TCP Throughput Analysis with Variable Packet Loss Probability for Improving Fairness among Long/Short-lived TCP Connections

Koichi Tokuda† Go Hasegawa‡ Masayuki Murata‡

†Graduate School of Information Science and Technology, Osaka University
1-3 Machikaneyama, Toyonaka, Osaka 560-8531,
Phone: +81-6-6850-6616, Fax: +81-6-6850-6589
E-mail: kouichit@ics.es.osaka-u.ac.jp

‡Cybermedia Center, Osaka University
1-30 Machikaneyama, Toyonaka, Osaka 560-0043, Japan
Phone: +81-6-6850-6616, Fax: +81-6-6850-6589
E-mail: {hasegawa, murata}@cmc.osaka-u.ac.jp

***Abstract*-Short-lived TCP connections, which send small documents, suffer from significant low throughput compared with long-lived TCP connections, which transmit large documents. It is because of an inherent nature of the ACK-based window flow control of TCP. In this paper, we focus on the unfairness problem of TCP connections transmitting differently sized documents. To improve the fairness, it is necessary to treat packets from short-lived TCP connections preferentially over those from long-lived TCP connections. For realizing it, we first introduce a new analysis approach to estimate the throughput of TCP connections when the packet loss probability of each transmitted packet from the same connection is different. Our analysis can give a better estimation of the TCP throughput than the existing analysises. Using the analysis results, we propose a new algorithm to improve the fairness, and investigate its effectiveness through simulation experiments. We show that our proposed algorithm can improve the fairness without degradation of the network 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 efforts have been recently made 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, the fairness among users within a class is still important to be achieved.

For an inherent nature of the ACK-based window flow control of TCP, there exists the inevitable unfairness among TCP connections that transmit differently sized documents even when all other characteristics of the connections such as link bandwidths and propagation delays are identical [1]. This unfairness is brought by the difference of TCP window sizes. A TCP connection transmitting small documents (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 documents (*long-lived connection*) becomes large enough since it takes many RTTs to transmit the documents. As a result, when the short-lived connection and the long-lived connection share the bottleneck link, the long-lived connection utilizes larger amount of the network bandwidth.

Therefore, in this paper, we tackle the unfairness problem between long-lived and short-lived TCP connections. We first confirm the unfairness through some simple simulation experiments. One possible way to overcome the unfairness is to distinguish long-lived and short-lived connections and treat packets from short-lived TCP connections preferentially over those from long-lived TCP connections at the router buffer. For realizing that mechanism, We propose hash-RED method, which use hash table for differentiation of long-lived and short-lived connections, and RED (Random Early Detection) [2] for protecting short-lived connections.

To derive an appropriate parameter of hash-RED method, it is necessary to estimate throughput of TCP connections when we change the packet discarding probability during the connection. Accordingly, we develop a new analysis technique to estimate TCP throughput under such a situation. It can give higher accuracies than the previously proposed analysis methods described in [3, 4].

The rest of this paper is organized as follows. Section 2 shows some simulation results to confirm the unfairness property between long-lived and short-lived TCP connections. Section 3 proposes a new analysis approach for estimation of the throughput of TCP connections when the packet loss probability of each transmitted packet is different. The hash-RED method to improve the fairness and simulation results is shown in Section 4. Finally, we present conclusions and future work in Section 5.

2 Unfairness between Long-lived and Short-lived TCP Connections

In this section, we confirm the unfairness between long-lived and short-lived TCP connections 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 bandwidth of both links are 500 [Mbps], which are enough large so as not to limit the throughput of TCP connections. We set the packet length fixedly to 1460 [Bytes]. The receive buffer at the receiver host and the maximum window size of TCP are set enough large not to limit the TCP throughput. Although no packet loss occurs 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 various sizes of documents by TCP, and the throughput value of each document transfer is observed.

Figure 2 shows the relationship between the transmitted document size in packet and the average throughput of 1000 times transmissions of each size of the document. We can observe from this figure that if the transmitted document size is small (a short-lived TCP connection), the TCP connection suffers from very low throughput. It is because the short-lived TCP connection finishes its document transmission before its

congestion window becomes large, which is inevitable due to an inherent nature of the ACK-based window flow control of TCP. Furthermore, when packet losses occur, the short-lived TCP connection cannot detect the loss by duplicate ACK packets because of its too small window size. It brings TCP timeout, which results in serious performance degradation.

If the transmitted document size is large (a long-lived TCP connection), on the other hand, the TCP connection can obtain large throughput as shown in Figure 2. It is because the congestion window of the connection becomes large enough during its document transfer, which results in that it can utilize the link bandwidth effectively. Especially when the number of transmitted packets is from 100 to 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 loss occurs in the network. In addition, a large size of the 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-sized documents exist with large probabilities. That is, the unfairness between the long-lived and short-lived TCP connections is serious in 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 mechanism.

3 TCP Throughput Analysis

In this section, we develop a new analysis modeling of TCP to derive a TCP throughput when packet discarding probability of each transmitted packet is different. We then verify the accuracy of our analysis by comparing the analysis results with simulation results.

3.1 Assumptions

In our analysis, we make the following assumptions. The sender host uses the congestion control mechanisms of TCP Reno [7]. ACK packets are never lost in the network. The time needed to transmit all packets within a congestion window is smaller 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 probability is independent of the window size. For treating the model in which packet discarding probabilities of transmitted packets are different, we denote the packet discarding probability of the i-th packet by p_i.

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

3.2 Slow Start Phase

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], especially the difference of the packet discarding probabilities.

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

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

By using $E[d_{ss}]$, we can obtain the following equations where $E[T_{ss}]$ is the time length of the initial slow start phase, w_1 is the initial window size, W_{max} is the maximum window size, and $\gamma = (1 + \frac{\delta}{b})$. For more detail, refer to [3].

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

3.3 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, TCP sender host transmits its packets allowed by its congestion window. By receiving ACK packets from the receiver host, the TCP sender host increases its congestion window and transmits new packets to the receiver. When packet loss occurs, 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 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, j-th round in 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 when the TCP sender host is sending packets is denoted by A_i, and the time duration that the sender waits for a retransmission timeout (called as a *retransmission phase*) is Z_i. Therefore, $S_i = A_i + Z_i$. Further, M_i is defined as the number of packets transmitted in epoch$_i$. We further denote the number of packets transmitted in A_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:

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

Following [3], we can obtain Q_i. We then have n_{ca}, the number of epochs in the congestion avoidance phase necessary to finish the document transmission, which can be obtained by solving the following equation:

$$d = E[d_{ss}] + \sum_{i=1}^{n} E[M_i]$$

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

$$B = \frac{\sum_{i=1}^{n}(E[Y_i] + Q_i \cdot E[R_i])}{\sum_{i=1}^{n}(E[A_i] + Q_i \cdot E[Z_i])} \quad (2)$$

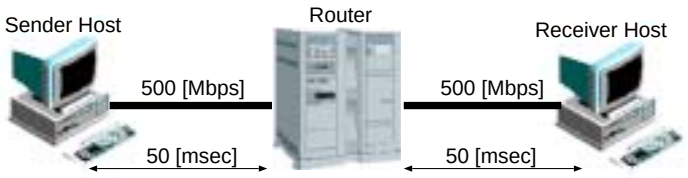


Figure 1: Network model (1)

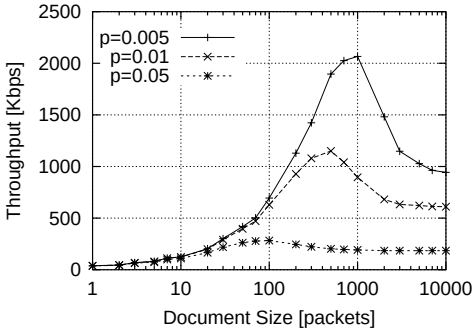


Figure 2: Unfairness between long-lived and short-lived connections

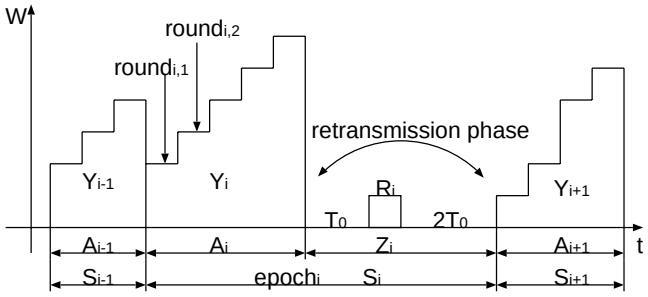


Figure 3: Evolution of TCP window sizein congestion avoidance phase

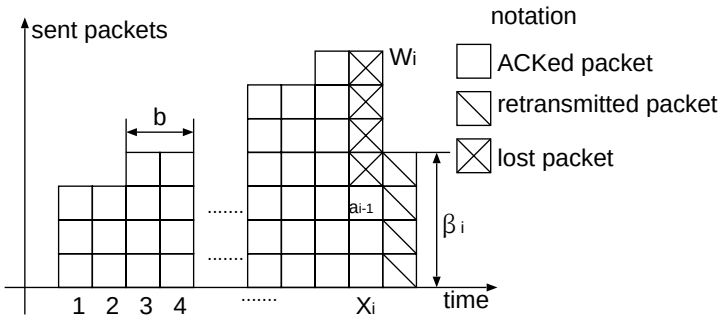


Figure 4: Detailed behavior of TCP in a epoch

In what follows, we calculate $E[Y_i]$, $E[R_i]$, $E[A_i]$, 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 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 [7]:

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

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

Here, we denote the number of packets that the TCP can transmit from the beginning of the connection to epoch$_i$ by AL_i, and ALY_i is defined as $AL_{i-1} + Y_i$. We then obtain the following equations for $E[Z_i]$ and $E[R_i]$.

$$E[Z_i] = \sum_{k=1}^{d-ALY_i} P[R_i = k]L_k + \prod_{k=1}^{d-ALY_i} p_{ALY_i+k} L_{d-ALY_i} \quad (3)$$

$$E[R_i] = \sum_{k=1}^{d-ALY_i} P[R_i = k]k + \prod_{k=1}^{d-ALY_i} p_{ALY_i+k}(d - ALY_i) \quad (4)$$

Figure 4 shows the 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,

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

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

$$E[\alpha_i] = \sum_{k=1}^{d-AL_{i-1}} P[\alpha_i = k]k + \prod_{k=1}^{d-AL_{i-1}} (1 - p_{AL_{i-1}+k})d \quad (5)$$

Observing Figure 4 leads to the following equation.

$$E[Y_i] = E[\alpha_i] + E[W_i] - 1 \quad (6)$$

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

$$E[W_i] = \frac{E[W_{i-1}]}{2} + \frac{E[X_i]}{b} \quad (7)$$

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

$$E[Y_i] = \sum_{k=0}^{\frac{E[X_i]}{b}-1} (\frac{[W_{i-1}]}{2} + k)b + E[\beta_i]$$
$$= \frac{E[X_i]}{2}(\frac{E[W_{i-1}]}{2} + E[W_i] - 1) + \frac{E[W_i]}{2} \quad (8)$$

By solving Equations (5), (7) and (8) about W_i, we obtain

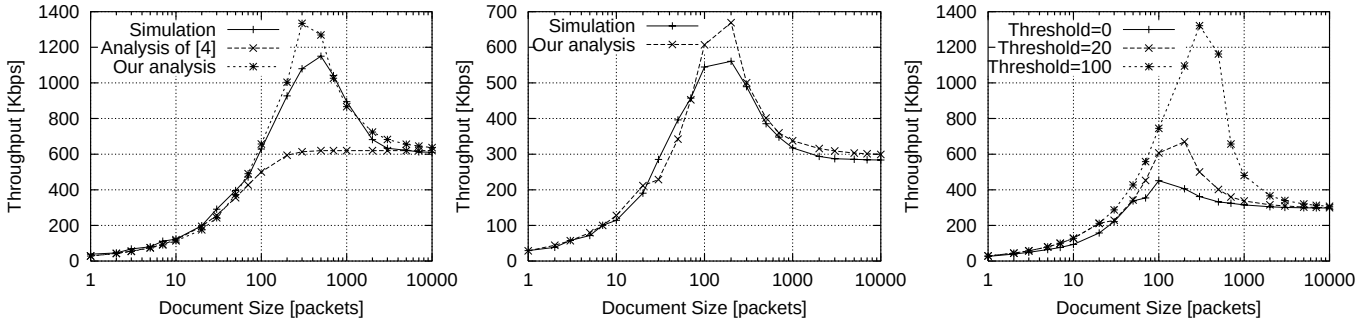
$$E[W_i] = \frac{b+1}{2b} + \frac{\sqrt{(b+1)^2 - 2b(-\frac{2}{b}E[W_{i-1}]^2 + bE[W_{i-1}] - 4E[\alpha_i] + 4)}}{2b} \quad (9)$$

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

$$E[A_i] = (E[X_i] + 1) \cdot E[r] \quad (10)$$

$$E[X_i] = b(E[W_i] - \frac{E[W_{i-1}]}{2}) \quad (11)$$

Finally, by using Equations (3), (4), (6) and (10), we can calculate B from Equation (2).



(a) Case of constant packet loss probability (b) Case of changing packet loss probability (c) Case of changing threshold K

Figure 5: Numerical results of the analysis

3.4 Validation of Analysis and Discussions

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 purpose. Figure 5 shows the relationship between the transmitted document size (in packets) and the throughput in the document transfer where we set the packet discarding probability of the link between the sender host and the router to 0.01 (Figure 5(a)). This figure includes the results of simulation experiments, our analysis, and the analysis presented by [4]. From this figure, we can see that the analysis of [4] cannot estimate the throughput well, especially when the number of transmitted packets is around 100 through 1000. It is because the analysis of [4] assumes that the TCP host shifts its state to be steady 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 document transfer with higher accuracy because we model the detailed behavior of TCP after the initial slow start phase with large window size.

We next evaluate the accuracy of our analysis when packet discarding probabilities differ in each transmitted packets of a document, which is the main objective of our analysis. Figure 5(b) shows the relationship between the document size and the throughput in TCP document transfer where we set the packet discarding probability to 0 for 1st through 20th transmitted packets, and 0.03 for 21st and later transmitted packets. From this figure, we can observe that our analysis again estimates the throughput of the TCP exactly even when the packet discarding probabilities are dynamically changed.

We present the analysis results in Figure 5(c) where we change the threshold value (denoted by K). At the given threshold, the packet loss probability is increased from 0 to 0.03. Note that the results labelled by 'Threshold=0' in Figure 5(c) means that all packets are discarded with probability of 0.03. It is shown in Figure 5(c) that by setting the threshold value larger, the throughput of the small document transfer can be improved. That is, we can improve the fairness between long-lived and short-lived TCP connections by setting the packet discarding probability of short-lived connections to 0, and that of long-lived connections to a higher value. In the next section, we will propose the algorithm to set the threshold value and packet discarding probability for long-lived connections appropriately by using the analysis results.

4 hash-RED method

To overcome the unfairness problem, we treat packets from short-lived TCP connections preferentially over those from long-lived TCP connections at the router buffer. For realizing that mechanism, we propose hash-RED method, which uses hash table for differentiation of long-lived/short-lived connections, and RED-based algorithm for protecting short-lived connections.

For differentiation of long-lived/short-lived connections we use hash table, and the hash key is a combination of source/destination IP addresses, port numbers. Each entry of the hash table includes a counter which records the number of transmitted packets by the connection, and a timestamp to record the time at which the last packet of the connection arrived at the router. When a new packet arrives at the router, the router seeks the corresponding entry in the hash table using the hashing function, increments the counter, and updates the timestamp value by the current time. If the difference of the timestamp value and the current time is larger than T, the router determines that the connection begins another document transmission and resets the counter value. Detection of long-lived TCP connection can be made by checking whether the counter value is larger than K or not.

Next, we propose the algorithm which is based on RED for protecting short-lived TCP connections. RED discards the incoming packets with a certain probability, which is calculated by the average queue length at the router buffer and some control parameters. In the RED-based algorithm proposed here, we use the packet discarding probability calculated by the original RED algorithm as a *target probability* p_t. In the hash-RED algorithm, we set the packet discarding probability for packets from short-lived TCP connections to 0, and set higher value than p_t for those from long-lived TCP connections, so that the total packet discarding probability becomes p_t.

As shown in Figure 5(c), to determine the threshold value K is important. In the hash-RED, We use the analysis result in the previous section in order to choose K appropriately. From the analysis results, we can obtain the relation between the transmitted document size, denoted by f, and throughput, ρ, as shown in Figure 5(c). By denoting the relation by the function $\rho(f)$, we define the fairness index value $F(K)$ as follows;

$$F(K) = \sum_{i=1}^{L} P(i) \cdot (\rho(i) - \rho(L))^2$$

where $P(i)$ is the probability that the document contains i packets. It can be obtained from the WWW document size distribution in [8]. By setting L to large value, $\rho(L)$ would represent the throughput of long-lived TCP connections. Figure 7 shows the sample results of $F(K)$ values, where we set L = 10000 [packets]. We can see from this figure that once p_t is given, we can find the optimal value of K such that $F(K)$ becomes the smallest value. In the RED-based algorithm, we use this fairness index $F(K)$ to determine the threshold value K.

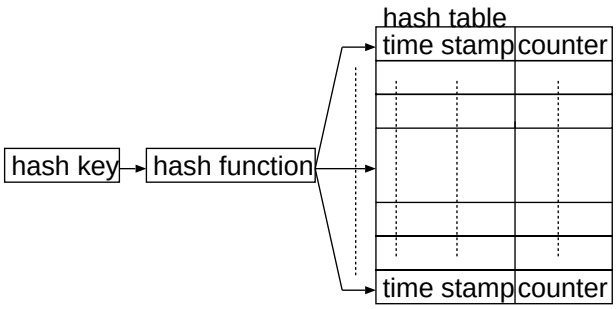


Figure 6: Differentiation of long-lived/short-lived connections

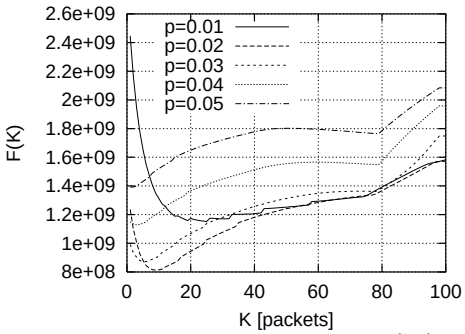


Figure 7: Sample of change of $F(K)$

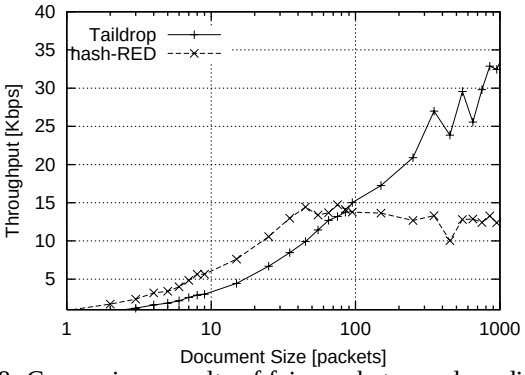


Figure 8: Comparison results of fairness between long-lived and short-lived connections

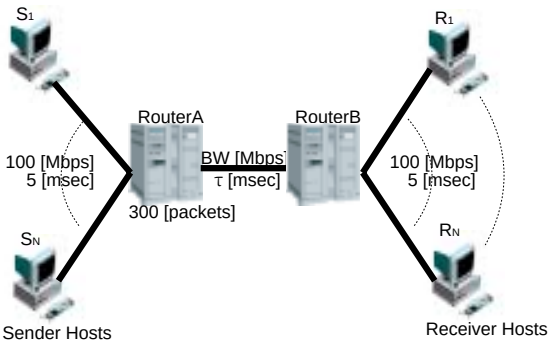


Figure 9: Network model (2)

Table 1: Link utilization of the bottleneck link

	utilization
Tail-Drop	0.9536
hash-RED	0.9521

Figure 8 shows the relationship between the transmitted document size (in packets) and the throughput where we use tail-drop and hash-RED as the buffer discipline. All sender hosts behave as WWW sender hosts where the size of the transmitted documents follows the distribution presented in [8]. Table 1 shows the link utilization of the bottleneck link. From the simulation results, We can conclude that hash-RED can improve the fairness without degradation of the network utilization.

5 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 connection that transmits small documents suffer from lower throughput than the long-lived connection. We have also pointed out, through the new analysis for TCP throughput, that it is necessary to introduce a mechanism to distinguish the long-lived and short-lived TCP connections at the router, in order to give preferential service to the short-lived TCP connections for fairness improvement. Our proposed algorithms have realized such service. Our evaluation results through simulation experiments have revealed that our proposed algorithms can detect the long-lived TCP connections at the router and improve the fairness, without any interactions between routers.

Acknowledgement

This work was partly supported by support system for R&D activities in info-communications area by Telecommunication Advancement Organization of Japan, a Grant-in-Aid for Scientific Research (B) 13450158 from the Japanese Ministry of Education, Culture, Sports and Science and Technology of Japan, and a Grant-in-Aid for Encouragement of Young Scientists (A) 13750349 from The Ministry of Education, Culture, Sports and Science and Technology of Japan.

References

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

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

[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*, Sept. 1998.

[4] N. Cardwell, S. Savage, and T. Anderson, "Modeling TCP Latency," in *Proceedings of IEEE INFOCOM 2000*, 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*, pp. vol. 34, no. 4, 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*, May 1996.

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

[8] P. Barford and M. E. Crovella, "Generating Representative Web Workloads for Network and Server Performance Evaluation," in *Proceedings of the 1998 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, June 1998.