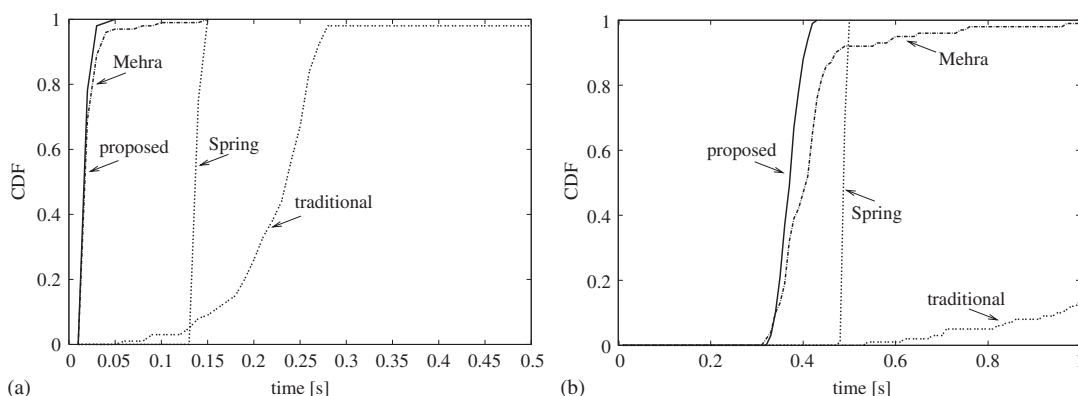


Figure 15. Simulation results of Scenario-3(1): (a) short-lived connections establishment time; and (b) data transfer time of short-lived connections.

average queue length of the router buffer. The value of this fairness index is always between 0 and 1. The closer the value is to 1, the fairer the throughput of each long-lived connection becomes. The other way around, the values closer to 0 are less fair. That is to say, we define the fairness between long-lived connections as the access link available bandwidth being equally shared by all long-lived connections. As an example, if the fairness index is 0.20, it means that the access link bandwidth equally shared by each long-lived connection is fair for 20% of all long-lived connections.

In this scenario, since the congestion occurs more frequently at the access link, Mehra shows the larger average queue length than that in Scenario-1, as shown in Figure 16(b). This brings the increase of the data transfer times for short-lived connections in Mehra as presented in Figure 15(b). Moreover, from Figure 16(a), Mehra shows lower fairness than the others. This is because Mehra adjusts the receive socket buffer according to the throughput of each TCP connection, and this adjustment approach enlarges the fluctuation of the throughput itself, especially when the access link is highly congested. On the other hand, our proposed scheme shows the high fairness among the throughput of the six long-lived connections, while maintaining a small connection establishment and data transfer times for short-lived connections. This shows the effectiveness of our proposed scheme in being independent of the congestion level of the access network link.

*4.2.4. Scenario-4: Existence of UDP streaming at the access link.* Figures 17 and 18 show the CDFs of the connection establishment and data transfer times for short-lived connections, the change of the utilization of the access link bandwidth and the packet loss rate during the simulation.

In Figure 18(b), Spring shows a high packet dropping rate. This is because the change of the access link bandwidth which TCP connections can use is not considered. That is to say, even if the available access link bandwidth for TCP connections changes caused by the UDP flow in this scenario, Spring cannot deal with the change since it controls the throughput of TCP connections only according to the parameters shown in Table II. On the other hand, Mehra also shows the high packet loss rate. Since Mehra shows the lowest utilization of the access link after

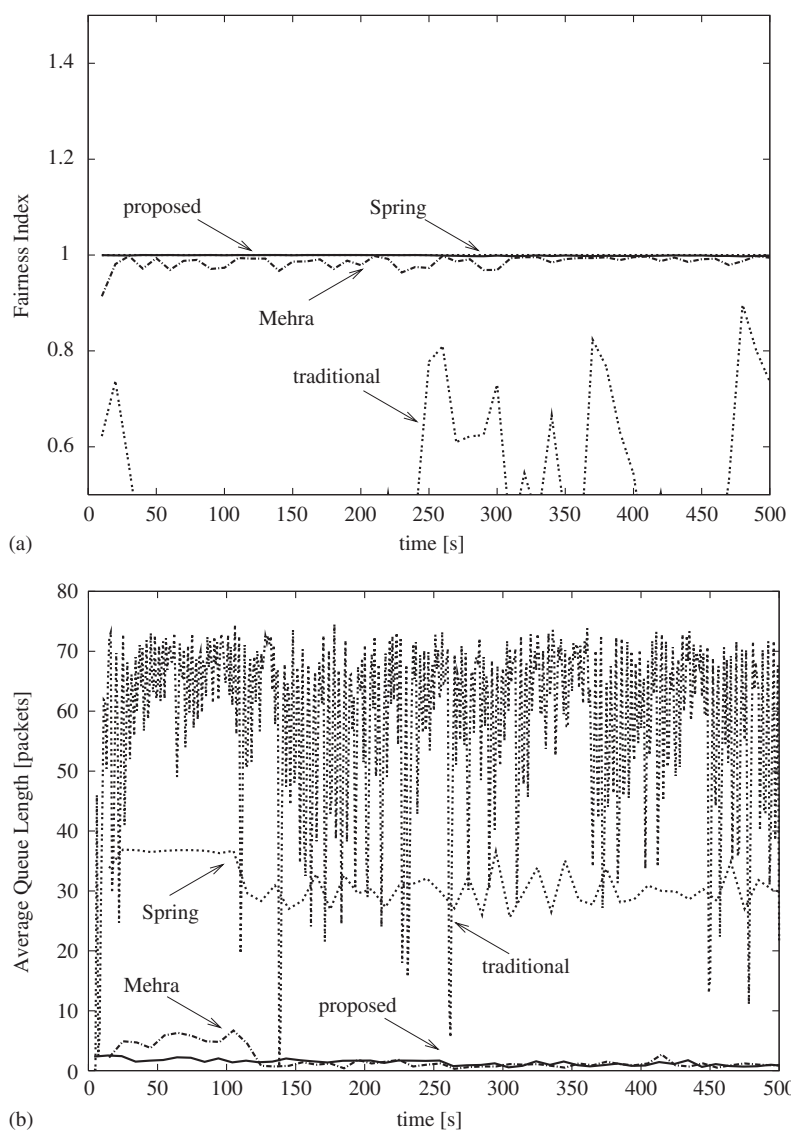


Figure 16. Simulation results of Scenario-3(2): (a) fairness among long-lived connections; and (b) average queue length.

400 s as shown in Figure 18(a), these packet losses are considered to occur only when long-lived connections are active from 0 to 400 s. The main reason is that Mehra focuses on the throughput of TCP connections in its control mechanism. That is, although the available access link bandwidth for TCP connections changes due to the UDP packets, Mehra tries to fully utilize the available access link bandwidth estimated from the throughput of TCP connections.

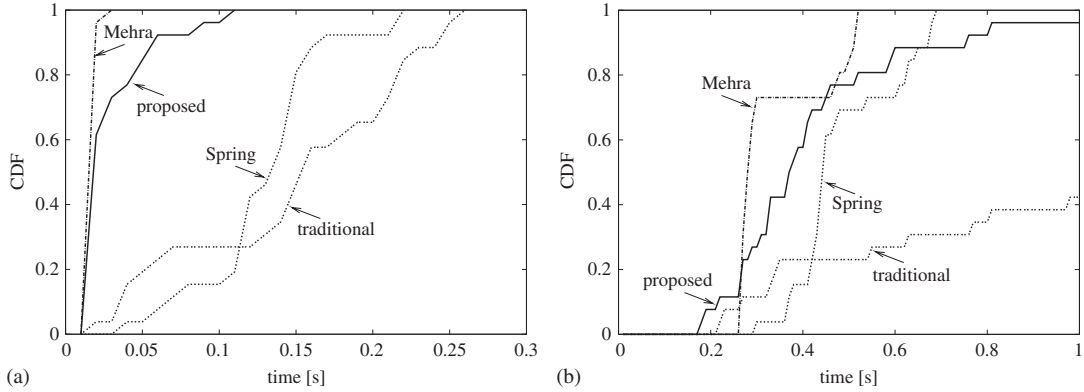


Figure 17. Simulation results of Scenario-4(1): (a) short-lived connections establishment time; and (b) data transfer time of short-lived connections.

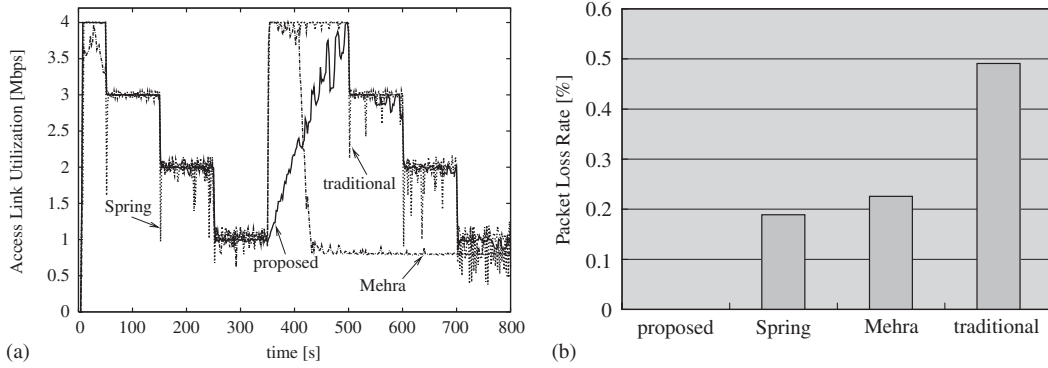


Figure 18. Simulation results of Scenario-4(2): (a) access link utilization; and (b) packet loss rate.

Table II. Parameters in Spring and Mehra ([13], [14]).

Spring ([13])	Mehra ([14])
$X_{put_{link}}$	4 Mbps
$QLength_{Loss}$	64 KBytes
$QLength_{Delay}$	5 KBytes
$RcV_{short}$	2 KBytes
$RcV_{long}$	Weight of Connection A (in Scenario-5)
	Weight of Connection B (in Scenario-5)
	Weight of Connection C (in Scenario-5)
	Weight of Connection D (in Scenario-5)
	Weight of Connection E (in Scenario-5)
	Priority
	Minimal rate
	Weight
	0 (except Scenario-5)
	5
	10
	20
	100
	200

As a result, Mehra over-estimates the available access link bandwidth for the TCP connections and the packet losses occur.

On the other hand, our proposed scheme shows no packet loss during the simulation. This is because our proposed scheme observes the congestion level at the access link. That is, even if the

available access link bandwidth for TCP connections changes, our proposed scheme can estimate congestion at the access link from the changes of the RTTs of the TCP connections, and adjust the amount of the receive socket buffer for all TCP connections to avoid congestion.

However, we observe that our proposed scheme indicates an under-utilization of the available bandwidth from 350 to 500 s, as shown in Figure 18(a). This is because our proposed scheme slowly increases its utilization of the access link bandwidth when there is an unused bandwidth. Consequently, when the available access link bandwidth suddenly increases, our proposed scheme cannot catch up with the change immediately. Although it is one of our future works to solve this problem, we consider it a key idea to employ a mechanism to accurately estimate the available bandwidth of the access link.

*4.2.5. Scenario-5: Difference of long-lived connections.* Figure 19 shows the change of the throughput of each connection during the simulation. Figure 19(a) shows the result of our proposed scheme and Figure 19(b) shows Mehra.

From Figure 19(b), Mehra shows the lower throughput of Connection D than Connection C. This is because Mehra over-/under-estimates the receive socket buffer size till Mehra measures precisely RTT. That is, this means that Mehra may assign the receive socket buffer to a TCP connection too much or less, if the RTT of this connection is not measured precisely, since Mehra controls the receive socket buffer in the consideration of the receiving packet rate at the receiver host.

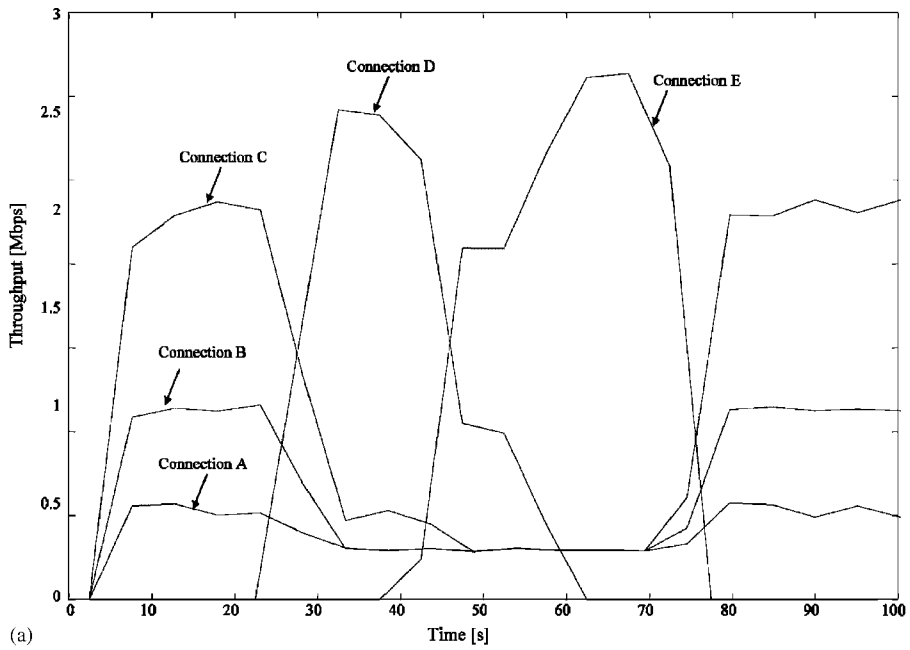
On the other hand, from Figure 19(a), our proposed scheme shows the high throughput of Connection D in proportion to the priority. This is because our proposed scheme can assign properly the receive socket buffer in the consideration of the initial TCP connection. That is, our proposed scheme is not effected easily, even if our proposed scheme cannot measure precisely RTT of a TCP connection in slow start phase.

Moreover, from Figure 19(a), our proposed scheme shows that the throughput of Connection A is equal that of Connection B and Connection C for 70 s from 40 s. This is because our proposed scheme assigns the receive socket buffer of 1 *mss* for each connection to avoid the TCP's silly window syndrome.

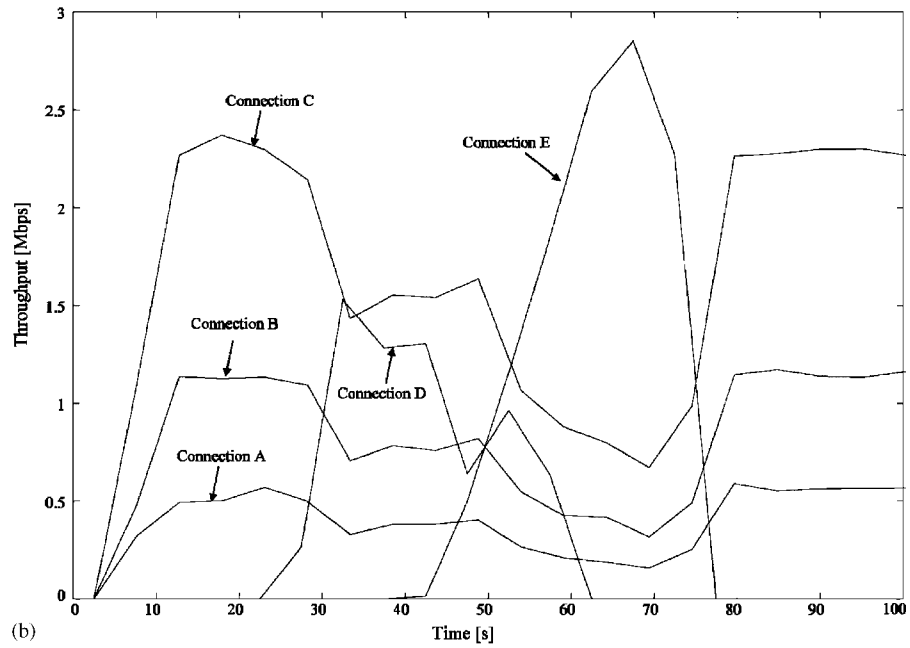
## 5. CONCLUDING REMARKS

In this thesis, we have proposed a receiver-based access link resource management scheme at the receiver host. Our proposed scheme virtually adjusts the amount of the receive socket buffer for all TCP connections at the user hosts in order to avoid congestion at the access link, and assigns it to each TCP connection so that for a short-lived connection, the packets are treated with high priority, and for a long-lived connection, the upper-layer application's QoS and the user's demands are reflected. We have evaluated the performance of our proposed scheme through extensive simulation experiments, and confirmed that it can effectively utilize the access link resources; that is, it can improve the performance of short-lived TCP connections, and maintain the throughput of long-lived connections as expected, while keeping the utilization of the access link bandwidth. Moreover, we have compared our proposed scheme with the schemes in References [13, 14] and confirmed the advantages of our proposed scheme.

Furthermore, we have confirmed that our proposed scheme becomes more effective by integrating with the endhost resource management scheme, by presenting simulation results that



(a)



(b)

Figure 19. Simulation results of Scenario-5: (a) proposed scheme; and (b) Mehra.

the integrated system can deal well with situations where a performance bottleneck exists not at the access link, but at the other links or at the endhost resource. Therefore, we consider that our proposed scheme is effective even in networks where the bottleneck point changes dynamically, such as P2P networks.

As for future work, we plan to implement the proposed scheme in the actual receiver host, and to evaluate it through experiments using the actual network. We consider this implementation will be straightforward since we can estimate from past experience that we can realize our proposed scheme by adding about 1000 codes to the source code of the kernel system. We will have to solve a problem in realizing our proposed scheme. This is the granularity of the timer in the kernel system is too rough. This causes our proposed scheme to measure incorrectly the RTT of TCP connection. On the other hand, we have the outstanding point in order that it is straightforward to implement. The point is to *virtually* adjust the amount of the receive socket buffer for all TCP connections. It is unnecessary for us to perform the socket buffer management, which is the memory management in a kernel system.

## REFERENCES

1. Proxy Survey, available at <http://www.delegate.org/survey/proxy.cgi>
2. Okamoto T, Terai T, Hasegawa G, Murata M. A resource/connection management scheme for HTTP proxy servers. *Proceedings of Second International IFIP-TCP6 Networking Conference*, May 2002; 252–263.
3. Broadband networking report in Japanese, available at <http://www.musen-lan.com/speed/htmldata/>
4. Guo L, Matta I. The war between mice and elephants. *Proceedings of the 9th IEEE International Conference on Network Protocols*, no. 2001-005, November 2001.
5. W3C Recommendations Reduce ‘World Wide Wait’, available at <http://www.w3.org/Protocols/NL-PrefNote.html>
6. Balakrishnan H, Rahul HS, Seshan S. An integrated congestion management architecture for Internet hosts. *SIGCOMM 1999*, September 1999; 175–187.
7. Touch J. TCP control block interdependence. *Request for Comments (RFC) 2140*, April 1997.
8. Postel J. Transmission control protocol (TCP). *Request for Comments (RFC) 793*, September 1981.
9. Stevens WR. *TCP/IP Illustrated, Volume 1: The Protocols*. Addison-Wesley: Reading, MA, 1994.
10. Milojicic D, Kalogeraki V, Lukose R, Nagaraja K, Pruyne J, Richard B, Rollins S, Xu Z. Peer-to-peer computing. *Technical Report HPL-2002-57*, HP Laboratories, March 2002.
11. Gnutella, available at <http://www.gnutella.com/>
12. KaZaA, available at <http://www.kazaa.com/>
13. Spring NT, Chesire M, Berryman M, Sahasranaman V, Anderson T, Bershad BN. Receiver based management of low bandwidth access links. *Proceedings of IEEE INFOCOM 2000*, 2000; 245–254.
14. Mehra P, Zakhor A, Vleeschouwer CD. Receiver-driven bandwidth sharing for TCP. *Proceedings of IEEE INFOCOM 2003*, March 2003.
15. Hsiao P-H, Kung H, Tan K-S. Active delay control for TCP. *Proceedings of IEEE Globecom '01*, September 2001.
16. Brakmo LS, Peterson LL. TCP Vegas: end to end congestion avoidance on a global Internet. *IEEE Journal on Selected Areas in Communications* 1995; 13:1465–1480.
17. Hartling M, Claypool M, Kinicki R. Active queue management for Web traffic. *Technical Report WPI-CS-TR-02-20*, Computer Science Department, Worcester Polytechnic Institute, May 2002.
18. Karandikar S, Kalyanaraman S, Bagal P, Packer B. TCP rate control. *ACM Computer Communications Review* 2000; 30:45–58.
19. Altman E. A stateless approach for improving TCP performance using Diffserv, Submitted.
20. Floyd S, Jacobson V. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking* 1993; 1:397–413.
21. Floyd S, Jacobson V. Link-sharing and resource management models for packet networks. *IEEE/ACM Transactions on Networking* 1995; 3:365–386.
22. Clark DD. Window and acknowledgement strategy in TCP. *Request for Comments (RFC) 813*, July 1982.
23. Wright GR, Stevens WR. *TCP/IP Illustrated, Volume 2: The Implementation*. Addison-Wesley: Reading, MA, 1995.
24. The VINT Project. UCB/LBNL/VINT network simulator - ns (version 2), available at <http://www.isi.edu/nsnam/ns/>
25. Jain R, Durresti A, Babic G. Throughput fairness index: an explanation. *ATM Forum Contribution: AF/99-0045*, February 1999.

AUTHORS' BIOGRAPHIES



**Kazuhiro Azuma** received the ME degree in Information and Computer Sciences from Osaka University, Osaka, Japan, in 2004. In April 2004, he moved to Oki Electric Industry Co. Ltd. His research work is in the area of resource management architecture for future Internet servers.



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



**Masayuki Murata** received the ME and DE 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 Division, Cybermedia Center, Osaka University in April 2000, and he is now a Professor of Graduate School of Information Science and Technology, Osaka University. He has more than two hundred papers of international and domestic journals and conferences. His research interests include computer communication networks, performance modelling and evaluation. He is a member of IEEE, ACM, The Internet Society, IEICE and IPSJ.