Master’s Thesis

Title

A Study on Receiver–based Management Scheme of Access Link Resources for QoS–Controllable TCP Connections

Supervisor

Professor Hideo Miyahara

Author

Kazuhiro Azuma

February 13th, 2004

Department of Information Networking

Graduate School of Information Science and Technology

Osaka University

Master’s Thesis

A Study on Receiver–based Management Scheme of Access Link Resources for QoS–Controllable TCP Connections

Kazuhiro Azuma

Abstract

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. We compare our proposed scheme with the traditional TCP and other related works through simulation experiments and evaluate the effectiveness of our proposed scheme. From these results, we confirm that our proposed scheme has some great effects in terms of the reduction of data transfer time, avoidance of congestion at the access link, and a decrease in delays. One of the obtained results 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.

Furthermore, we consider the situation where the bottleneck exists not at the access network link but at the endhost resources. We therefore combine the access link resource management scheme in this thesis, and the endhost resource management scheme we have previously proposed, into an integrated system. We can apply the integrated system to various kinds of networks, including P2P networks, where performance bottlenecks exist at various points because of network dynamics. We validate the effectiveness of the integrated scheme through some simulation experiments. From these results, we conclusively confirm that the integrated scheme can improve the receiver host performance greatly and, utilize effectily the access link resources and endhost resources.

Keywords

TCP (Transmission Control Protocol), Access Network, Socket Buffer, QoS (Quality of Service), Resource Management

Contents

List of Figures

List of Tables

1 Introduction

The rapid increase in Internet users has been the impetus for the performance of backbone networks into solving network congestion posed against the context of 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 large network capacity. Furthermore, Web proxy servers [3] must also accommodate a large number of TCP connections, since they are usually prepared by ISPs (Internet Service Providers) 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 [4].

On the other hand, the bandwidth of access networks is also rapidly increasing with the latest techniques, such as DSL (Digital Subscriber Line) and FTTH (Fiber to the Home). 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 [5], 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,

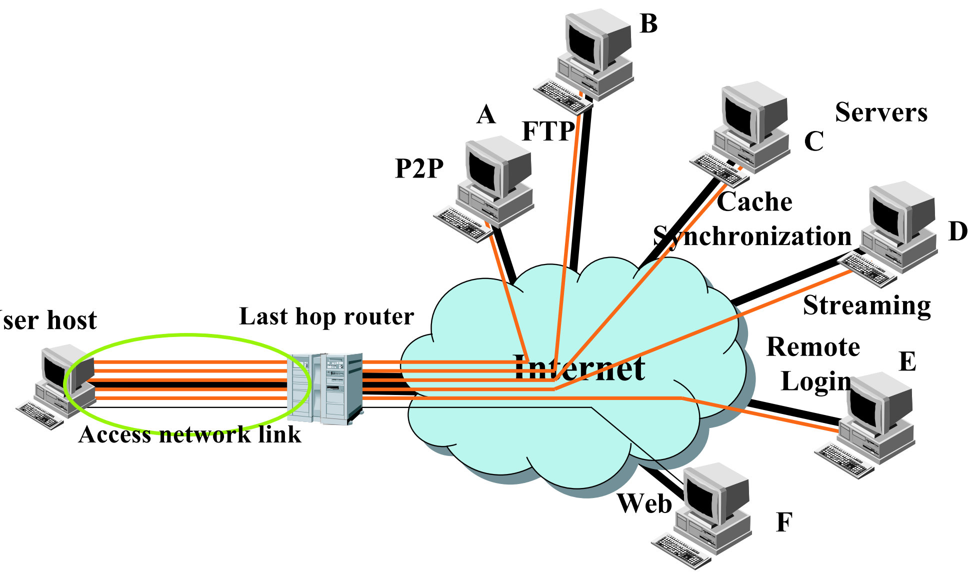


Figure 1: Bottleneck at Access Network

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 [6, 7]. 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 [8, 9], 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 [10, 11], 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 behavior of TCP sender hosts to differentiate their throughputs because each TCP connection lives independently on 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 the access link resource management scheme in this thesis, and the endhost resource management scheme we have previously proposed [4], into an integrated system. We can apply the integrated system to the various kind of networks including P2P networks [12–14], 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. In Section 5, we discuss integration with the endhost resource management scheme for supporting the various kind of networks. Finally, we present our concluding remarks in Section 6.

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 [1, 2, 15]. The objectives of [1] are to improve the response time of interactive network applications and to keep a high throughput of bulk data transfer. The authors in [1] 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 [1] 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 [2] 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 [2] 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 [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 [1] and [2] with our proposed schemes, and confirm the above–mentioned characteristics.

2.2 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 [6, 16, 17]. 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 AQM (Active Queue Management) mechanisms such as RED (Random Early Detection) [18] and CBQ (Class–Based Queuing) [19], which cannot almost be used in the current Internet.

2.3 Endhost Resource Management Scheme

In [4], 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 QoS (Quality of Service) perceived by the receiver host. Consequently, we need the additional scheme to manage the access link resources, which is the focus on 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

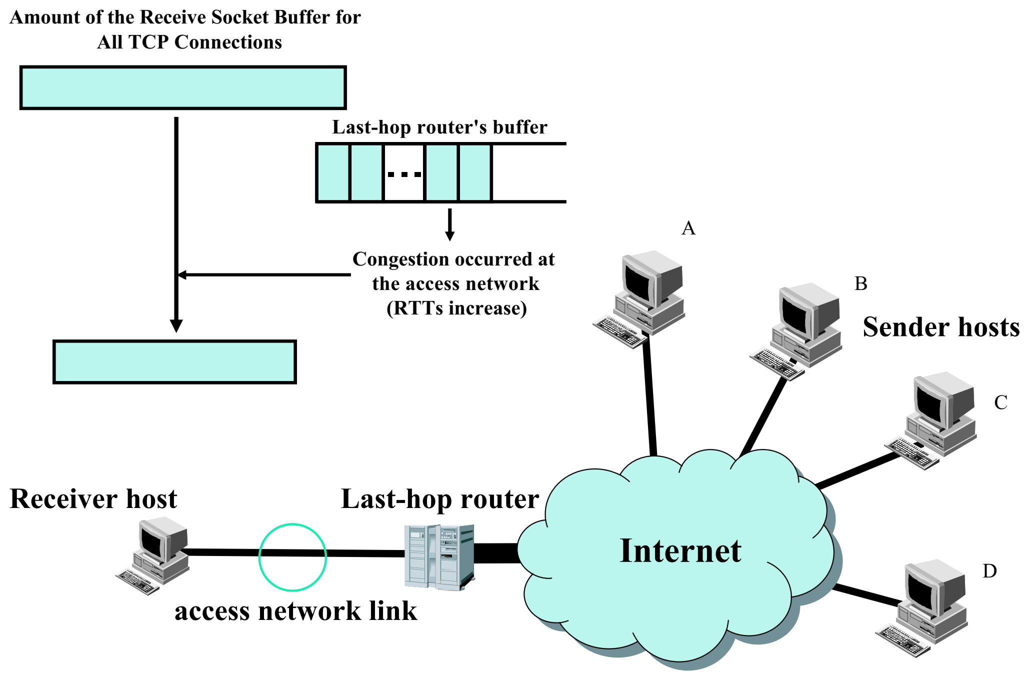
Our proposed scheme can be divided into two mechanisms: adjusting the amount of the receive socket buffer for all TCP connections and assigning it to each TCP connection.

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 dynamically changes, 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.

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.



Amount of the Receive Socket Buffer for All TCP Connections
Last-hop router's buffer
Congestion occurred at the access network (RTTs increase)
A
B
Sender hosts
C
D
Receiver host
Last-hop router
Internet
access network link

Figure 2: Adjustment of the Amount of the Receive Socket Buffer

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 [10], instead of increasing/decreasing the actual size of the receive socket buffer.

3.2 Assigning Receive Socket Buffer to TCP Connections

Before we assign receive a 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 threshold value for the receive

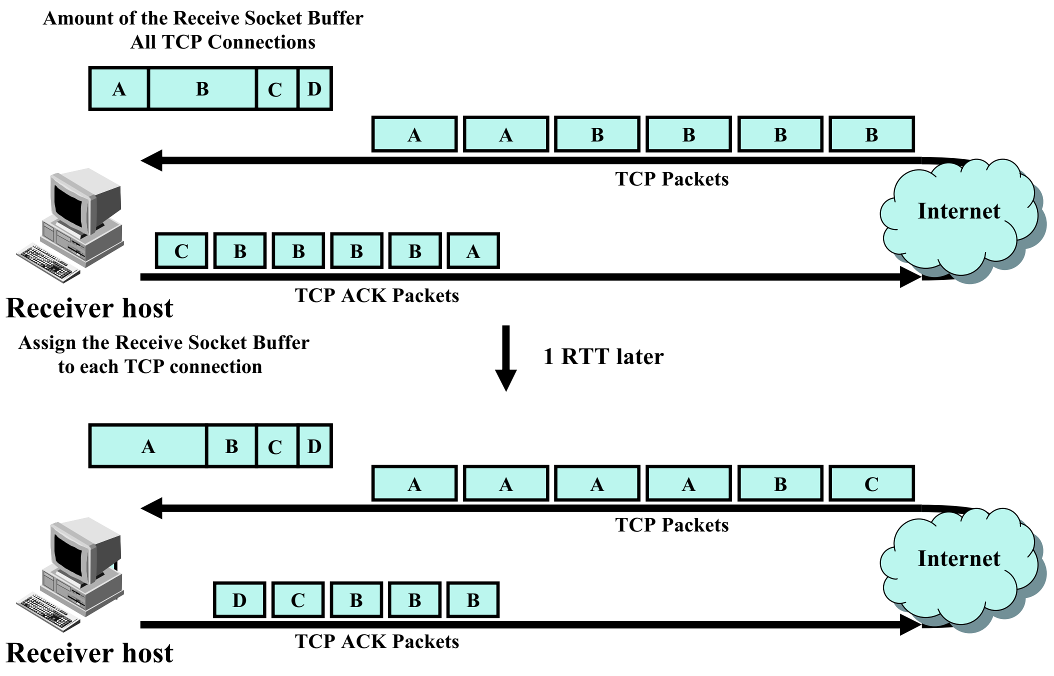


Figure 3: Assignment of Receive Socket Buffer

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 follows and 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.

(1–a) 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.

(1–b) 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 $threshold_i$. Then, the receive buffer size assigned to initial connection i, R_i, can be described as the following equations:

$$R_i = \begin{cases} 2 \cdot 2^{t_i} & \text{(1–a)} \\ B \cdot \dfrac{(threshold_i - 2 \cdot 2^{t_i})}{\sum_{j}^{N_{is}} (threshold_j - 2 \cdot 2^{t_j})} & \text{(1–b)} \end{cases}$$

(2) For persistent TCP connections

It is important to consider each network application's characteristic and user's demands for persistent connections to utilize effectively 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.

(2–a) 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.

(2–b) 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 1 mss for each connection to avoid the TCP's silly window syndrome [20, 21].

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 rtt_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 as the following equations:

$$R_i = \begin{cases} (B - T_{is}) \cdot \dfrac{p_i \cdot rtt_i}{\sum_{j}^{N_{ps}} (p_j \cdot rtt_j)} & \text{(2–a)} \\ 1\, mss_i & \text{(2–b)} \end{cases}$$

3.3 Discussions on Bottleneck Discovery

Our proposed scheme estimates the network congestion level from the changes of the RTTs of all TCP connections at the receiver host. Therefore, if RTTs increase due to a bottleneck link other than the access link, our proposed scheme may fail to estimate the access link congestion. However, when the RTT of most of TCP connections increase, 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. On the other hand, when the RTTs of a few TCP connections increase, 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 receiver host increase, our proposed scheme determines that the bandwidth of the link other than the access link is a bottleneck.

Moreover, as mentioned in Section 1, a bottleneck sometimes exists at the endhost resources such as CPU resources and the amount of receive socket buffer. To support these situations, we combine the access link resource management scheme in this section, and the endhost resource management scheme we have previously proposed [4], into an integrated system. We discuss the integrated system in Section 5.

4 Simulation Experiments

In this section, we compare our proposed scheme with the standard TCP, the scheme proposed in [1] and the scheme proposed in [2] through simulation experiments with ns–2 [22] and evaluate the effectiveness of our proposed scheme. In this section, we denote the scheme proposed in [1] *Spring* and that in [2] *Mehra*.

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.

4.1.1 Scenario–1 : The Ideal Case

Figure 4 shows the simulation model of Scenario 1. It consists of four sender hosts A through D and one receiver host. The propagation delays between the sender hosts and the receiver host are: A : 35 msec, B : 45 msec, C : 55 msec and D : 10 msec, respectively. A performance bottleneck in this scenario is the receiver host's access link bandwidth (4 Mbps). The bandwidths and propagation delays of other links are as described in Figure 4. The router buffer size is set to 128 KBytes, and packet size is 1,500 Bytes. Our proposed scheme (and Spring and Mehra for comparison) is implemented at the receiver host. In this scenario, the bulk data transfers are performed from the sender hosts A through C 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 at 100 seconds with random intervals (5 sec average). The interval to check RTTs of TCP connections in our proposed scheme is 5 seconds and we set the parameters in Spring and Mehra as summarized in Table 1, except that the parameters of Mehra (priority, minimal rate and weight) is not set. Note

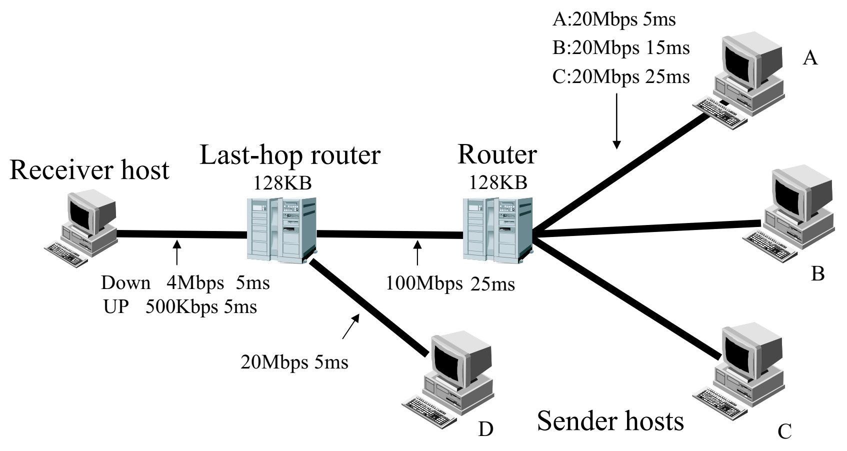


Figure 4: Simulation Topology of Scenario–1

Table 1: Parameters in Spring and Mehra ([1, 2])

Spring ([1])		Mehra ([2])	
access link bandwidth	4 Mbps	priority	0
queue length (to avoid the congestion)	64 KBytes	minimal rate	0
queue length (to reduce the delay)	5 KBytes	weight	0
threshold (interactive–short)	2KBytes		
threshold (short–long)	8KBytes		

that the parameters in Table 1 are the values recommended in the papers [1, 2] and there is almost no difference in parameter selection caused by the changes in network topology and/or simulation scenario.

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.

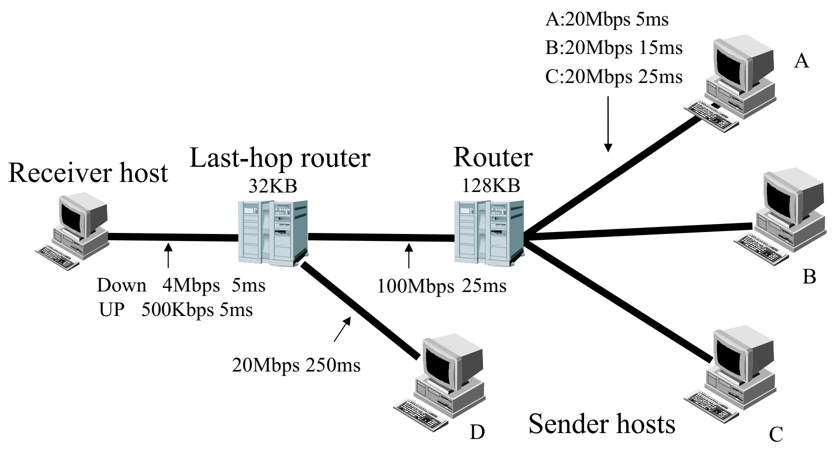


Figure 5: Simulation Topology of Scenario–2

4.1.2 Scenario–2 : Large Differences in RTT and Small Buffers

Figure 5 depicts the simulation model of Scenario–2. The topology is the same as that of Scenario–1, but we change the propagation delays between the sender host D and the receiver host to 255 msec, and the router buffer size to 32 KBytes. All other parameters and settings are the same as those of Scenario–1. 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 to say, 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.

4.1.3 Scenario–3 : A Large Number of Long–Lived Connections

Scenario–3 is performed with the network topology in Figure 6. From Scenario–1, we change the number of long–lived TCP connections from three to six. The propagation delays between the sender hosts and the receiver host are: A : 35 msec, B : 45 msec, C : 55 msec, D : 65 msec, E : 75 msec, F : 85 msec and G : 95 msec, respectively. All other parameters and settings are identical

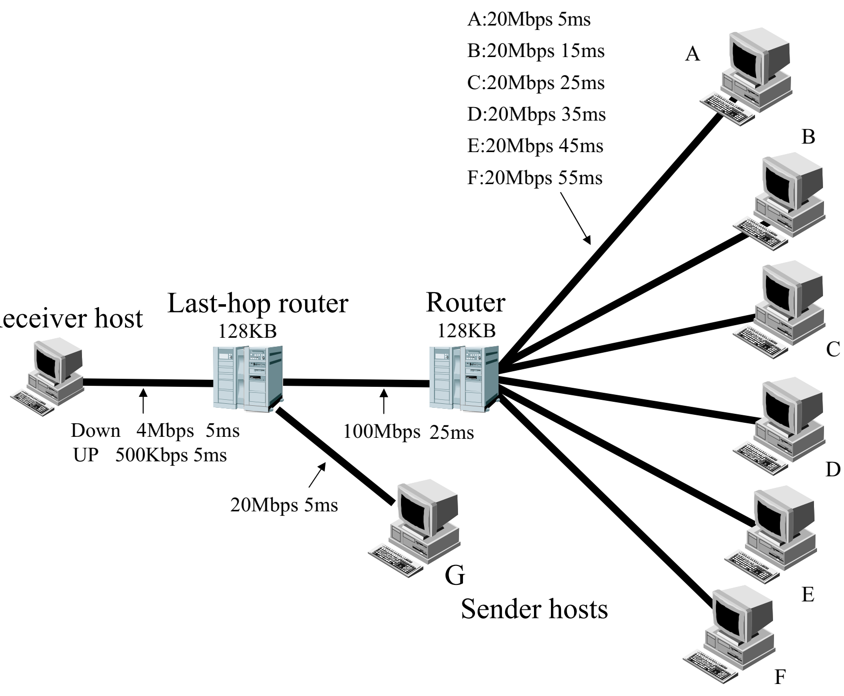


Figure 6: Simulation Topology of Scenario–3

to those of Scenario–1. In this scenario, 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.

4.1.4 Scenario–4 : Existence of UDP Streaming at the Access Link

In this scenario we use the simulation topology in Figure 7. In this simulation, the bulk data transfer is performed from the sender hosts A through C to the receiver host (long–lived connections), and 30 short–lived connections, each of which transfers 30 KBytes data, start being activated from the sender host D at 400 seconds with random intervals (5 sec average). In addition, UDP traffic is added from the sender host E to the receiver host. The rate of UDP traffic is changed as follows: 1 Mbps in 50–150 sec and 500–600 sec, 2 Mbps in 150–250 sec and 600–700 sec, and 3 Mbps in 250–350 sec and 700–800 sec. All other parameters and settings are the same as those

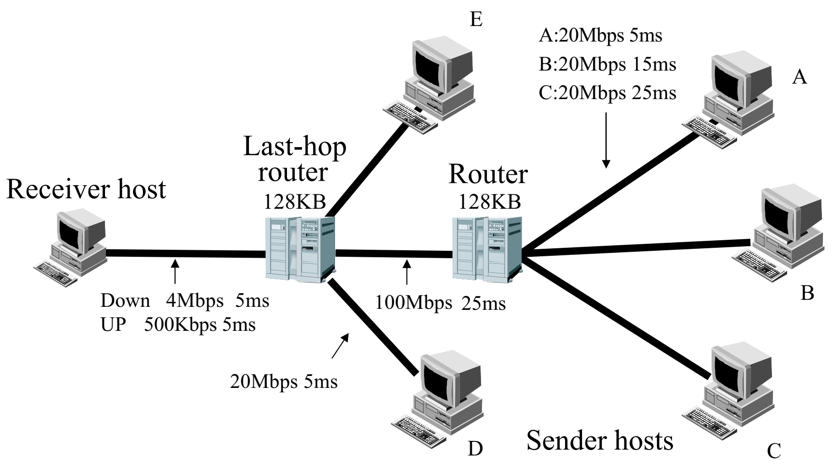


Figure 7: Simulation Topology of Scenario–4

of Scenario–1. In this scenario, we assume the receiver host enjoys the UDP video streaming simultaneously. It changes the bandwidth available for TCP connections at the receiver host, which may affect the performance of the proposed scheme, Spring, and Mehra.

4.2 Simulation Results

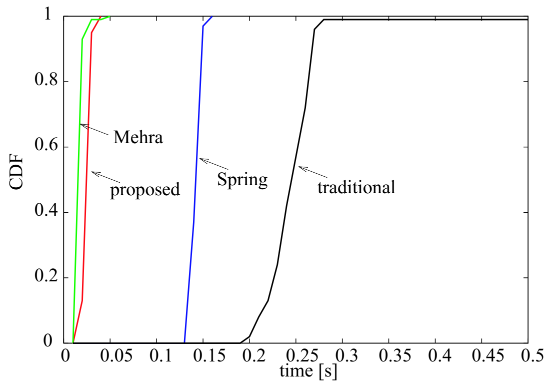
4.2.1 Scenario–1 : The Ideal Case

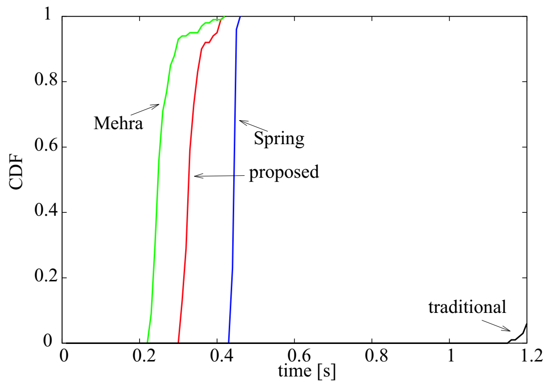
Figure 8 shows the CDFs (cumulative relative frequencies) of connection establishment time and data transfer time for short–lived connections, the change of utilization of the access link during 500 seconds simulation time, and the change of the average queue length of the router buffer. In this figure, our proposed scheme is labeled as "proposed," the scheme of [1] as "Spring," the scheme of [2] as "Mehra," and the standard TCP as "traditional." From Figures 8(a) and 8(b), we can 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 8(c), 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 8(a) and 8(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 8(c). 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 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 8(d) also confirms the under–utilization of the access link.

From Figure 8(d), 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 8(a) and 8(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 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.

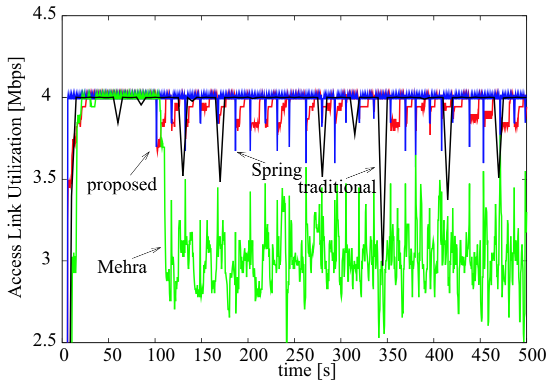
Figure 9 shows the change of the throughput of the long–lived connections in the simulation time. In this figure, we label the throughput of the connection from the sender host A as “flow A,” that from the sender host B as “flow B” and that from the sender host C as “flow C,” respectively. The label of “best” represents the throughput value when the access link bandwidth is shared most fairly and effectively. From Figure 9, 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

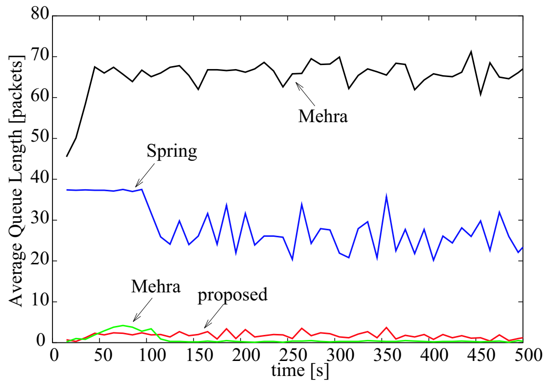




(a) Short–Lived Connections Establishment Time

(b) Data Transfer Time of Short–Lived Connections





(c) Access Link Utilization

(d) Average Queue Length

Figure 8: Simulation Results of Scenario–1 (1)

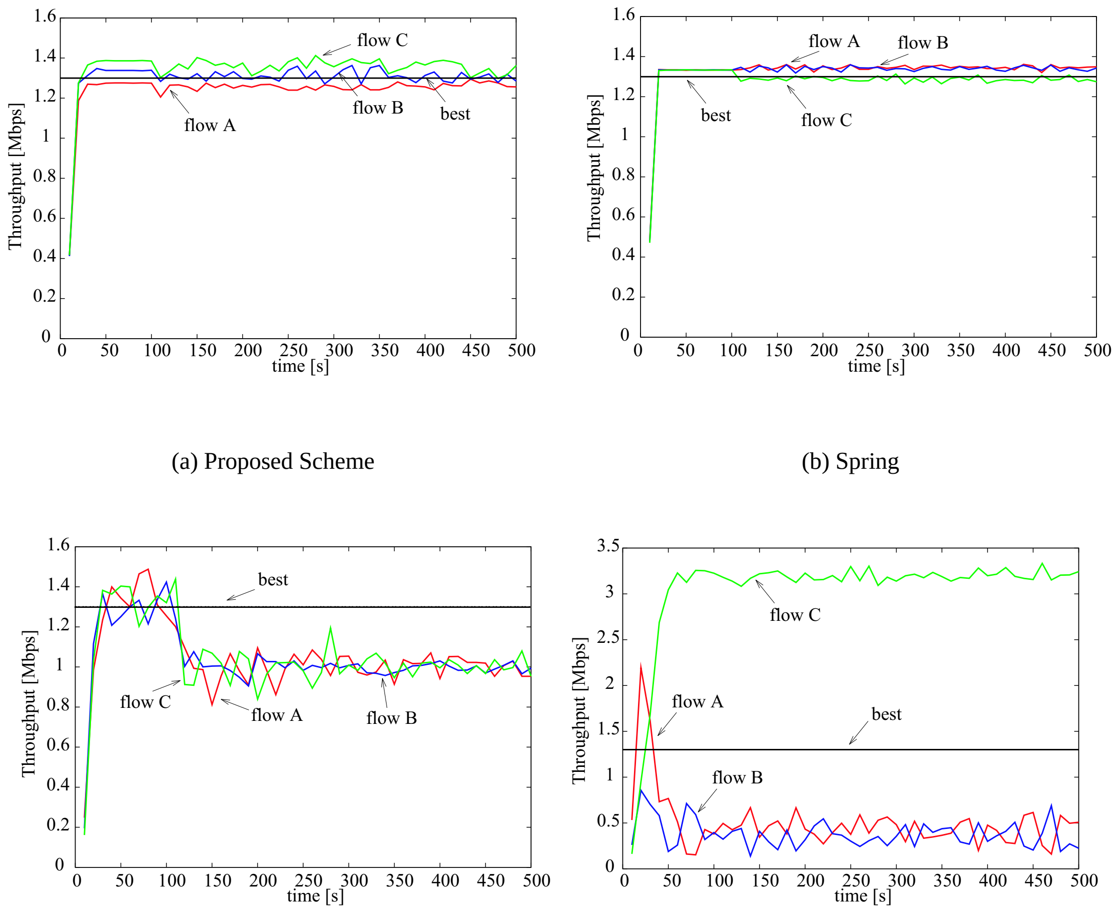
repeats the adjustment to make all TCP connections share the access link bandwidth equally. On the other hand, as we can see from Figure 9(a) and Figure 9(b), our proposed scheme and Spring show almost the same throughputs as the “best” case.

4.2.2 Scenario–2 : Large Differences in RTTs and Small Buffers

Figure 10 shows the simulation results of Scenario–2, as shown in Figure 8 for Scenario–1. From Figure 10(c), 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 becomes also 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 10(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 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

Figure 11 shows the CDFs of the connection establishment and data transfer times for short–lived connections, the fairness index [23] among the throughput of the six long–lived connections, and the change of the average queue length of the router buffer. In this scenario, since the conges-



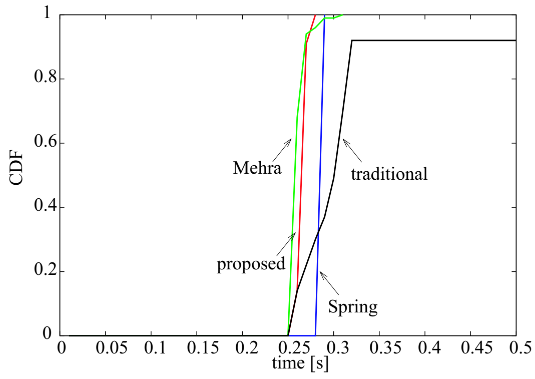
(a) Proposed Scheme

(b) Spring

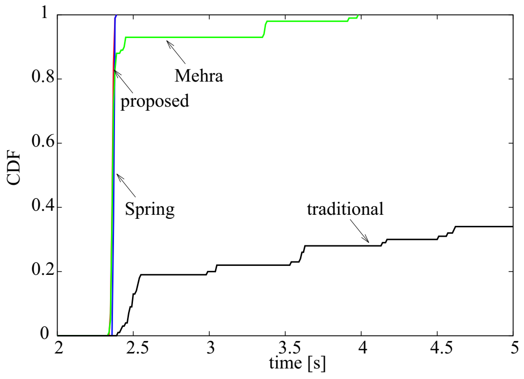
(c) Mehra

(d) Traditional Scheme

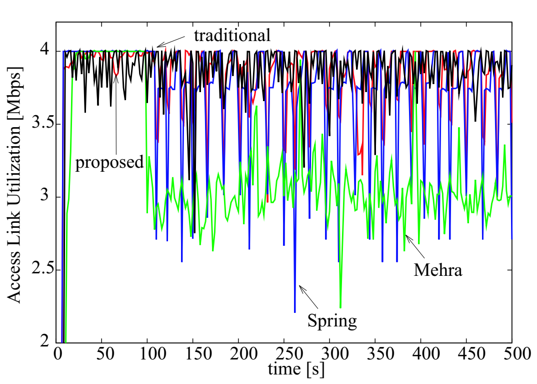
Figure 9: Simulation Results of Scenario–1 (2)



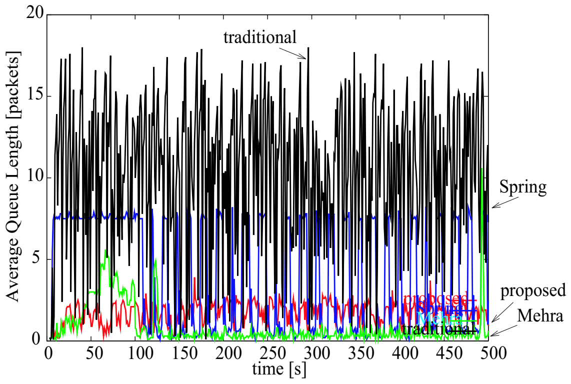
(a) Short–Lived Connections Establishment Time



(b) Data Transfer Time of Short–Lived Connections



(c) Access Link Utilization



(d) Average Queue Length

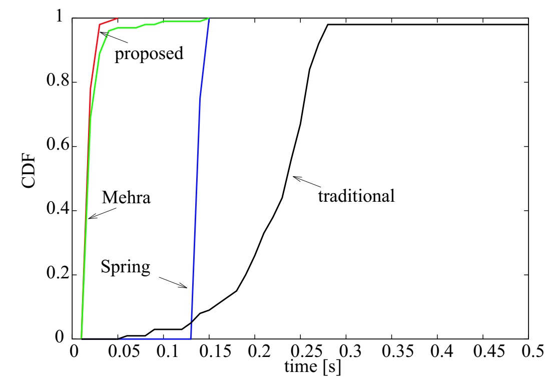
Figure 10: Simulation Results of Scenario–2

tion occurs more frequently at the access link, Mehra shows the larger average queue length than that in Scenario–1 as shown in Figure 11(d). This brings the increase of the data transfer times for short–lived connections in Mehra as presented in Figure 11(b). Moreover, from Figure 11(c), Mehra shows the 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 on the congestion level of the access network link.

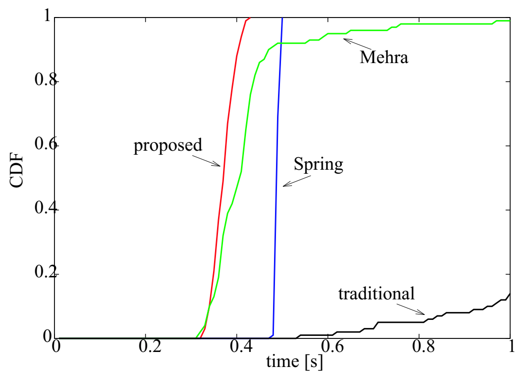
4.2.4 Scenario–4 : Existence of UDP Streaming at the Access Link

Figure 12 shows 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.

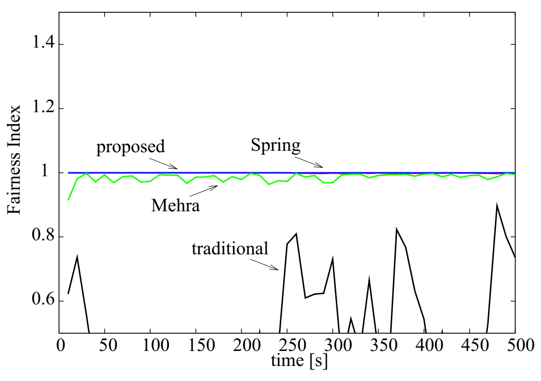
From Figure 12(d), Spring shows the high packet dropping rate. This is because the change of the access link bandwidth which TCP connections can use is not considered. That is to say, even if the available access link bandwidth for TCP connections changes caused by the UDP flow in this scenario, Spring cannot deal with the change since it controls the throughput of TCP connections according only to the parameters shown in Table 1. On the other hand, Mehra also shows the high packet loss rate. Since Mehra shows the lowest utilization of the access link after 400 seconds as shown in Figure 12(c), these packet losses are considered to occur when only long–lived connections are active from 0 to 400 seconds. 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. As



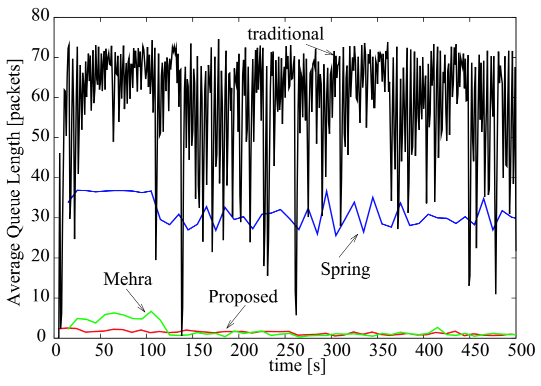
(a) Short–Lived Connections Establishment Time



(b) Data Transfer Time of Short–Lived Connections



(c) Fairness among Long–Lived Connections



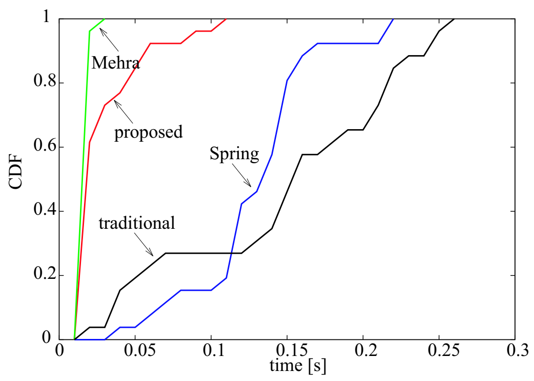
(d) Average Queue Length

Figure 11: Simulation Results of Scenario–3

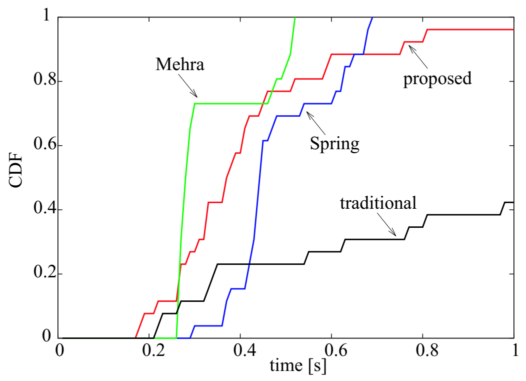
a result, Mehra over–estimates the available access link bandwidth for the TCP connections and the packet losses occur.

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 adjusts the amount of the receive socket buffer for all TCP connections to avoid congestion.

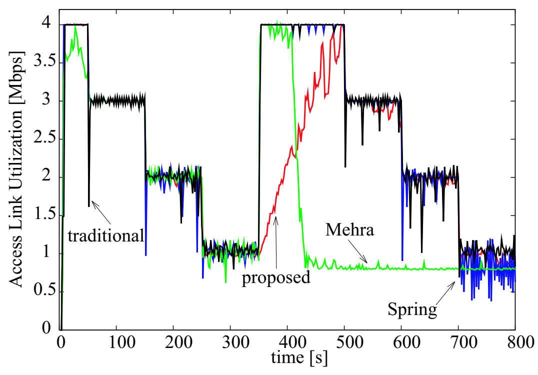
However, we observe that our proposed scheme indicates an under–utilization of the available bandwidth from 350 to 500 seconds, as shown in Figure 12(c). This is because our proposed scheme slowly increases its utilization of the access link bandwidth when there is an unused bandwidth. Consequently, 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.



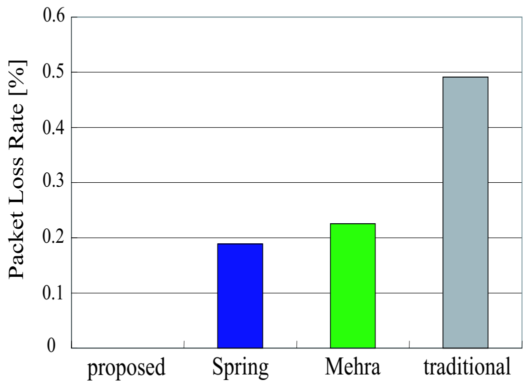
(a) Short–Lived Connections Establishment Time



(b) Data Transfer Time of Short–Lived Connections



(c) Access Link Utilization



(d) Packet Loss Rate

Figure 12: Simulation Results of Scenario–4

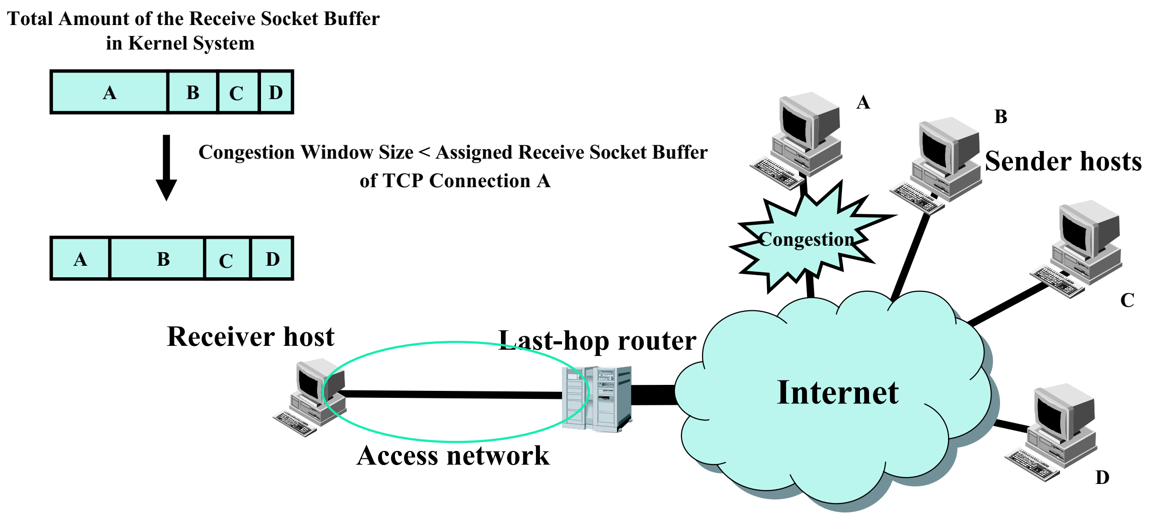
5 Integration with Endhost Resource Management Scheme

In our previous work, we have proposed the endhost resource management scheme for TCP connections [4]. The objectives of this scheme are to maintain the TCP connection's resource at the endhosts, such as a Web/Web proxy server to which many connections are established, especially when the amount of the endhost resources is insufficient. One of the features of the endhost resource management scheme is that the TCP receiver host monitors the utilization of the receive socket buffer and decreases the assigned size when it determines that an excessive amount of the buffer is assigned. By this mechanism, we can avoid the waste of the receive socket buffer, since the excess buffer is reassigned to connections that require a larger size of receive socket buffer. However, this scheme cannot deal with cases where a bottleneck exists at the access network link, not at the endhosts. On the other hand, our scheme as proposed in Section 3 cannot deal with cases where a bottleneck exists at the endhost. Therefore, in this section, we discuss the integration of the access link resource management scheme and the endhost resource management scheme.

We consider it is effective for our integrated system to be implemented to the endhosts when the bottleneck resources changes according to the network dynamics. For example, in P2P networks [12–14], it is known that some peers may have a large number of neighbor peers [24]. In these networks, even if the access link bandwidth connected to the endhost is sufficient, the amount of the endhost resources may be a bottleneck. Furthermore, since the topology of P2P network changes dynamically due to frequent peer join/leave, the bottleneck resources may change dynamically. To solve this kind of problem, our integrated system in this section estimates the bottleneck resources dynamically and maintains the bottleneck resources effectively.

5.1 Proposed Scheme

We consider the total amount of the receive socket buffer in the kernel system to be the endhost resource. The proposed scheme first estimates the amount of the receive socket buffer required



Total Amount of the Receive Socket Buffer in Kernel System
A
B
C
D
Congestion Window Size < Assigned Receive Socket Buffer of TCP Connection A
A
B
C
D
Receiver host
Last-hop router
Access network
Congestion
Internet
Sender hosts
A
B
C
D

Figure 13: Image of the Integrated System

to utilize effectively the access link bandwidth according to the access link resource management scheme proposed in Section 3. Then, the scheme compares the estimated amount of receive socket buffer with the total amount of receive socket buffer in the receiver host. If the total amount in the receiver host is less than the estimated amount, the assigned size of receive socket buffer to TCP connections is simply decreased according to the deficiency. Then each TCP connection is assigned the receive socket buffer according to the algorithm in Subsection 3.2.

While the TCP connection is active, the endhost resource management scheme proposed in [4] estimates the sender host's congestion window size of the connection by monitoring the utilization of receive socket buffer and calculates the required size of the receive socket buffer for each TCP connection. When the calculated sizes is less than the size assigned to the connection, the receive socket buffer for the connection is reduced and the excess buffer is reassigned to other connections whose calculated sizes are larger than the assigned sizes. This simple integration mechanism can help improve the utilization of the access link bandwidth even when a bottleneck does not exist at the access link. Figure 13 shows the image of the proposed scheme.

5.2 Evaluation Results

In this subsection, we confirm the effectiveness of the proposed scheme in Subsection 5.1 through the simulation experiments.

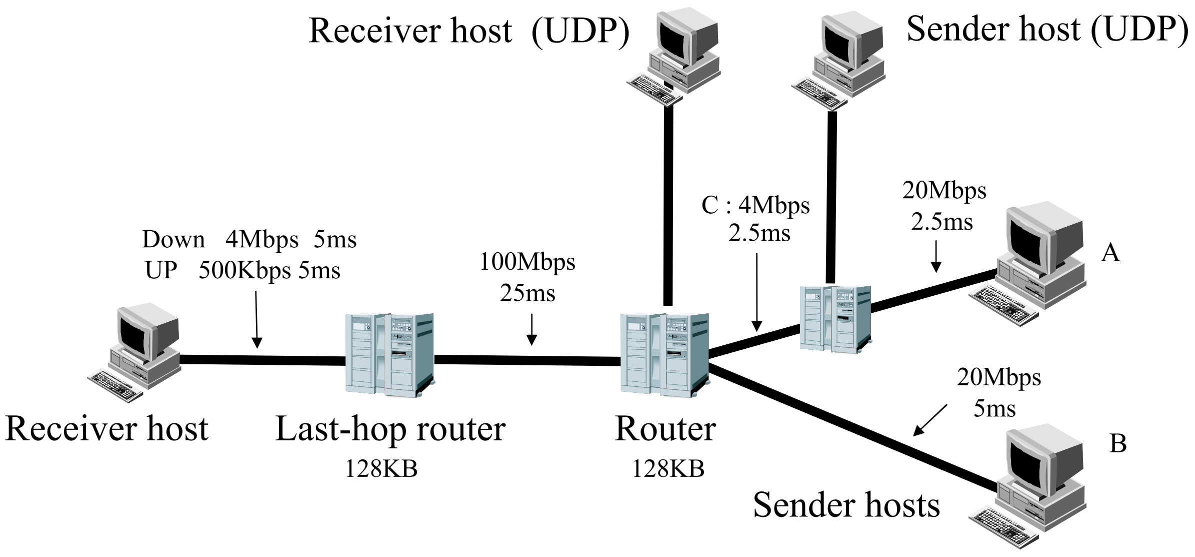
5.2.1 Scenario–5 : A Bottleneck Exists at the Backbone Network

Figure 14(a) shows the simulation model of Scenario–5. It consists of two sender hosts, A and B, and one receiver host. There are two additional hosts to inject an UDP traffic into link C. The propagation delays between the sender hosts and the receiver host are: A : 35 msec and B : 35 msec, respectively. The proposed scheme is implemented at the receiver host. In this simulation, TCP bulk data transfers are performed from the sender hosts A and B to the receiver host. Furthermore, a 3.5 Mbps UDP traffic is generated at the link C from 100 to 500 seconds. Consequently, a bottleneck for host A during this period moves from the access link to link C.

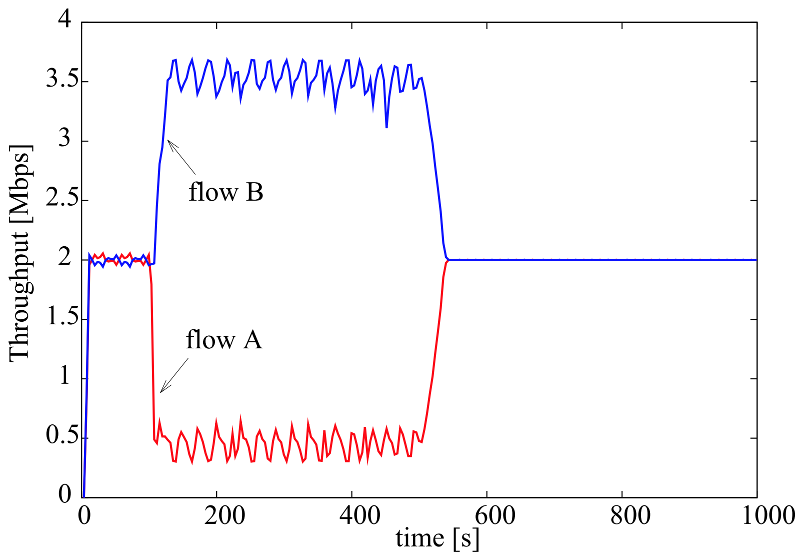
Figure 14(b) shows the change of the throughput of each TCP connection. In this figure, we represent the throughput of the connection between the sender host A and the receiver host as "flow A," and that between the sender host B and the receiver host as "flow B." From this figure, we observe that when the throughput of flow A decreases to 500 Kbps because of UDP traffic, the throughput of flow B increases successfully. This means that the access link resources can be effectively utilized. That is to say, since the receive socket buffer assigned to flow A by the access link resource management scheme is too large due to the decrease of flow A's throughput, the excess buffer is re–assigned to flow B by the endhost resource management scheme.

5.2.2 Scenario–6 : A Bottleneck Exists at the Endhost Resource

Figure 15(a) shows the simulation model of Scenario–6. It consists of 20 sender hosts and one receiver host. There are three additional hosts to inject UDP traffic into link E or link F. The sender hosts are divided into four groups, A through D, each of which has five sender hosts. The sender



(a) Simulation Topology of Scenario–5

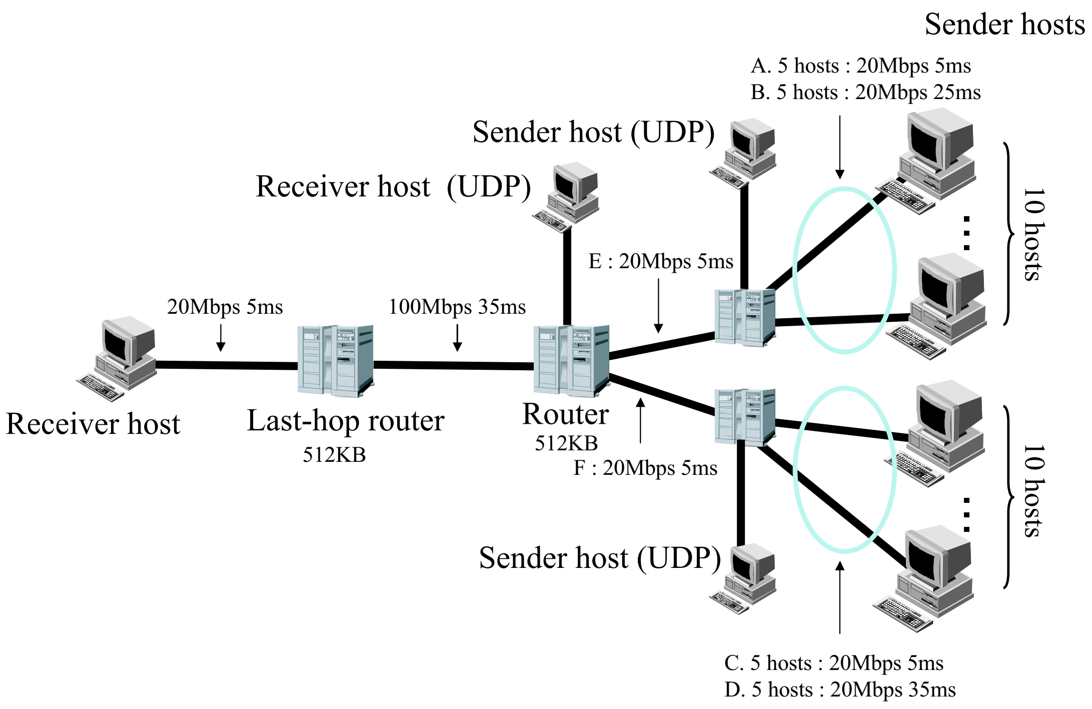


(b) Throughput of each TCP connection

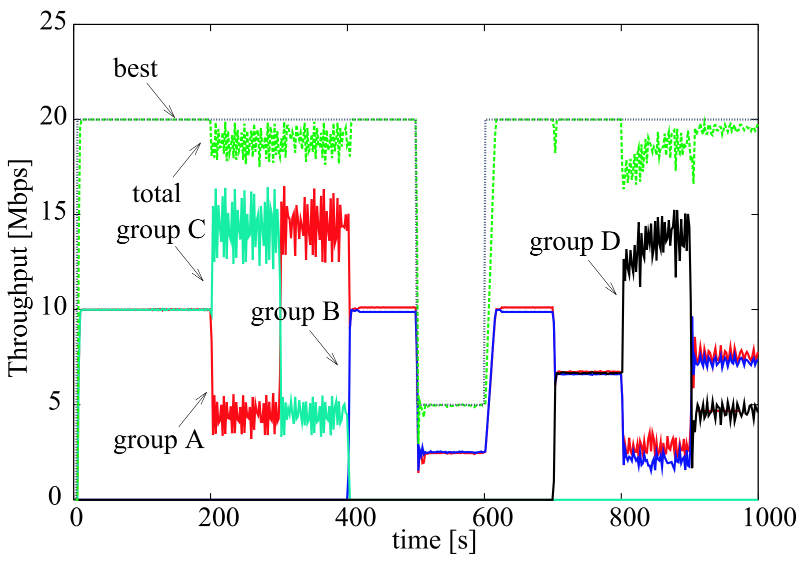
Figure 14: Simulation Topology and Results of Scenario–5

hosts in each group are connected to the links with the same propagation delays and bandwidths. The propagation delays between the sender host groups and the receiver host are: A : 50 msec, B : 70 msec, C : 50 msec and D : 80 msec. The access link bandwidth is set to 20 Mbps. The proposed scheme is implemented at the receiver host. In this simulation, the bulk data transfer is performed from the sender groups A to the receiver host from 0 to 1,000 seconds, from the sender groups B from 400 to 1,000 seconds, from the sender groups C from 0 to 400 seconds, and from the sender groups D from 700 to 1000 seconds. Furthermore, the 15 Mbps UDP traffic is generated at link E from 200 to 300 seconds, from 500 to 600 seconds and from 800 to 900 seconds, and at link F from 300 to 400 seconds and from 900 to 1,000 seconds. Consequently, a bottleneck changes frequently in the simulation.

Figure 15(b) shows the change of the total throughput of each sender group. In this figure, we label the aggregated throughput of flows between each sender group and the receiver host as “group A,” “group B,” “group C,” and “group D,” respectively. The label of “total” represents the total throughput of all groups, and that of “best” represents the ideal utilization of the access link bandwidth. From this figure, we observe that from 200 to 300 seconds, the throughput of group A decreases to 5 Mbps because of the UDP traffic, and the throughput of group C increases. The throughput of group C decreases to 5 Mbps and the throughput of group A increases from 300 to 400 seconds. After group B’s joining and group C’s leaving, the throughput of group A and group B decreases to about 2.5 Mbps from 500 to 600 seconds due to the UDP traffic of link E and F. After group D’s joining, the access link bandwidth is assigned to flows of each group as expected. These changes are caused by the integrated system proposed in the previous subsection. For example, from 200 to 300 seconds, a bottleneck for group A moves to link E. Since the receive socket buffer assigned to group A is then too large, the excess buffer is reassigned to group C by the endhost resource management scheme. On the other hand, from 800 to 900 seconds, whereas the number of the activated TCP connections is large, 15, the access link bandwidth is effectively utilized by the endhost resources management scheme. Note that the total throughput of TCP connections at the receiver host is close to the “best” line. It means that our scheme successfully changes the receive socket buffer for each TCP connection according to the number of TCP connections and the fluctuation of the UDP traffic volume on links E and F.



(a) Simulation Topology of Scenario–6



(b) Throughput of each Group

Figure 15: Simulation Topology and Results of Scenario–6

6 Concluding Remarks

In this thesis, we have proposed a receiver-based access link resource management scheme at the receiver host. Our proposed scheme adjusts virtually 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 be treated with high priority, and for a long-lived connection, the upper-layer application's QoS and the user's demands be reflected. We have evaluated the performance of the our proposed scheme through extensive simulation experiments, and confirmed that it can utilize effectively 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 [1, 2] 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 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 to the actual receiver host, and to evaluate it through experiments using the actual network.

Acknowledgements

I would like to express my gratitude to my supervisor, Prof. Hideo Miyahara of Osaka University, for his encouragement and valuable comments.

I would like to express my sincere appreciation to Prof. Masayuki Murata for his extensive help and continuous support through my studies and preparation of this thesis.

I am most grateful to Associate Prof. Go Hasegawa who has always given me appropriate guidance and invaluable direct advice.

I am also indebted to Associate Profs. Naoki Wakamiya of Osaka University who gave me helpful comments and feedback.

I also deeply thank Research Assistants Shin ’ichi Arakawa, and Ichinoshin Maki for their invaluable support and advice.

Finally, I heartily thank my friends and colleagues in the Department of Informatics and Mathematical Science of Osaka University for their support.

References

[1] N. T. Spring, M. Chesire, M. Berryman, V. Sahasranaman, T. Anderson, and B. N. Bershad, "Receiver based management of low bandwidth access links," in *Proccedings of IEEE INFOCOM 2000*, pp. 245–254, 2000.

[2] P. Mehra, A. Zakhor, and C. D. Vleeschouwer, "Receiver-driven bandwidth sharing for TCP," in *Proceedings of IEEE INFOCOM 2003*, Mar. 2003.

[3] Proxy Survey, available at `http://www.delegate.org/survey/proxy.cgi`.

[4] T. Okamoto, T. Terai, G. Hasegawa, and M. Murata, "A resource/connection management scheme for HTTP proxy servers," in *Proceedings of Second International IFIP-TC6 Networking Conference*, pp. 252–263, May 2002.

[5] Broadband networking report in japanese, available at `http://www.musen-lan.com/speed/htmldata/`.

[6] L. Guo and I. Matta, "The war between mice and elephants," in *Proccedings of the 9th IEEE International Conference on Network Protocols*, no. 2001-005, Nov. 2001.

[7] W3C Recommendations Reduce 'World Wide Wait', available at `http://www.w3.org/Protocols/NL-PrefNote.html`.

[8] H. Balakrishnan, H. S. Rahul, and S. Seshan, "An integrated congestion management architecture for Internet hosts," in *SIGCOMM 1999*, pp. 175–187, Sept. 1999.

[9] J. Touch, "TCP control block interdependence," *Request for Comments (RFC) 2140*, Apr. 1997.

[10] J. Postel, "Transmission control protocol (TCP)," *Request for Comments (RFC) 793*, Sept. 1981.

[11] W. R. Stevens, *TCP/IP Illustrated, Volume 1: The Protocols*. Reading, Massachusetts: Addison-Wesley, 1994.

[12] D. Milojicic, V. Kalogeraki, R. Lukose, K. Nagaraja, J. Pruyne, B. Richard, S. Rollins, and Z. Xu, "Peer-to-Peer Computing," Tech. Rep. HPL-2002-57, HP Laboratories, Mar. 2002.

[13] Gnutella, available at `http://www.gnutella.com/`.

[14] KaZaA, available at `http://www.kazaa.com/`.

[15] P.-H. Hsiao, H. Kung, and K.-S. Tan, "Active delay control for TCP," in *Proccedings of IEEE Globecon '01*, Sept. 2001.

[16] M. Hartling, M. Claypool, and R. Kinicki, "Active queue management for Web traffic," Tech. Rep. WPI-CS-TR-02-20, Computer Science Department, Worcester Polytechnic Institute, May 2002.

[17] S. Karandikar, S. Kalyanaraman, P. Bagal, and B. Packer, "TCP rate control," *ACM Computer Communications Review*, 2000.

[18] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Transactions on Networking*, vol. 1, pp. 397–413, Aug. 1993.

[19] S. Floyd and V. Jacobson, "Link-sharing and resource management models for packet networks," *IEEE/ACM Transactions on Networking*, vol. 3, pp. 365–386, Aug. 1995.

[20] D. D. Clark, "Window and acknowledgement strategy in TCP," *Request for Comments (RFC) 813*, July 1982.

[21] G. R. Wright and W. R. Stevens, *TCP/IP Illustrated, Volume 2: The Implementation*. Reading, Massachusetts: Addison-Wesley, 1995.

[22] The VINT Project UCB/LBNL/VINT network simulator - ns (version 2), available at `http://www.isi.edu/nsnam/ns/`.

[23] R. Jain, A. Durresi, and G. Babic, "Throughput fairness index: An explanation," *ATM Forum Contribution: AF/99-0045*, Feb. 1999.

[24] L. A. Adamic, R. M. Lukose, A. R. Puniyani, and B. A. Huberman, "Search in power-law networks," *Physical Review E 64 46135*, 2001.