Analysis and Improvement of Fairness among TCP Connections transmitting Differently Sized Data

Koichi Tokuda† Go Hasegawa‡ Masayuki Murata‡

†Graduate School of Engineering Science, Osaka University

1-3 Machikaneyama, Toyonaka, Osaka 560-0043, Japan

E-mail: kouichit@anarg.jp

‡Cybermedia Center, Osaka University

1-32 Machikaneyama, Toyonaka, Osaka 560-0043, Japan

Phone: +81-6-6850-6863, Fax: +81-6-6850-6868

E-mail: {hasegawa, murata}@anarg.jp

Abstract

Short-lived TCP connections, which transmit small-sized data, suffer from significant low throughput compared with long-lived TCP connections, which transmit large-sized data, because of the inherent nature of the ACK-based window flow control of TCP. In this paper, we focus on the unfairness problem caused by transmitting differently sized data. We first confirm the unfairness property through simple simulation results, and show the necessity of devising a method to improve the fairness property at the router buffer. As preparation for realizing this method, we next introduce a new analysis approach to estimate the throughput of a TCP connection when the packet loss probabilities of each transmitted packet are different. Our analysis gives a better estimation of the TCP throughput than the existing analyses. Using the analysis results, we propose a new method to improve the fairness property, and show that our proposed method can improve the fairness property without degradation of the network link utilization.

1 Introduction

Fair service among users is one of the most important goals for those who are concerned with the quality of best–effort traffic. It is becoming more important as the limit on the use of network resources is alleviated by broadband access technologies such as the cable modem, wireless and xDSL (Digital Subscriber Line) accesses. While much research has recently been carried out on the QoS guarantee/discrimination mechanisms by IntServ and DiffServ architectures, the fairness issue is often more important than those mechanisms. Even if the DiffServ architecture will be successfully deployed in the future network, fairness among users within a class is still important to be achieved.

TCP, the most popular transport-layer protocol in the current Internet, has an inevitable unfairness property, that is, TCP connections with different network environments (propagation delays, link bandwidths, congestion levels, and so on) achieve quite different throughputs. Furthermore, even when the TCP connections have the same network environment, there exists another unfairness among TCP connections that transmit differently sized data for the inherent nature of the ACK-based window flow control of TCP [1]. This unfairness is brought by the difference of TCP window sizes while transmitting the data. A TCP connection transmitting small-sized data (which is called *short-lived connections* hereafter) ends its transmission before it opens TCP window largely. On the other hand, the window size of a TCP connection transmitting large-sized data (*long-lived connection*) becomes sufficiently large since it takes many RTTs to transmit the data. As a result, when the short-lived connection and the long-lived connection share the same bottleneck link, the long-lived connection utilizes larger amount of the network bandwidth.

Furthermore, long-lived and short-lived TCP connections are affected differently by packet losses in the network. For the long-lived TCP connections, the packet losses are likely to be detected by duplicate ACKs and retransmitted by the fast retransmit algorithm [2], since the window size is sufficiently large. For short-lived TCP connections, on the other hand, the fast retransmit algorithm is not invoked due to their small window sizes, and they must wait for the expiration of the retransmission timer for a few seconds to retransmit the lost packets. It further degrades the fairness property between long-lived and short-lived TCP connections.

Therefore, in this paper, we tackle the unfairness problem caused by transmitting differently sized data, and propose a new method to achieve fairness. We first confirm the unfairness property through some simple simulation experiments. One possible way to overcome the unfairness problem is to count the number of packets

transmitted by each connection and change the packet discarding probabilities depending on the number of transmitted packets. That is, we set the packet discarding probabilities for packets from short-lived connections to low values and for packets from long-lived connections to high values to treat packets from short-lived TCP connections preferentially over those from long-lived TCP connections.

To derive appropriate packet discarding probabilities, it is necessary to estimate the throughput of a TCP connection when we change the packet discarding probabilities during its transmission. Therefore, we have developed a new analysis technique to estimate TCP throughput under such a situation. Through simulation experiments, we show that our method can yield results of higher accuracy than those obtained by the previously proposed analysis methods described in [3, 4]. To improve fairness, we propose our method, which counts the number of packets transmitted by each connection and changes the packet discarding probabilities, which is calculated by our analysis technique, depending on the number of transmitted packets.

The rest of this paper is organized as follows. In Section 2 we show some simulation results to confirm the unfairness property caused by transmitted data size. In Section 3 we propose a new analysis approach to estimate the throughput of a TCP connection when the packet loss probabilities for each transmitted packet are different. We explain our proposed method to improve fairness in Section 4, and show that our proposed method can improve fairness without degradation of the network utilization in Section 5. We next describe the simplification of our proposed method to reduce processing overheads in Section 6. Finally, we present concluding remarks and future works in Section 7.

2 Unfairness Caused by Transmitting Differently Sized Data

In this section, we confirm unfairness caused by transmitting differently sized data through simple simulation experiments. In the simulation, we use the simple network model depicted in Figure 1, where we identically set the propagation delays of the links between a router and sender/receiver hosts to 50 msec. The bandwidths of both links are 100 Mbps, which are large enough not to limit the throughput of TCP connections. The packet length is fixed at 1460 Bytes. The receive buffer at the receiver host and the maximum window size of TCP are also large enough not to limit the throughput of TCP. Although no packet losses occur at the router buffer, we intentionally introduce packet losses at the link between the sender host and the router. The packet loss ratio of the link is denoted by p. In the simulation, the sender host sends differently sized data by TCP, and we observe

the throughput of each data transmission.

Figure 2 shows the relationship between the size of transmitted data in packets and the average throughput of 1000-times transmissions of each sized data. From this figure, we can observe that if the size of transmitted data is small (short-lived TCP connection), the TCP connection suffers from very low throughput. It is because the short-lived TCP connection completes its data transmission before its congestion window becomes large, which is inevitable due to the inherent nature of the ACK-based window flow control of TCP. Furthermore, when packet losses occur, the short-lived TCP connection cannot detect the losses by duplicate ACK packets because its window size is too small. It brings TCP timeout, which results in serious performance degradation.

If the size of transmitted data is large (long-lived TCP connection), on the other hand, the TCP connection can obtain high throughput as shown in Figure 2. It is because the congestion window of the connection becomes sufficiently large during its data transmission, which enables it to utilize the link bandwidth effectively. Particularly when the number of transmitted packets is between 100 and 1000 in Figure 2, the obtained throughput is quite high. This is caused by the inflated window in the initial slow start phase of TCP. Note that since TCP doubles its window size in every RTT during the slow start phase, the window size increases very fast as no packet losses occur in the network. In addition, a large congestion window makes it possible to retransmit lost packets by fast retransmit and fast recovery algorithms without retransmission timeouts.

These simulation results clearly show the existence of significant unfairness in throughput between long-lived and short-lived TCP connections. It was reported in [5] that the average size of Web documents at several Web servers was under 10 KBytes. More importantly, Crovella and Bestavros [6] reported that the Web document size exhibits a heavy-tailed nature, meaning that very large documents exist with certain probabilities, but at the same time, small documents exist with large probabilities. That is, unfairness between the long-lived and short-lived TCP connections is serious on the Internet. One possible way to overcome this problem is to set the packet discarding probability for packets from long-lived TCP connections higher than that for packets from short-lived TCP connections. In the next section, we develop a new analysis method of estimating the TCP throughput to realize such a mechanism.

3 TCP Throughput Analysis

As confirmed in Section 2, there exists significant unfairness in throughput caused by transmitting differently sized data. One possible way to overcome this problem is to treat packets from short-lived TCP connections preferentially over those from long-lived TCP connections. It can be performed at the router buffer by explicitly discarding incoming packets appropriately as the router becomes congested. That is, for each connection, the discarding probabilities for the packets are first set to be low. Then, as the number of packets from the connection is increased, the packet discarding probabilities should be set to be higher. This is our key idea and we can expect that the occupation of link bandwidth by the long-lived connections is avoided by this method.

For realizing such a method, we must develop a new analysis to derive the throughput of TCP when packet discarding probabilities for each transmitted packet are different, which is the objective of this section. The analytic derivation of the throughput of a TCP connection has already been developed. See, e.g., [3, 4, 7]. In [3, 7], the authors assumed that packet discarding probability p is constant, and derived the steady state throughput of a bulk TCP flow. The authors in [4] extended the analytical model in [3] to consider the effects of the initial slow start phase of TCP, and investigated the data transmission delays more precisely. However, the assumption of a constant packet discarding probability is still made. We therefore cannot apply those analytical results directly to our approach, which attempts to improve fairness by dynamically changing the packet discarding probabilities for packets transmitted by each TCP connection. Moreover, the analysis presented in [3, 7] estimates the steady state throughput of TCP, which is another reason why we cannot adopt their analysis methods: we want to estimate the throughput values of the small-sized data under 10 KByte, which is a typical size of WWW documents [5].

In what follows, we will describe the development of an analysis method to estimate the throughput of the TCP connection transmitting arbitrary sized data when packet loss probabilities are changed dynamically.

3.1 Assumptions

In our analysis, we make the following assumptions. The sender host uses the congestion control algorithm of TCP Reno [2]. ACK packets are never lost in the network. The time needed to transmit all packets within a congestion window is less than the round trip time of the TCP connection. When a packet loss occurs during packet transmission in a window, all of the remaining packets in the window are also lost. The packet discarding probabilities are independent of the window size. For treating the model in which packet discarding probabilities

for transmitted packets are different, we denote the packet discarding probability for the i-th packet by p_i.

We further assume that the TCP receiver hosts send back an ACK packet every b packets , and the TCP sender hosts increase their congestion window during the slow start phase by δ on receiving an ACK packet. The initial congestion window size of the TCP connection is w_1. Note that the regular TCP uses $\delta = 1$ and $w_1 = 1$. We will obtain the throughput in TCP data transmission of d packets by dividing the analysis into two parts: one is for the initial slow start phase, and the other is the following congestion avoidance phase. We show the details of our analysis in the following subsections.

3.2 Analysis

The analysis method during the initial slow start phase follows that in [4]. Hence, we mainly describe the difference between our analysis and that in [4], particularly the difference of the packet discarding probabilities.

We define d_{ss} as the number of packets transmitted during the initial slow start phase and $E[r]$ as the average round trip time. Since the initial slow start phase continues until the first packet loss occurs, we can derive $E[d_{ss}]$, the expectation of d_{ss}, as follows:

$$\begin{aligned} P[d_{ss} = k] &= \prod_{l=1}^{k-1}(1-p_l)\cdot p_k \\ E[d_{ss}] &= \sum_{k=1}^{d} P[d_{ss}=k]\cdot k + \prod_{k=1}^{d}(1-p_k)\cdot d \end{aligned} \quad (1)$$

Since the TCP sender host increases its congestion window by δ packets on receiving each ACK packet, we have

$$w_{n+1} = w_n + \frac{\delta}{b} w_n = \gamma w_n$$

where $\gamma = (1 + \frac{\delta}{b})$. We assume that it takes $n_{ss} \cdot E[r]$ to send $E[d_{ss}]$ packets during the slow start phase. Then,

$$\begin{aligned} E[d_{ss}] &= w_1 + \gamma w_1 + \gamma^2 w_1 + \cdots + \gamma^{n_{ss}-1} w_1 \\ &= w_1 \frac{\gamma^{n_{ss}} - 1}{\gamma - 1} \end{aligned} \quad (2)$$

From Equations (1) and (2), we can obtain n_{ss}. By considering the limitation presented by the maximum window size of the sender TCP, we can obtain the following equations for $E[T_{ss}]$, the time length of the initial slow start phase. For more details, refer to [4].

$$E[T_{ss}] = \begin{cases} E[r]\cdot[\log_\gamma \frac{W_{max}}{w_1} + 1 + \frac{1}{W_{max}}(E[d_{ss}] - \frac{\gamma W_{max} - w_1}{\gamma - 1})] & if\ E[W_{ss}] > W_{max} \\ E[r]\cdot[\log_\gamma(\frac{E[d_{ss}]\cdot(\gamma-1)}{w_i} + 1)] & if\ E[W_{ss}] \le W_{max} \end{cases}$$

Next, we describe our analysis during the congestion avoidance phase. We depict a typical evolution of the TCP window size at the sender host in Figure 3. In the congestion avoidance phase, the TCP sender host transmits as many packets as its congestion window size. By receiving ACK packets from the receiver host, the TCP sender host increases its congestion window and transmits new packets to the TCP receiver host. When packet losses occur, the TCP sender host detects them by duplicate ACKs or retransmission timeouts. It then retransmits the lost packets and decreases its congestion window. The TCP sender host repeats this cycle until all of the $(d - d_{ss})$ packets are successfully transmitted.

We define an *epoch* as a cycle from the previous packet loss to the next packet loss. The i-th epoch from the beginning of the TCP connection is referred to as epoch$_i$. In each epoch, *round* is defined as a time slice from the first transmission of packet in a window to the receipt of the ACK for the first transmitted packet. Especially, the j-th round in the i-th epoch is referred to as round$_{i,j}$. The number of rounds included in epoch$_i$ is denoted by X_i. Note that packet loss actually occurs in round$_{i,X_i-1}$, and the sender detects the packet loss in the next round (round$_{i,X_i}$) by receiving three duplicate ACKs or by waiting a retransmission timeout. The time length of epoch$_i$ is denoted by S_i. In S_i, the time duration in which the TCP sender host is sending packets is denoted by K_i, and the time duration that the sender waits for a retransmission timeout (called a *retransmission phase*) is Z_i. Therefore, $S_i = K_i + Z_i$. Further, M_i is defined as the number of packets transmitted in epoch$_i$. We further denote the number of packets transmitted in K_i and Z_i by Y_i and R_i, respectively. We also use Q_i as a probability that timeout occurs in epoch$_i$. Then, the following equations hold:

$$S_i = K_i + Z_i, \quad P[Z_i = 0] = 1 - Q_i, \quad P[Z_i > 0] = Q_i \tag{3}$$

$$M_i = Y_i + R_i, \quad P[R_i = 0] = 1 - Q_i, \quad P[R_i > 0] = Q_i \tag{4}$$

Equations (3) and (4) yield the following relations on the average values of S_i and M_i. Note that the suffix i is necessary because the packet discarding probabilities for each packet are different.

$$\begin{aligned} E[S_i] &= E[K_i] + Q_i \cdot E[Z_i] \\ E[M_i] &= E[Y_i] + Q_i \cdot E[R_i] \end{aligned}$$

Following [3], Q_i is obtained as follows:

$$Q_i = \begin{cases} 1 & w \leq 3 \\ \sum_{j=0}^{2} K_i(W_i, j) + \sum_{j=3}^{W_i} K_i(W_i, j) \sum_{m=0}^{2} C_i(k, m) & w > 3 \end{cases} \tag{5}$$

where W_i is the window size at the end of epoch$_i$, $A_i(W_i, k)$ is the probability that the TCP host can transmit successive k packets without packet loss in round$_{i,X_i-1}$ under the condition that at least one packet is lost in the round. $C_i(m, n)$ is the probability that initial n packets among m packets are successfully transmitted in round$_{i,X_i}$. Both $A_i(W_i, k)$ and $C_i(m, n)$ have been calculated in [3]. However, we extend the results in [3] to be applied to our analysis model where packet discarding probabilities are changed dynamically. Here, we denote the number of packets that the TCP can transmit from the beginning of the connection to epoch $_i$ by U_i, and V_i is defined as $U_{i-1} + Y_i$. We then obtain the following equations. For more details, refer to [3].

$$
\begin{aligned}
U_i &= d_{ss} + \sum_{j=1}^{i} M_j \\
V_i &= d_{ss} + \sum_{j=1}^{i-1} M_j + Y_i \\
A_i(W_i, k) &= \frac{\prod_{j=1}^{k}(1 - p_{j+V_i})}{1 - \prod_{j=1}^{W_i}(1 - p_{j+V_i})} \\
C_i(m, n) &= \begin{cases} \prod_{k=1}^{m}(1 - p_{V_i+k}) \cdot p_{V_i+k} & m \leq n-1 \\ \prod_{k=1}^{n}(1 - p_{V_i+k}) & m = n \end{cases}
\end{aligned}
$$

We then have n_{ca}, the number of epochs during the congestion avoidance phase necessary to finish the data transmission, which can be obtained by solving the following equation:

$$
d = E[d_{ss}] + \sum_{i=1}^{n_{ca}} E[M_i] \tag{6}
$$

From all of the above equations, we can calculate B, the throughput of TCP during the congestion avoidance phase, as follows:

$$
B = \frac{\sum_{i=1}^{n_{ca}}(E[Y_i] + Q_i \cdot E[R_i])}{\sum_{i=1}^{n_{ca}}(E[K_i] + Q_i \cdot E[Z_i])} \tag{7}
$$

In what follows, we calculate $E[Y_i]$, $E[R_i]$, $E[K_i]$, and $E[Z_i]$ to determine B. We denote the average value of the initial retransmission timeout by T_0. We want to derive $P[R_i = k]$, the probability that in the retransmission phase the TCP host fails to retransmit the lost packet $(k-1)$ times and finally transmits the k-th packet successfully. $P[R_i = k]$ and L_k, the time length of the retransmission phase, can be obtained as follows by considering the exponential backoff of the retransmission timeout [2]:

$$
P[R_i = k] = \prod_{l=1}^{k-1} p_{V_i+l} \cdot (1 - p_{V_i+k})
$$

$$
L_k = \begin{cases} (2^k - 1)T_0 & k \leq 6 \\ (63 + 64(k-6))T_0 & k > 7 \end{cases}
$$

$E[Z_i]$ and $E[R_i]$ are then given by,

$$E[Z_i] = \sum_{k=1}^{d-V_i} P[R_i = k] \cdot L_k + \prod_{k=1}^{d-V_i} p_{V_i+k} \cdot L_{d-V_i} \quad (8)$$

$$E[R_i] = \sum_{k=1}^{d-V_i} P[R_i = k] \cdot k + \prod_{k=1}^{d-V_i} p_{V_i+k} \cdot (d - V_i) \quad (9)$$

Figure 4 shows more detailed behavior of TCP in epoch$_i$. Referring to this figure, we derive $E[X_i]$ and $E[Y_i]$ as follows. When we define α_i as the number of packets that can be transmitted from the beginning of epoch$_i$ without packet loss in epoch$_i$, the probability of $\alpha_i = k$ is given by,

$$P[\alpha_i = k] = \prod_{l=1}^{k-1}(1 - p_{U_{i-1}+l}) \cdot p_{U_{i-1}+k}$$

Then, $E[\alpha_i]$ is derived as follows:

$$E[\alpha_i] = \sum_{k=1}^{d-U_{i-1}} P[\alpha_i = k] \cdot k + \prod_{k=1}^{d-U_{i-1}} (1 - p_{U_{i-1}+k}) \cdot d \quad (10)$$

Observing the behavior of TCP illustrated in Figure 4 leads to the following equation.

$$E[Y_i] = E[\alpha_i] + E[W_i] - 1 \quad (11)$$

In the congestion avoidance phase, the sender TCP host increases its congestion window by 1 packet every b rounds. Then we have the following equation for W_i.

$$W_i = \frac{W_{i-1}}{2} + \frac{X_i}{b} \quad (12)$$

Taking the expectation of both sides of Equation (12) yields

$$E[W_i] = \frac{E[W_{i-1}]}{2} + \frac{E[X_i]}{b} \quad (13)$$

We can also obtain Y_i by summing up the number of transmitted packets in each round$_{i,j}$ until packet loss occurs:

$$\begin{aligned} Y_i &= \sum_{k=0}^{\frac{X_i}{b}-1} (\frac{W_{i-1}}{2} + k)b + \frac{W_i}{2} \\ &= \frac{X_i W_{i-1}}{2} + \frac{X_i}{2}(\frac{X_i}{b} - 1) + \frac{W_i}{2} \\ &= \frac{X_i}{2}(\frac{W_{i-1}}{2} + W_i - 1) + \frac{W_i}{2} \end{aligned} \quad (14)$$

By taking the expectation of both sides of Equation (14), we have

$$E[Y_i] = \frac{E[X_i]}{2} \cdot (\frac{E[W_{i-1}]}{2} + E[W_i] - 1) + \frac{E[W_i]}{2} \quad (15)$$

By solving Equations (10), (13) and (15) for W_i, we obtain

$$E[W_i] = \frac{b+1+\sqrt{(b+1)^2-2b(-\frac{2}{b}E[W_{i-1}]^2+bE[W_{i-1}]-4E[\alpha_i]+4)}}{2b}$$

From these equations, we determine $E[X_i]$ and $E[K_i]$ as follows:

$$\begin{aligned} E[K_i] &= (E[X_i]+1)\cdot E[r] \\ E[X_i] &= b\cdot(E[W_i]-\frac{E[W_{i-1}]}{2}) \end{aligned} \tag{16}$$

Finally, by using Equations (5), (8), (9), (11), (16), we can calculate B from Equation (7).

3.3 Validation and Discussion

In this subsection, we verify the accuracy of the analysis given in the previous subsection by comparing the analysis results with simulation results. In the simulation, we use the same network model depicted in Section 2. The parameters such as bandwidths or propagation delays are also unchanged.

We first show the evaluation results when the packet discarding probability is constant for comparison purposes. Figure 5 shows the relationship between the size of transmitted data in packets and the throughput in the data transmission where we set the packet discarding probability for the link between the sender host and the router to 0.005 (Figure 5(a)) and 0.01 (Figure 5(b)), respectively. These figures include the results of simulation experiments, our analysis, and the analysis presented in [4]. From these figures, we can see that the analysis in [4] cannot estimate the throughput well, especially when the number of transmitted packets is between 100 and 1000. It is because the analysis in [4] assumes that the TCP host shifts its state to a steady state just after the end of the initial slow start phase. Therefore, it cannot capture the large window size after the slow start phase. On the other hand, it can be observed that our analysis estimates the throughput of TCP with higher accuracy because we model the detailed behavior of TCP after the initial slow start phase with large window size.

These figures also indicate that the throughput of TCP data transmission is quite high when the size of transmitted data is around 500 packets in Figure 5(a), and 200 packets in Figure 5(b). The reason is that we set the maximum window size to so large value that the throughput of the TCP is not limited by the maximum window size, and the TCP sender can sufficiently increase its congestion window during its initial slow start phase. Figure 6 shows the cases where the maximum window sizes are set to 32 and 64 packets as in the usual implementation. Here, the packet discarding probability is fixed at 0.01. We can find in this figure that the maximum

throughput decreases since the window size during the initial slow start phase is limited by the maximum window size. Note that in both cases, our analysis again gives a better estimation of the throughput than the analysis in [4].

We next evaluate the accuracy of our analysis when packet discarding probabilities for each transmitted packet are different, which is the main objective of our analysis. Figure 7 shows the relationship between the data size and the throughput in TCP data transmission where we set the packet discarding probability to 0 for the 1st through 20th transmitted packets, and to 0.03 for the 21st and later transmitted packets. From this figure, we can observe that our analysis again estimates the throughput of the TCP precisely even when the packet discarding probabilities are dynamically changed.

We present the analysis results in Figure 8 for the case in which we change the threshold value. At the given threshold, the packet loss probabilities are increased from 0 to 0.03. Note that the results labelled 'Threshold=0' in Figure 8 indicates that all packets are discarded with probability of 0.03.

It is shown in Figure 8 that by setting the threshold value larger, the throughput of the small-sized data transmission can be improved. That is, we can improve fairness between long-lived and short-lived TCP connections by setting the packet discarding probabilities for short-lived connections to 0, and that of long-lived connections to a higher value. In the next section we will describe our proposed method to improve fairness, which changes packet discarding probabilities appropriately by using our analysis developed in this section.

4 Proposed Method

One possible way to overcome the unfaimess problem caused by transmitted data size is to treat packets from short-lived connections preferentially over those from long-lived connections at the router buffer. The authors of [1] realized this prioritizing mechanism by applying RIO (RED In and Out) [8], which is used in DiffServ [9]. The proposed method in [1] works under the cooperation between edge routers at the entrance of the network and core routers in the network. The edge routers classify incoming packets into two categories, the packets from long-lived connections and those from short-lived connections. The core routers discard incoming packets according to the RIO algorithm having two control parameters of RED (Random Early Detection) [10]. One of the control parameters is set for the packets from short-lived connections so that the packet discarding probabilities are low. The other is set for those from long-lived connections to discard them with high probabilities. Although the authors show the effectiveness of the proposed method, it has difficulties in requiring the cooperation between the edge

routers and core routers, and in setting appropriate control parameters of RIO which has two control parameters of RED. Furthermore, it categorizes TCP connections into only two types, which are short-lived and long-lived connections, which cannot completely remove the unfairness among connections transmitting differently sized data.

On the other hand, our proposed method does not require the cooperation between edge routers and core routers, and gives no difficulty in parameter setting. Basically, our proposed method counts the number of packets transmitted by each connection by using hashing mechanism, and discards incoming packets at the probabilities which are calculated by the TCP throughput analysis described in Section 3 so that each connection obtains fair throughput, independent of the number of packets transmitted by each connection.

Hereafter, we first discuss the goal of our proposed method in Subsection 4.1, and describe our proposed method in detail in Subsection 4.2.

4.1 The Goal of Proposed Method

Since we aim to remove the unfairness problem caused by the transmitted data size, the ultimate goal of our proposed method is to satisfy the relation presented in Figure 9(a), which shows the relation between the transmitted data size and its throughput. This relation illustrates a completely fair property, that is, the throughput of the data transfer by TCP is perfectly independent of its size. As shown in the previous sections, however, we cannot realize this relation, especially when the data size is small, because of the essential nature of the TCP's congestion control algorithm. Therefore, we modify the goal of our proposed method as follows. Since the small data transfer cannot achieve so high throughput in Figure 9(a), the *ideal* throughput for such small data transfer is changed as shown in Figure 9(b). In Figure 9(b), the ideal throughput of the small data transfer is what can be achieved when no packet loss occurs during the data transfer. When the throughput with no packet loss exceeds the *target throughput* shown in Figure 9(b), the ideal throughput becomes the target throughput. Note that the ideal throughput in Figure 9(b) is normalized to packet per RTT (Round Trip Time), since we ignore unfairness of TCP connections caused by different RTTs.

4.2 Detailed Algorithm

In this subsection, we describe the proposed method which realizes the relation in Figure 9(b). To do that, we need to count the number of packets of each TCP connection passing through the router. We use a hashing mechanism to maintain the number of packets transmitted by each TCP connection. The hash key is the combination of source/destination IP addresses and port numbers. Each hash entry memorizes a counter for the number of packets and the time at which the last packet arrived at the router. When a packet of a certain TCP connection arrives, the memorized time and the arrival time are compared. If the difference is smaller than T_{update} sec, the corresponding counter value is incremented by 1. Otherwise, the counter value is reset to 0. That is, T_{update} is the parameter to determine whether the arriving packet belongs to the existing TCP connection or a newly established TCP connection.

After determining the number of packets which have been transmitted by the TCP connection, we discard the incoming packets with the probabilities calculated by using the analysis results in Section 3, to realize the throughput value shown in Figure 9(b). Seeing Figure 9(b), the target throughput needs to be first determined. We use the similar method to RED to calculate the target throughput. That is, every time a new packet arrives at the router, the average queue length q_{avg} is updated by Equation (17). In Equation (17), q is the current queue length and w_q is the weight value in averaging the current queue size and the previous value of the average queue length.

$$q_{avg} = w_q q + (1 - w_q) q_{avg} \tag{17}$$

RED determines the packet discarding probability by comparing the average queue length with two threshold values, min_{th} and max_{th} [10]. On the other hand, our proposed method determines the target throughput in a similar way. It compares the average queue length with min_{th} and max_{th} every time N packets arrive at the router, and regulate the target throughput as follows:

- When $q_{avg} < min_{th}$

  The router judges that the network is not so congested. Then, the target throughput is increased by the following equation:

  $$target \leftarrow \alpha \cdot target \tag{18}$$

  In Equation (18), α represents an increasing rate of the target throughput.

- When $min_{th} \leq q_{avg} < max_{th}$

  The target throughput is not changed.

- When $max_{th} < q_{avg}$

  The router judges that the network is congested. Then, the target throughput is decreased as follows:

$$target \leftarrow \beta \cdot target \tag{19}$$

  In Equation (19), β is a decreasing rate of the value of target throughput.

Now we have obtained the target throughput value in Figure 9(b), we next determine the packet discarding probabilities as a function of the number of transmitted packets. We define $p_{i,d}$ as the discarding probability for the i-th packet transmitted by the TCP connection which transmits d packets and $\rho_{i,d}$ as the average throughput when the connection finishes transmitting i packets. By using the TCP throughput analysis in Section 3, we can derive $\rho_{i,d}$ from the number of transmitted packets d and the packet discarding probabilities p_i $(i = 1, 2, \cdots, d)$ as follows:

$$\rho_{i,d} = f(d, p_{1,d}, p_{2,d}, \cdots, p_{d,d})$$

Where f is the function determining the throughput of a TCP connection, obtained from the analysis results in Section 3. We also determine the function $g(j)$ which determines the ideal throughput of the j packets' transfer, shown in Figure 9(b).

We first derive $p_{1,1}$, the packet discarding probability for the connection which transmits only one packet. It is determined so that the value of $\rho_{1,1}$ becomes $g(1)$, which is the goal of our proposed method described in Figure 9(b). That is, we determine $p_{1,1}$ so that the following equation is satisfied:

$$f(1, p_{1,1}) = g(1)$$

Next, we derive $p_{1,2}$ and $p_{2,2}$ for the connection transmitting two packets. After setting $p_{1,2}$ equal to $p_{1,1}$, we determine $p_{2,2}$ to satisfy the following equation:

$$f(2, p_{1,2}, p_{2,2}) = g(2)$$

By iterating this calculation, we can derive the packet discarding probabilities $p_{i,d}$ $(i = 1, 2, \cdots, d)$ after $p_{i,d-1}$ $(i = 1, 2, \cdots, d-1)$ are determined. That is, $p_{i,d}$ $(i = 1, 2, \cdots, d-1)$ are set equal to $p_{i,d-1}$ $(i = 1, 2, \cdots, d-1)$,

respectively. Then, $p_{d,d}$ is determined to satisfy the following equation by using the analysis results:

$$f(d, p_{1,d}, p_{2,d}, \cdots, p_{d,d}) = g(d)$$

Every time N packets arrive at the router, we recalculate the target throughput and the packet discarding probabilities according to the above algorithms. That is, we use the same packet discarding probabilities until the router buffer receives N packets and the recalculation is completed.

5 Evaluation

In this section, we evaluate the performance of our proposed method described in the previous section through simulation experiments from the viewpoint of fairness among TCP connections transmitting differently sized data. We use the network simulator ns [11] for the simulation.

For evaluation of fairness degree, the *fairness index* proposed in [12] has been used in many studies. Equation (20) shows the definition of the fairness index. x_i denotes sample data such as throughput or completion time and n denotes the number of sample data. The value of f $(0 \leq f \leq 1)$ indicates the degree of fairness among sample data x_i $(i = 1, 2, \cdots, n)$. Fairness among sample data x_i $(i = 1, 2, \cdots, n)$ is better as the value of f is close to 1.

$$f = \frac{(\sum_{i=1}^{n} x_i)^2}{n \sum_{i=1}^{n} x_i^2} \tag{20}$$

In this paper, we adopt the fairness index for fairness evaluation as follows. For sample data x_i, we use the throughput values of each file transfer normalized by the ideal throughput shown in Figure 9(b). We also normalize the throughput value by the RTT value of the TCP connection, to remove the effect of the RTT value on the TCP throughput. The normalized value is denoted by x_i $(= \frac{x_i'}{r_i})$. We adopt x_i as sample data of the fairness index f. Here, x_i' is the throughput value in packet per RTT of the TCP connection and r_i is the ideal throughput shown in Figure 9(b).

Figure 10 depicts the network topology used in this simulation. It consists of N sender hosts $(S_1, S_2, \cdots, S_N)$, N corresponding receiver hosts $(R_1, R_2, \cdots, R_N)$, two routers (router A and router B), and links connecting the hosts and the routers. The bandwidth and the propagation delay of the links between the hosts and the routers are all set to 100 Mbps and 5 msec, respectively. The bottleneck link between the two routers has 30 Mbps bandwidth and τ msec propagation delay. The buffer size of router A is 100 packets. In this simulation, the sender host S_i

transmits a certain-sized data to the corresponding receiver host R_i $(i = 1, 2, \cdots, N)$. The distribution of the transmitted data size follows the Web document size distribution reported in [13]. Table 1 shows the detailed distribution. We use TailDrop, RED (Random Early Detection), and the proposed method at the buffer of router A, and evaluate the fairness of the three methods. Table 2 and 3 show the control parameters of RED and the proposed method, respectively. The packet size is 512 Bytes.

Our proposed method requires the RTT value of each TCP connection to calculate the packet discarding probabilities by using the TCP throughput analysis described in Section 3. In this simulation, we assume that the RTT values of all TCP connections are identical and that they are known by the router in advance. That is, we set the RTT value for the calculation of the packet discarding probabilities (RTT_{an}) to the actual RTT value of the TCP connection (RTT_{ac}). However, in the actual network, the RTT value of TCP connections cannot be known at the router. Therefore, the accuracy of the analysis decreases when we set RTT_{an} to a value different from RTT_{ac}, which may result in the degradation of the fairness achieved by the proposed method. We investigate the effect of the difference between RTT_{an} and RTT_{ac} in the next section.

Figure 11 shows the relationship between transmitted data size and average throughput when we set τ, the propagation delay of the bottleneck link, to 1 msec (Figure 11(a)), 10 msec (Figure 11(b)), and 100 msec (Figure 11(c)), respectively. To investigate the performance of the proposed method with the same packet loss probability, we change the number of sender/receiver hosts N to 300 in Figure 11(a), 500 in Figure 11(b), and 1000 in Figure 11(c), respectively. In each graph, we plot the results of TailDrop (labelled "Taildrop"), RED ("RED") and the proposed method ("Our method"). We also show the ideal throughput ("Ideal"), which means that if the simulation results are closer to this line, better fairness is achieved. From these figures, we can observe that when the number of transmitted packets is around 1-10 packets, the throughput of our proposed method is higher than that of Taildrop and RED. It means that the proposed method raised the throughput of short-lived TCP connections. It is because our proposed method does not discard the packets from the short-lived connections, which has a lower throughput than the target throughput. We also find by comparing Figures 11(a), 11(b) and 11(c) that our proposed method can increase the throughput of short-lived TCP connections regardless of the RTT values. We can also find that in the TailDrop case the throughput increases as the size of transmitted data becomes large. In the RED case, the throughput inflates temporarily when the transmitted data size is around 50-100 packets. On the other hand, our proposed method can restrain the throughput inflation of the long-lived TCP connections. That is, only our proposed method can achieve the fairness between short and long-lived TCP connections. It

is because our proposed method discards the incoming packets with appropriate probability which is calculated by the TCP throughput analysis. Note that the slight difference of the throughputs between the proposed method and the ideal case in 1–10 packets of the proposed method is caused by queueing delays at the bottleneck router. We can identify this phenomenon by observing that if the propagation delay becomes large, the difference of the throughputs become small.

When the propagation delay becomes large (Figure 11(c)), however, the throughput of 30-100 packets transmission becomes temporarily large even in the case with the proposed method. It is because our proposed method uses a timer to determine whether an arriving packet belongs to the existing TCP connection or a newly established TCP connection. If the propagation delay becomes large, the packet from the existing TCP connection is often recognized as that from the newly established TCP connection. Then the packet is discarded with lower probability than the appropriate probability calculated by TCP throughput analysis.

Table 4 shows the fairness index values of each method. From this table, we can observe that the proposed method provides the best fairness among the three methods, regardless of the propagation delay τ. We further show the utilization of the bottleneck link interconnecting router A and router B in Table 5. We can say from the table that the link utilization of our proposed method is not degraded as compared with Taildrop and RED, by introducing the additional mechanism at the router.

Through those results, we can conclude that our proposed method improves the unfairness property caused by transmitted data size without degrading the link utilization.

6 Implementation Issues of Proposed Method

In this section, we introduce two issues of the proposed method for the calculation of the packet discarding probabilities. One is the difficulty in setting the RTT value for the calculation, and the other is the means to decrease the calculation overhead.

6.1 Setting of RTT for the analysis

Our proposed method calculates packet discarding probabilities by the TCP throughput analysis given in Section 3. The analysis needs the RTT value, RTT_{an}, to be set for the calculation. In the simulation results in the previous section, we assumed that all TCP connections had identical propagation delays for the round trip paths. In this

case, we set the RTT_{an} value equal to the propagation delays of the round trip paths of the TCP connections. In the actual networks, however, each TCP connection has a different RTT value. Furthermore, a router, which activates the proposed method, cannot be informed of the RTT values of TCP connections. Therefore, we need to investigate the sensitivity of the RTT_{an} value in the proposed method, which is the objective of this subsection. That is, we give the simulation results for the case when the RTT_{an} value is set to a value different from the actual RTT value of the TCP connection, RTT_{ac}.

Figure 10 depicts the network topology used in the simulation. In Figure 10, the propagation delay of the bottleneck link τ is set to 5 msec. The propagation delay of the link between each sender host and router A is varied from 10 msec to 105 msec in 5 msec steps. That is, propagation delays of TCP connections for the round trip paths are varied from 40 msec to 230 msec in 10 msec steps. The other parameters of the simulation environments are the same as those given in Section 4. We apply Taildrop, RED, and our proposed method to the buffer of router A. In the case of our proposed method, we set the RTT_{an} value to the minimum value of all TCP connections' propagation delays for the round trip paths (40 msec), the average of them (135 msec), and the maximum of them (230 msec).

The relationship between the fairness index and the propagation delay of each link between the sender hosts and bottleneck router A is shown in Figure 12. The figure includes the results of Taildrop (labelled "Taildrop"), RED ("RED"), our proposed method of which the RTT_{an} values are 40 msec ("Our method 40 [ms]"), 135 msec ("Our method 135 [ms]") and 230 msec ("Our method 230 [ms]"). From this figure, we can observe that the proposed method can improve fairness as compared with Taildrop and RED, regardless of the RTT_{an} value. It is because our proposed method discards incoming packets from short-lived connections at low probabilities and those from long-lived connections at high probabilities, regardless of the RTT_{an} value. Furthermore, we can say that our proposed method improves the unfairness property even when the RTT_{an} value is largely different from the RTT_{ac} value. This means that our proposed method has little sensitivity in setting the RTT_{an} value.

6.2 Reduction of Processing Overhead

Our proposed method improves fairness by changing the packet discarding probability for every packet that passes through the router buffer. We depict this situation in Figure 13(a). Although this method can achieve quite good fairness as shown in the previous section, it requires a large amount of processing overhead in calculating the discarding probability for every packet by using the TCP throughput analysis, and in setting that value to each

packet. Therefore, we introduce and evaluate the modified method, which reduces the processing overhead.

In the modified method, we change the packet discarding probability only at certain points (we call these points *thresholds* hereafter), instead of varying it for every packet. That is, the packet discarding probability between two thresholds remains unchanged, and it is changed when the number of packets exceeds the threshold values. Figure 13(b) shows the relationship between the number of transmitted packets and the packet discarding probabilities. In this figure, we define n as the number of thresholds and d_i $(i = 1, 2, \cdots, n)$ as threshold values. The packet discarding probability at the interval $[d_{i-1}, d_i]$ is defined as p_i $(i = 1, 2, \cdots, n)$. If a connection transmits more than M $(= d_n)$ packets, the packet discarding probability is fixedly set to p_n. The value of M is set to a sufficiently large value (1000 packets in the following evaluation results). By changing the packet discarding probabilities at certain thresholds, our proposed method need not (1) change the packet discarding probability for every packet and (2) calculate the packet discarding probability for every packet by using the TCP throughput analysis. Thus, we reduce the processing overhead in our proposed method.

We can derive the packet discarding probability p_i $(i = 1, 2, \cdots, n)$ at each interval $[d_{i-1}, d_i]$, in a similar way to that explained in Section 4. The remaining issue is how to determine the threshold values. We consider the following two methods. The first method is quite simple, where we equally divide the interval $[1, M]$ into n parts in the log scale. The threshold values d_i becomes as follows:

$$d_i = 10^{i\frac{\log_{10} M}{n}}$$

The reason for dividing in the log scale is that we need to change the packet discarding probabilities more frequently when the number of transmitted packets is small, since the throughput drastically changes in such regions, as shown in Section 2.

The other method is to determine the optimal setting of threshold values analytically. We omit the detailed algorithm to do that, since the problem in finding the best threshold values d_i $(i = 1, 2, \cdots, n)$ from the interval $[1, M]$ corresponds to the simple combinatorial optimization problem. Note that since the problem is NP-complete, we use the heuristic way to solve the problem, and derive suboptimal threshold values.

Figure 14 shows the simulation results in the same environments given in Section 4 when we apply our modified method to router A, which presents the relation between the number of thresholds and the modified fairness index. We show the analysis and simulation results of the two methods described above (the former method is labelled “log” and the latter “optimum”).

We can say from the figure that in both methods the fairness property improves as the number of thresholds becomes large. This is because when the number of thresholds is large we can change packet discarding probabilities frequently to achieve good fairness. It is important result that when the number of thresholds is larger than 3, fairness is sufficiently good. Therefore, we can conclude that 3 thresholds are enough to obtain good fairness in our proposed method. Moreover, when we compare the two methods used to determine the threshold values, we can observe that when the number of thresholds is small, fairness in the "optimum" method is better than in the "log" method. But, when the number of thresholds becomes large, fairness of both methods becomes almost identical. Therefore, we can conclude that the "optimum" method, which requires complex calculations to determine the threshold values, is not necessary.

7 Conclusion

In this paper, we have first confirmed the unfairness problem between long-lived and short-lived TCP connections, and shown that the short-lived connections that transmit small data suffer from lower throughput than the long-lived connections. We have also pointed out that it is necessary to treat packets from short-lived connections preferentially over those from long-lived connections. To realize that, we have developed the new analysis to derive the throughput of TCP when the packet discarding probabilities for each transmitted packet are different. By using our analysis, we have proposed our new method to improve fairness. Our evaluation results through simulation experiments have revealed that our proposed method can improve the fairness property without degradation of the network utilization.

References

[1] L. Guo and I. Matta, "The War Between Mice and Elephants," *Technical Report BU-CS-2001-005*, May 2001.

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

[3] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, "Modeling TCP Throughput: A Simple Model and its Empirical Validation," in *Proceedings of ACM SIGCOMM '98*, pp. 303–314, Sept. 1998.

[4] N. Cardwell, S. Savage, and T. Anderson, "Modeling TCP Latency," in *Proceedings of IEEE INFOCOM '00*, pp. 1742–1751, Mar. 2000.

[5] M. Nabe, M. Murata, and H. Miyahara, "Analysis and Modeling of World Wide Web Traffic for Capacity Dimensioning of Internet Access Lines," *Performance Evaluation*, vol. 34, pp. 249–271, Dec. 1999.

[6] M. E. Crovella and A. Bestavros, "Self-similarity in World Wide Web Traffic: Evidence and Possible Cause," in *Proceedings of IEEE/ACM SIGMETRICS '96*, pp. 160–169, May 1996.

[7] M. Mathis, J. Semke, J. Mahdavi, and T. Ott, "The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm," *ACM SIGCOMM Computer Communication Review*, vol. 27, pp. 67–82, July 1997.

[8] D. D. Clark and W. Fang, "Explicit Allocation of Best Effort Packet Delivery Service," *IEEE/ACM Transaction on Networking*, vol. 6, pp. 362–373, Aug. 1998.

[9] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An Architecture for Differentiated Services," *Request for Comments (RFC) 2475*, Dec. 1998.

[10] S. Floyd and V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance," *IEEE/ACM Transactions on Networking*, vol. 1, pp. 397–413, Aug. 1993.

[11] The VINT Project, "UCB/LBNL/VINT Network Simulator - ns (Version 2)." available at `http://www.isi.edu/nsnam/ns/`.

[12] R. Jain, A. Durresi, and G. Babic, "Throughput Fairness Index: An Explanation," *ATM Forum Contribution '99*, Feb. 1999.

[13] P. Barford and M. E. Crovella, “Generating Representative Web Workloads for Network and Server Performance Evaluation,” in *Proceedings of Performance ’98/ACM SIGMETRICS*, pp. 151–160, June 1998.

List of Tables

Tables

Table 1: Transmitted data size distribution

Document size ($<$ 133 [KByte])	Log Normal	$Mean = 9.357$ [KByte]	$Std. = 1.318$
Document size ($\geq$ 133 [KByte])	Pareto	$Mean = 133$ [KByte]	$Shape = 1.5$
Think time	Pareto	$Mean = 15$ [sec]	$Shape = 1.5$

Table 2: Control parameters of RED

w_q	0.002
max_p	0.1
min_{th}	25 [packet]
max_{th}	75 [packet]

Table 3: Control parameters of proposed method

w_q	0.002
min_{th}	25 [packet]
max_{th}	75 [packet]
T_{update}	0.5 [sec]
N	100 [packet]
α	1.05
β	0.95

Table 4: Modified fairness index

	$tau = 1$ [ms]	$\tau = 10$ [ms]	$\tau = 100$ [ms]
Proposed Method	0.980	0.986	0.962
TailDrop	0.670	0.643	0.710
RED	0.815	0.795	0.878

Table 5: Link utilization between router A and router B

	$\tau = 1$ [ms]	$\tau = 10$ [ms]	$\tau = 100$ [ms]
Proposed Method	92.0%	92.8%	92.6%
TailDrop	92.8%	93.8%	93.2%
RED	92.5%	93.0%	92.5%

List of Figures

Figures

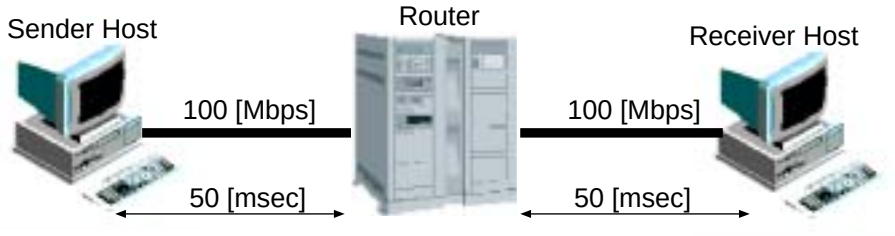


Figure 1: Simulation model in Section 2 and 3

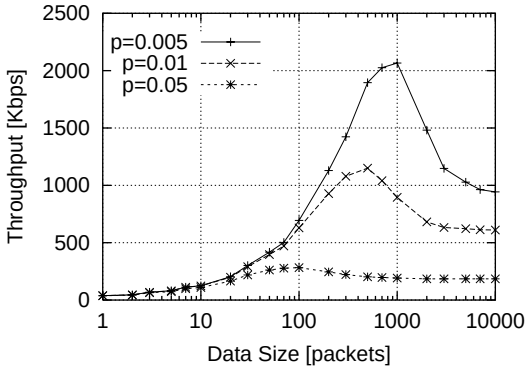


Figure 2: Unfairness caused by transmitted data size

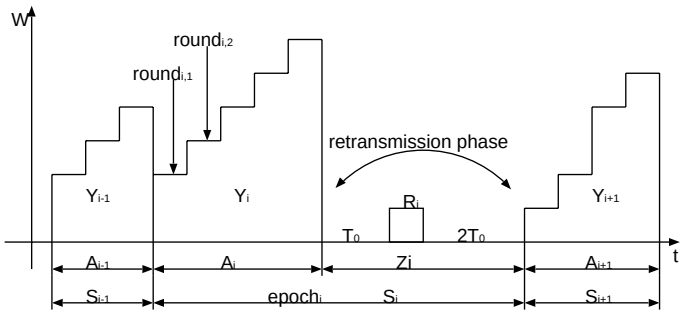


Figure 3: Window evolution of TCP in congestion avoidance phase

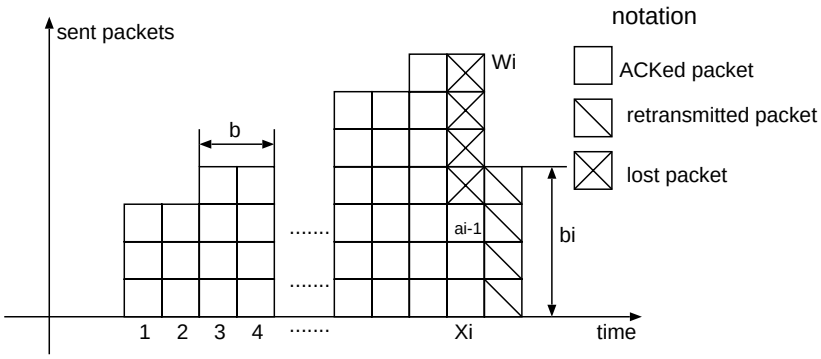
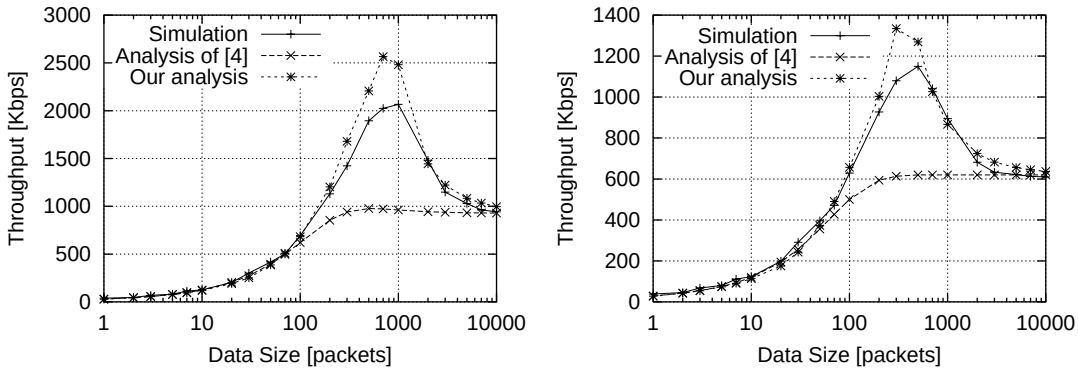
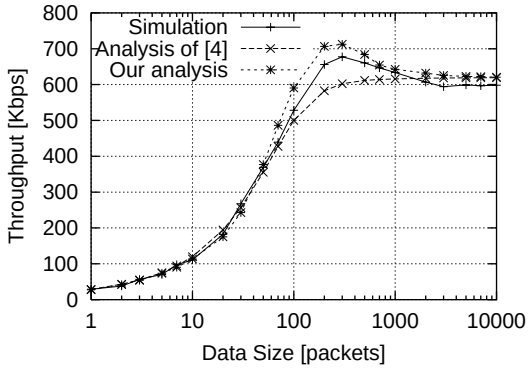


Figure 4: Detailed behavior of TCP in a epoch

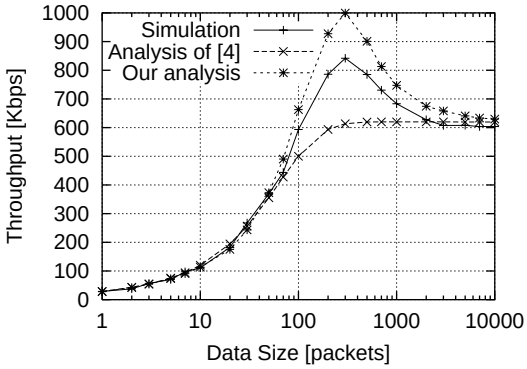


(a) p=0.005 (b) p=0.01

Figure 5: Validation of analysis: the case with constant packet loss probability



(a) The case with 32 packets of maximum window size



(b) The case with 64 packets of maximum window size

Figure 6: Effect of maximum window size

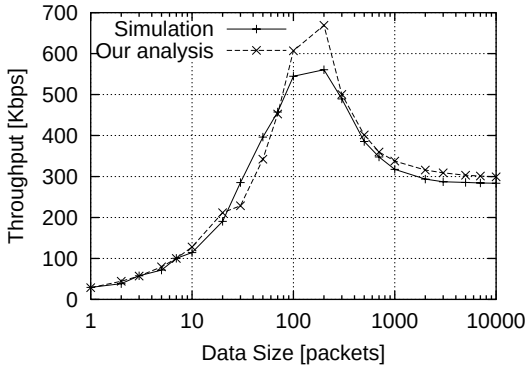


Figure 7: Validation of analysis:

the case with changing packet loss probability

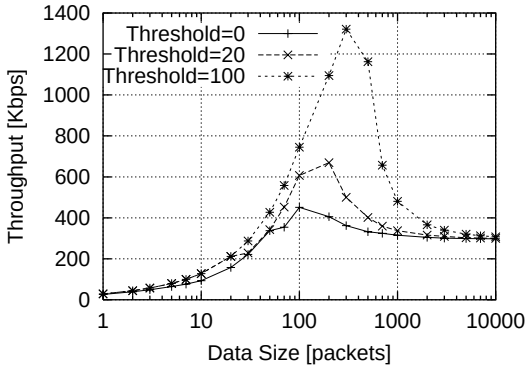
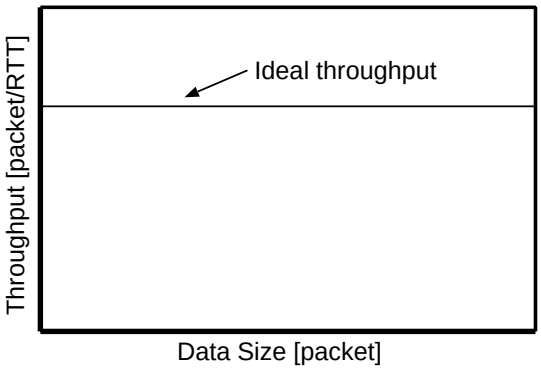
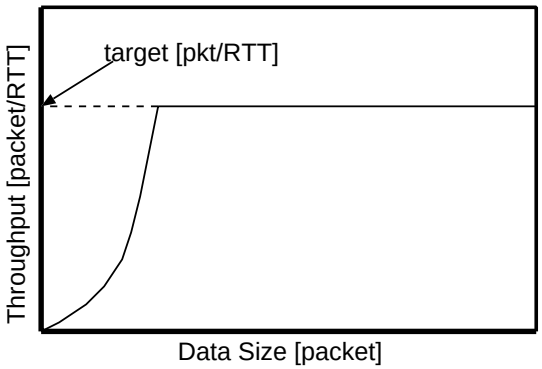


Figure 8: Effect of threshold value



(a) Complete Fairness



(b) Goal of proposed method

Figure 9: Modification of the fairness definition

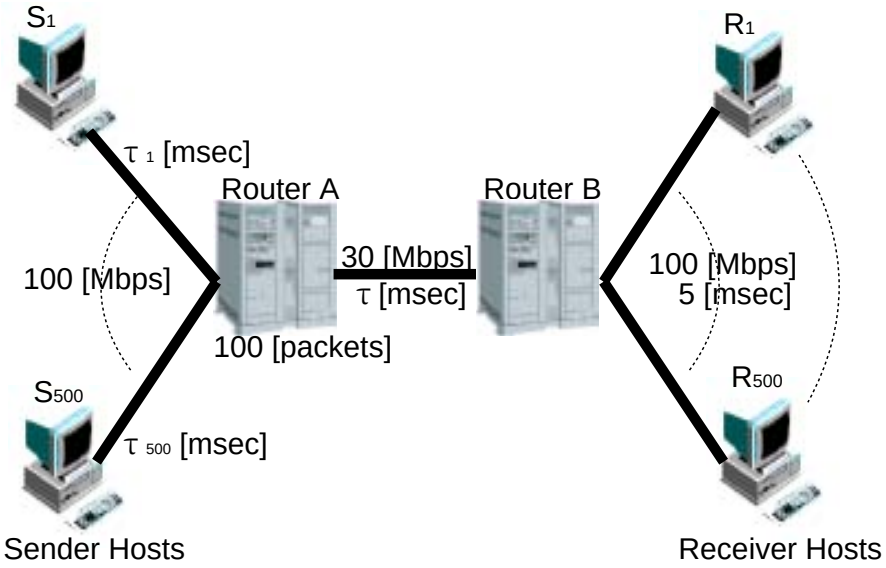
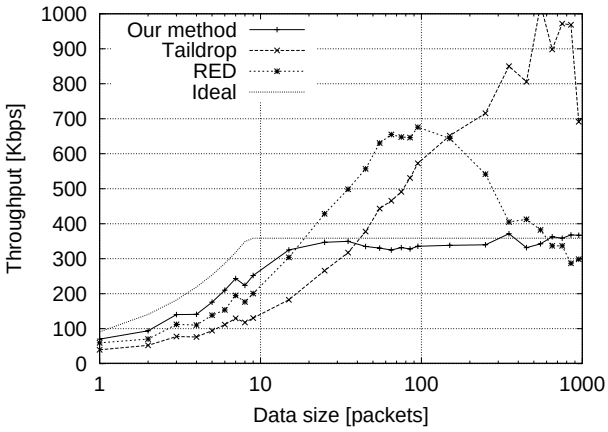
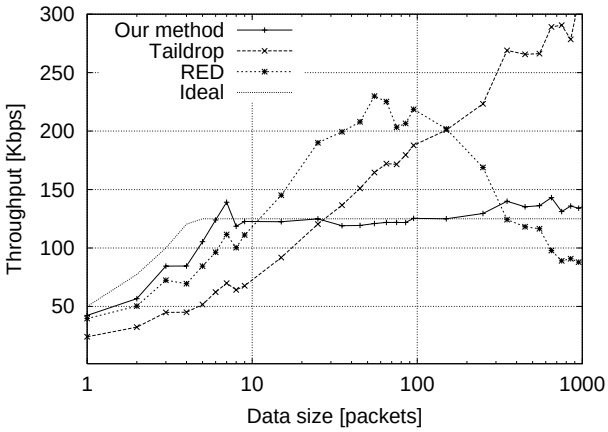


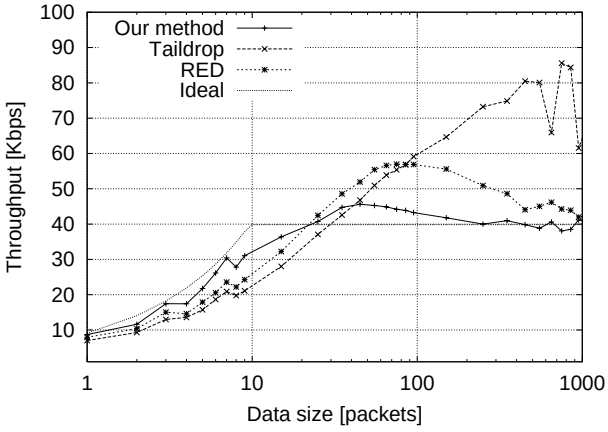
Figure 10: Network model



(a) $\tau = 1$ [ms]



(b) $\tau = 10$ [ms]



(c) $\tau = 100$ [ms]

Figure 11: Comparison of throughput between proposed method and other methods

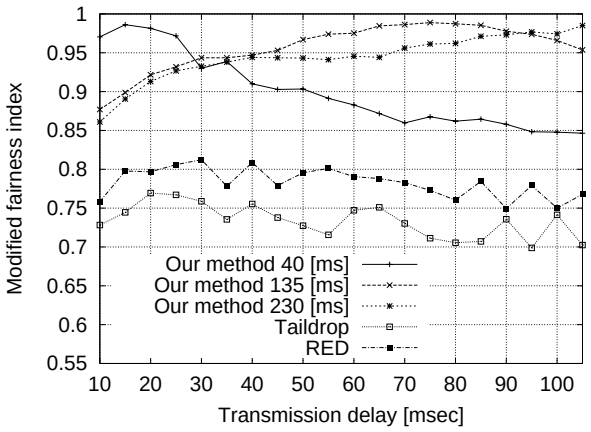
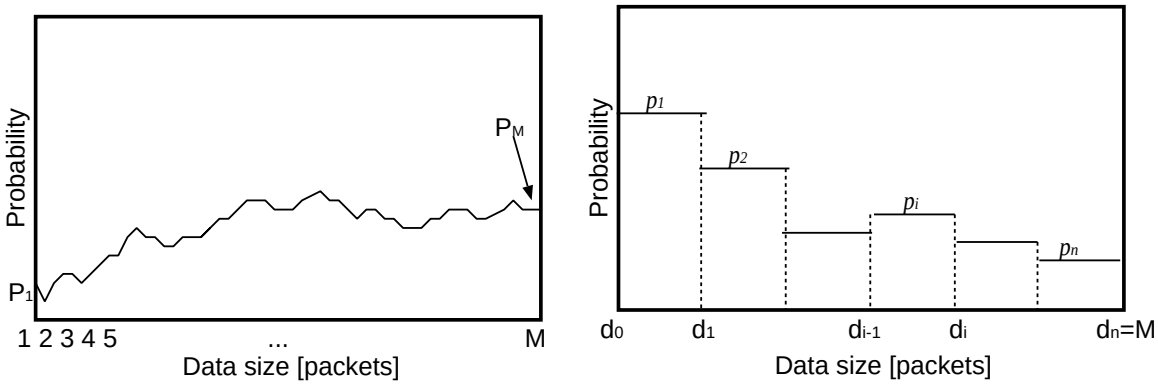


Figure 12: The case with various transmission delays



(a) The case with changing them at every packet

(b) The case with changing them at some thresholds

Figure 13: Packet discarding probabilities

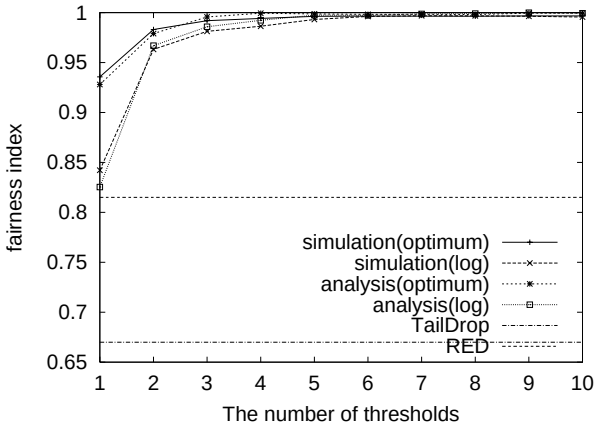


Figure 14: The relationship between the number of thresholds and modified fairness index