Analysis and Improvement of the Fairness between Long-lived and Short-lived TCP Connections

Koichi Tokuda, Go Hasegawa, and Masayuki Murata

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

Abstract. Short-lived TCP connections, which transmit small documents, suffer from significant low throughput compared with long-lived TCP connections, which transmit large documents, 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. We first confirm the unfairness property through simple simulation results, and show the necessity of fairness improvement algorithms at the router buffer. To improve the fairness, we propose new algorithms, and investigate their effectiveness through simulation results. We show that our proposed algorithms can provide better 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.

The authors in [1] have pointed out the unfairness among TCP (Transmission Control Protocol) connections that transmit differently sized documents, even when all other characteristics of the connections such as link bandwidths and propagation delays are identical. 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 congestion 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 (Round Trip Times) to transmit the documents. As a result, when the long-lived connection and short-lived connection share the bottleneck link, the long-lived connection utilizes larger amount of the network bandwidth.

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 change the packet discarding probability of packets from long-lived and short-lived TCP connections at the router buffer based on the RED algorithm [2]. However, differently from [1], we do not expect cooperation between edge and core routers. Instead, each router solely tries to improve the fairness among connections in our proposed method. For this purpose, we divide the problem into the following two sub-problems; (1) how to distinguish long-lived and short-lived TCP connections and detect long-lived TCP connections at the router, and (2) how to protect short-lived connections from long-lived connections. We propose some algorithms against each sub-problem. We then investigate the fairness property of the proposed algorithms and those proposed in [1] through several simulation experiments.

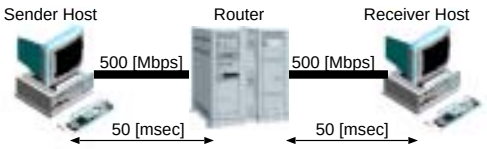


Fig. 1. Simulation model in Section 2

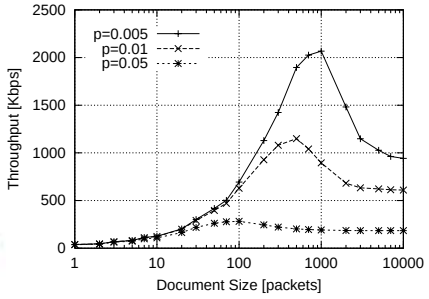


Fig. 2. Unfairness between long-lived and short-lived connections

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

In this section, we confirm the unfairness 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 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, 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, 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 enough large 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. In the next section, we propose some algorithms to improve the fairness.

3 Proposed Algorithms

We have confirmed the unfairness problem in terms of throughput in document transfer in Section 2. To improve the fairness, it is necessary to distinguish the long-lived and short-lived TCP connections, and protect the packets from short-lived connections. In this section, we propose some differentiation algorithms and preferential algorithms.

3.1 How to find long-lived TCP connection?

To distinguish long-lived and short-lived TCP connections at the router, we consider the following three algorithms; packet marking at edge routers, packet sampling, and hashing. The packet marking algorithm has been proposed in [1]. Hereafter, we define a long-lived connections as a TCP connection transmitting more than K packets, and a short-lived connections as a TCP connection whose transmitting document size is equal to or less than K packets.

Packet marking We first review the packet marking algorithm which has proposed in [1]. This algorithm works under the cooperation between edge routers at the entrance of the network and core routers within the network. The edge router has a table for maintaining the number of packets from each connection, and monitors all packets being injected into the network. Then it identifies the connection of the packets by source/destination IP addresses and port numbers, and increments the counter value of the corresponding entry of the table. The edge router marks packets from the connection whose counter value is larger than K. Then the core router can detect the long-lived TCP connections by checking the mark of incoming packets.

Packet sampling We next consider the packet sampling algorithm. It needs two tables at the router; a sampling table and classification table. The router samples incoming packets at a certain rate, and records the information of the packet to the sampling table. At regular intervals, the router checks the sampling table and classification table, and updates the classification table as follows; when a connection in the sampling table does not appear in the classification table, the router determines that the connection starts packet transmission and adds it to the classification table. When a connection in the sampling table is also found in the classification table, the router increments the counter value of the connection of the classification table. When a connection in the classification table cannot be found in the sampling table, the router decides that the connection finishes packet transmission and remove the connection from the classification table.

The detection of long-lived TCP connections can be done as follows. After updating the classification table, the router checks all counter values in the classification table. Assume the sampling rate is I [1/packets], the counter value of connection i is C_i, and the updating interval of the classification table is T [sec]. When the inequality $C_i \geq \frac{K}{I}$ is satisfied, the connection i is detected as a long-lived TCP connection.

Hashing The third one is based on the hashing algorithm. In this algorithm we use a hash table. 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 each 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 by 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.

3.2 How to differentiate long-lived and short-lived connections?

We next explain how to improve fairness after detecting long-lived TCP connections at the router. We first introduce the RIO-based algorithm which has been proposed in [1], and propose two new algorithms, which are RED-based and PRQ (priority queue)-based algorithms.

RIO-based algorithm RIO (RED In and Out) [3] is proposed for realizing the one service in the DiffServ architecture. Packets injected into the network are marked IN or OUT at the edge router according to the contracted profiles. The RIO-enabled core routers within the network have two sets of RED parameters for IN packets and OUT packets, and switch them according to the mark of the packets. Generally, the packet discarding probability for OUT packets is set higher than that for IN packets, by setting max_{th} and min_{th} smaller, and max_p larger. Therefore, this mechanism can be used to improve the fairness, by considering that packets from short-lived TCP connections are IN packets, and those from long-lived TCP connections are OUT packets. In [1], the authors used the RIO-based algorithm and packet marking as explained in Subsection 3.1

RED-based algorithm Next, we propose the algorithm which is based on RED. 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, we use the packet discarding probability calculated by the original RED algorithm as a *target probability* p_t.

In RED-based algorithm, we simply set the packet discarding probability for packets from short-lived TCP connections to 0, and set higher value for those from long-lived TCP connections, which is calculated by using the analysis results, so that the total packet discarding probability becomes p_t.

We assume the packet arriving ratios of long-lived and short-lived TCP connections are a_s and a_l, respectively, where $a_s + a_l = 1$. Then, we set the packet discarding probability of packets from long-lived and short-lived TCP connections, which are denoted by p_s and p_l, respectively, such that the equation, $p_t = p_l \cdot a_l + p_s \cdot a_s$, is satisfied. If we simply set $p_s = 0$, p_l is given by $p_l = \frac{p_t}{a_l}$.

To determine p_l, we need to know a_l. For simplifying the algorithm, we assume that the distribution of the transmitted document size follows the result in [4]. The authors in [4] have reported that the WWW document size follows log-normal distribution for small documents ($\leq$ 136 [KByte]), and pareto distribution for large documents ($>$ 136 [KByte]). Therefore, when we set the threshold value K to a certain value, we can calculate the packet arriving ratio of long-lived and short-lived TCP connections from the distributions. Then, we can determine p_l.

The remaining problem is how to set the threshold value K. We can use our analysis result to choose K appropriately. In [5] , we have developed the analysis for throughput estimation of a TCP connection when the packet loss probability for each transmitted packet is different. By using the analysis results, we can derive the TCP throughput when K-th or later packets are discarded with higher probability. By denoting the analysis results by the function $\rho(f)$, where f is the transmitted file size and ρ is the throughput, 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 [4]. By setting L to large value, $\rho(L)$ would represent the throughput of long-lived TCP connections. We determine the threshold K from the fact that the smaller $F(K)$ is, the better the fairness is.

Figure 3 shows the sample analytical results of F 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.

PRQ-based algorithm The second proposed algorithm uses a priority queue. The router prepares two queues; the one for packets from long-lived TCP connections (called *long*

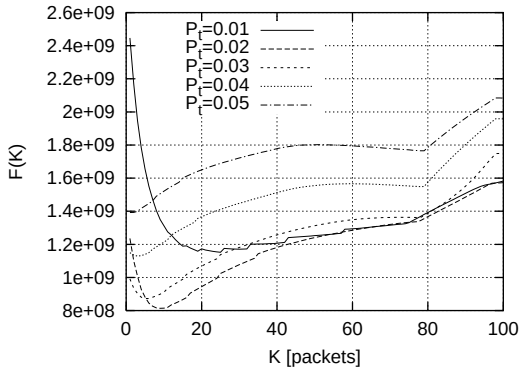


Fig. 3. Sample of change of $F(K)$

queue), and the other for short-lived TCP connections (*short queue*). An arriving packet is queued one of the two queues according to whether the packet is from long-lived connection or short-lived connection. The short queue has priority over the long queue to be served. That is, packets in the long queue are served only when the short queue is empty.

4 Evaluation Results

In this section, we investigate the performance of the algorithms explained in Section 3 through simulation results using ns-2 [6]. First, we compare the three classification algorithms (marking, sampling, and hash algorithms). Next we investigate the degree of fairness improvement where we compare the performance of nine methods, meaning that we combine the three differentiation algorithms (RIO, RED, and PRQ-based algorithms) with each of the three classification algorithms.

Figure 4 depicts the network model used in the simulation. The model consists of sender hosts (S_1,$\cdots$,S_N), receiver hosts (R_1,$\cdots$,R_N), routers (router A and B) and links interconnecting the hosts and routers. The sender host S_i transmits documents to the corresponding receiver host R_i. The bandwidths and the propagation delays of all links between the sender hosts and router A, and between the receiver hosts and router B are all set to 100 [Mbps] and 5 [msec], respectively. The bottleneck link between the two routers has BW [Mbps] of the bandwidth and τ [sec] of the propagation delay. We set the packet length fixedly to 1460 [Bytes], and the router buffer size to 300 [packets]. We run 1500 [sec] simulation in each experiment. We consider the following two scenarios defining the behavior of the sender hosts;

Scenario 1: We divide the sender hosts into two types; the hosts which send small documents (called *short hosts* in the below) and the hosts which send large documents (*long hosts*). The size of transmitted documents and thinking times between document transmissions used by each type of sender hosts follow pareto distributions whose parameters are shown in Table 1. We set the parameters in consideration of the average size of WWW document transfer for short-lived TCP connections, and FTP file transfer for long-lived TCP connections.

Scenario 2: All sender hosts behave as WWW sender hosts where the size of the transmitted documents follows the distribution presented in [4]. As shown in Table 2, the transmitted documents follow a log-normal distribution when the document size is smaller than 133 [KBytes], and a pareto distribution when the document size is equal to or larger than 133 [KBytes]. Thinking times follow the pareto distribution.

Table 1. Distribution parameters in Scenario 1

	short-lived connections	long-lived connections
Document size	Pareto avg=10 [KB] shape=1.13	Pareto avg=1000 [KB] shape=1.13
Thinking time	Pareto avg=2.0 [sec] shape=1.85	Pareto avg=20 [sec] shape=1.5

Table 2. Distribution parameters in Scenario 2

Document size ($<$ 133 [KB])	LogNormal avg=9.357 [packets]		std=1.318
Document size ($\geq$ 133 [KB])	Pareto	avg=133 [KB]	$shape$=1.5
Thinking time	Pareto	$avg = 15$ [sec]	$shape = 1.5$

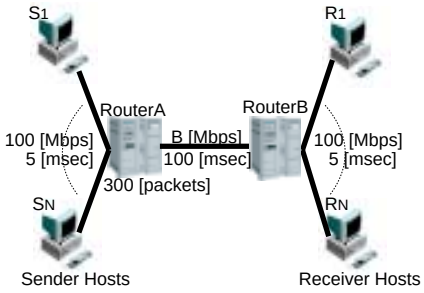


Fig. 4. Network model for fairness evaluation

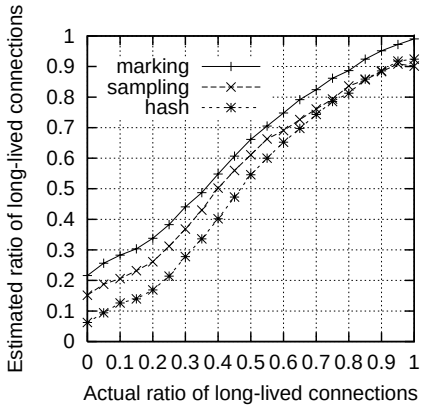


Fig. 5. Accuracy of classification algorithms (τ=30 [msec])

4.1 Accuracy of Classification Algorithms

We first investigate the performance of the three classification algorithms (marking, sampling, and hash) in terms of accuracy of detecting long-lived TCP connections at the router, where we use scenario 1 for the behavior of the sender hosts. We set the total number of sender hosts to 100, $B = 10$ [Mbps], $K = 7$ [packets]. For other parameters, we set $T = 0.5$ [sec] for sampling and hash algorithms, and $I = 3$ [1/packets] for the sampling algorithm. Figure 5 shows the average ratio of the sender hosts estimated as long-lived connections, as a function of the actual ratio of long-lived TCP connections when the propagation delay, tau, is $\tau = 30$ [msec].

We can observe from the figure that the estimated ratio is slightly higher than the actual ratio, especially in sampling and marking algorithms. This is because the short hosts sometimes send large documents and are classified as long-lived connections, due to the pareto distribution for the short hosts of scenario 1. However, all of three algorithms estimate the ratio of long-lived TCP connections with good precision. Especially, marking algorithm shows quite high accuracy because the marking algorithm can exactly detect the initiation and termination of TCP connections by observing SYN flag of incoming packets. On the other hand, the other two algorithms use timer mechanisms, and their performance is slightly lower than marking algorithm. However, the marking algorithm has the implementation problems. Therefore, we conclude that the sampling algorithm or hash algorithm should be used for the classification algorithm in consideration of the implementation difficulty and the effectiveness of the algorithms.

4.2 Overall Performance in Fairness Improvement

We next evaluate the performance of the algorithms shown in Section 3, in terms of the degree of fairness improvement. In this subsection, we consider nine methods by combining

Table 3. Control parameters for RED and RIO

Queue Parameters	min_{th}	max_{th}	P_{max}	w_q
RED	45	150	0.1	0.002
RIO for short	45	105	0.05	0.002
RIO for long	45	150	0.1	0.02

Table 4. Link utilization

	Marking	Sampling	Hash
RIO	0.9528	0.9528	0.9468
RED	0.9521	0.9512	0.9521
PRQ	0.9528	0.9531	0.9535
Tail-Drop	0.9536		

three classification algorithms (marking, sampling, and hash algorithms) and the three differentiation algorithms (RIO-, RED-, and PRQ-based algorithms). Note that in what follows, we represent nine methods as marking-RIO, marking-RED, marking-PRQ, sampling-RIO, sampling-RED, sampling-PRQ, hash-RIO, hash-RED and hash-PRQ.

We set the total number of sender hosts to 100, $BW = 10$ [Mbps], and $\tau = 30$ [msec]. The threshold value for detecting long-lived TCP connections, K, is set as follows; in RIO-based and PRQ-based algorithms, K is fixedly set to 7 [packets] considering the average size of WWW documents, since RIO-based and PRQ-based algorithms do not have any algorithms to determine appropriate value of K. In the RED-based algorithm, on the other hand, we dynamically change K according to the algorithm explained in Subsection 3.2, which uses our analysis. Other parameters for RIO-based and RED-based algorithms are shown in Table 3. We use the scenario 2 for evaluation.

Figure 6 shows the results of the throughput in document transfer as a function of the document size. The link utilization value of each algorithm is also shown in Table 4. We can see from these figures that the fairness can be actually improved, while keeping the total throughput as high as the case of the tail-drop algorithm.

If we use sampling-PRQ, the throughput becomes quite high for both of long-lived and short-lived connections as compared with other algorithms. This is because of the classification instability of the sampling algorithm and characteristics of the PRQ-based algorithm. The sampling algorithm sometimes fails to distinguish long-lived TCP connections, as shown in Figure 5. Then packets from long-lived TCP connections often enter the short queue. Those packets experience smaller queuing delays than those passing through the long queue, the long-lived connections sometimes are given high throughput as shown in Figure 6.

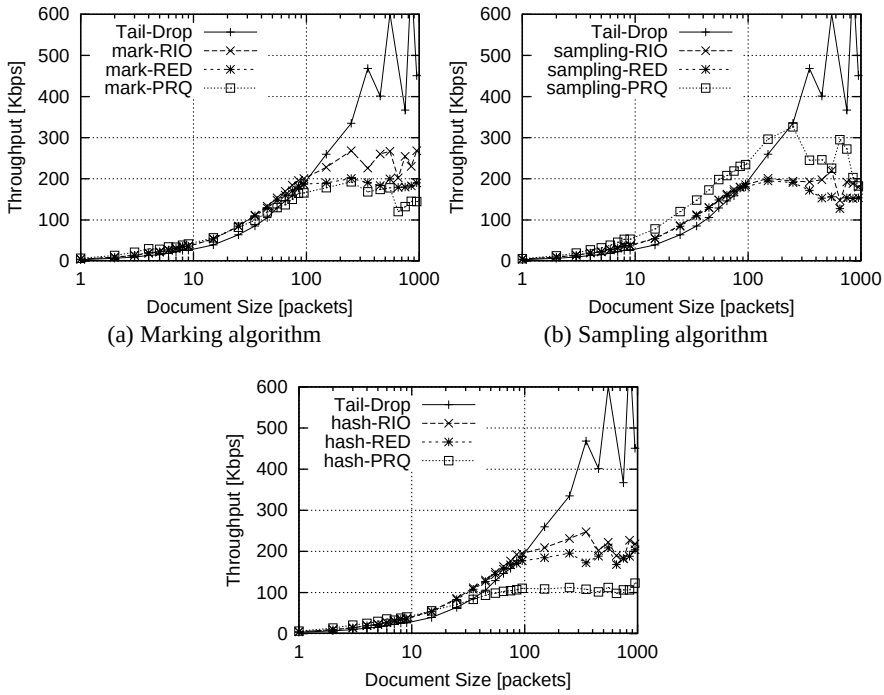
We can conclude from all of the above results that all of nine methods can improve the fairness between long-lived and short-lived TCP connections, without degradation of the utilization of the bottleneck link. With consideration of the implementation difficulty and simulation results, the combination of hash algorithm for classification and RED-based algorithms for differentiation are superior to the other algorithms.

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 connections suffer from lower throughput than the long-lived connections. It is necessary to introduce a mechanism to distinguish the long-lived and short-lived TCP connections at the router, 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. In consideration of performance and processing overhead, we conclude that hash-RED is superior to other methods.

Acknowledgements

This work was partly supported by the Research for the Future Program of the Japan Society for the Promotion of Science under the Project “Integrated Network Architecture for Advanced Multimedia Application Systems,” a Grant-in-Aid for Scientific Research (B)



(a) Marking algorithm (b) Sampling algorithm

(c) Hash algorithm

Fig. 6. Comparison results of fairness between long-lived and short-lived connections in scenario 2

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. 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.
4. 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.
5. K. Tokuda, G. Hasegawa, and M. Murata, "TCP Throughput Analysis with Variable Packet Loss Probability for Improving Fairness among Long/Short-lived TCP Connections," in *Proceedings of IEEE CQR 2002*, Mar. 2002.
6. Network Simulator - `ns` (version 2). available from *`http://www.isi.edu/nsnam/ns/`*.