Congestion Control with Two Fair Allocation Modes to Achieve RTT-Fairness

K. Ogura*, Z. Su* and J. Katto*

*Dept. of Computer Science, Waseda University
3-4-1, Okubo, Shinjuku-ku, Tokyo, 189-8555, Japan.
{ogura, katto}@katto.comm.waseda.ac.jp

ABSTRACT

This paper focuses on RTT-fairness of multiple TCP flows over the Internet, and proposes a new congestion control supporting RTT-fairness. Today, it is a serious problem that the flows having smaller RTT achieve more bandwidth than others when the flows having different RTT values compete. This means that a user with longer RTT may not be able to obtain sufficient bandwidth by the current methods. On the other hand, recent studies on the TCP congestion control to achieve RTT fairness are evolving actively. An example is TCP-Libra, which contains the RTT value in its window increase control. However, this method does not reflect RTT increase sufficiently when packets are buffered at a bottleneck router. Therefore, in this paper, we separate a resource allocation problem into two phases: fair allocation of bottleneck link capacity and that of buffer space at the bottleneck router. We then propose a new congestion control which switches two modes according to observed RTT values. Experiments are carried out to validate the proposed method and much better performances in RTT-fairness are achieved against conventional methods.

Keywords: Networks, Transport protocol, Internet, RTT fairness

1 INTRODUCTION

TCP (Transmission Control Protocol) is widely used in the current network and provides end-to-end, reliable congestion control. Although this TCP is originally designed for wired networks, many researches have been studied to extend TCP to be adapted to wireless networks [1,2].

The majority of data services from web surfing to HTTP multimedia streaming (like YouTube and P2P streaming) in the Internet are carried by TCP. In principle, an AIMD (Additive Increase and Multiplicative Decrease) behavior of TCP-Reno's congestion avoidance mechanism [3] is widely adopted, of which equivalent rate can be estimated from observable information (RTT and packet loss rate) [4,5]. However, since the AIMD mechanism of original TCP-Reno autonomously determines a sending rate according to the self-clocking principle, it is well-known that it suffers from RTT unfairness. Therefore, the steady state analysis of the case in which multi-flows having different RTT compete on the same bottleneck link is studied in [6]. And, RTT-fairness had been focused in many TCP papers such as TCP-Vegas [7], FAST-TCP [8], TCP-Libra [9] and so on. Delay-based protocols (TCP-Vegas and FAST-TCP) which use RTT to control their window size have a critical problem about friendliness with existing protocol (TCP-Reno). TCP-Libra achieves RTT-fairness and improves friendliness with TCP-Reno simultaneously. However, we will show that it causes unfairness in buffer space allocation when packets are buffered.

In this paper, we separate a resource allocation problem into fair allocation of the bottleneck link capacity and that of the router buffer space. We then propose a new congestion control which switches two modes according to observed RTT values. We combine the ideas of TCP-Libra [9], TCP-Alpha (explained in Section 2), and TCP-Westwood (TCPW) [10] in our proposal. We will show our approach brings much better performances in RTT-fairness, friendliness with TCP-Reno, and throughput efficiency. Our simulation experiments are carried out assuming wired networks. However, since our target covers both wired and wireless networks, we also explain from the viewpoint of wireless networks.

This paper is organized as follows: Section 2 presents research background. Section 3 explains our analysis model on RTT fairness. Section 4 introduces our proposal, and Section 5 demonstrates experimental results. Finally, Section 6 provides conclusions of this paper.

2 RESEARCH BACKGROUNDS

In this section, we explain RTT-unfairness of the AIMD congestion control, and introduce TCP-Westwood, TCP-Libra and TCP-Alpha, respectively.

2.1 RTT-unfairness of AIMD

A window increase rate of the AIMD congestion control based on TCP-Reno is proportional to RTT values in principle. The increase rates of long RTT flows are slow and inversely proportional to RTT values. For example, [11] provides an analytical result of RTT unfairness of the AIMD congestion control, in which throughput ratio of two TCP flows having different RTT values is given by

$$\frac{w_1}{w_2} \propto \left(\frac{RTT_2}{RTT_1} \right)^{\frac{d}{1-d}} \quad (1)$$

where w_i is an average congestion window size (corresponding to throughput) of flow i (i=1,2), RTT_i is an average RTT of flow i, and d is a constant which is determined by the congestion control mechanisms (e.g. d is 0.5 for TCP-Reno and BIC-TCP, 0.82 for High-speed TCP and 1.0 for Scalable TCP). This equation proves RTT unfairness, according to which TCP flows with smaller RTT values expel TCP flows with longer RTT values.

2.2 TCP-Westwood

We mention TCPW-RE (Rate Estimation) [10] which improves throughput efficiency in window decrement phase when a packet loss is detected.

In congestion avoidance, the behavior of TCPW is the same as TCP-Reno. But the decrease parameter after a packet loss is expressed by

$$cwnd = max\left(\frac{RTT_{min}}{RTT}\cdot cwnd, \frac{cwnd}{2}\right) \quad (2)$$

where RTT_{min} and RTT are the minimum RTT and RTT just before the packet loss, respectively. Due to the first term of Eq.(2), TCPW-RE just clears the router buffer instead of halving the congestion window and causes no vacant capacity. Due to this fact, TCPW-RE can achieve more throughput efficiency than TCP-Reno in lossy link, and prevent critical reduction of window size by unexpected packet losses which occur particularly in wireless networks.

2.3 TCP-Libra

TCP-Libra, which was proposed in [9], achieves RTT-fairness by introducing an observed RTT value into its window increase control.

Though the original formulation is more complicated, we can simplify the TCP-Libra's congestion control algorithm as

$$cwnd += k\cdot RTT^2 \quad (3)$$
$$cwnd += cwnd/2 \quad (4)$$

where k is a parameter of RTT (which is assumed to be a constant in this paper). TCP-Libra updates its window size using Eq.(3) as long as acknowledge packets are received successfully. Eq.(4) is used when a packet loss is detected.

2.4 TCP-Alpha

We define "TCP-Alpha", which is a linear version of the TCP-Libra's algorithm briefly mentioned in [12]. TCP-Alpha updates its congestion window size by

$$cwnd += k'\cdot RTT \quad (5)$$

where k' is a constant similar to k in Eq.(3). When a packet loss happens, TCP-Alpha halves its window size similar to Eq.(4).

3 ANALYSIS MODEL

We analyze single flow's window size behavior until a packet loss happens by overflow at a bottleneck router buffer. Assume the network has a single bottleneck link. Let B [pkt/s] represent the bottleneck link capacity and S [pkt] denote the buffer size of the bottleneck router. In this model, we use TCP-Reno, TCP-Libra and TCP-Alpha.

We separate the processes until overflow into two parts. Fig.1 shows the window size behavior of TCP-Reno. The first part is *Model I* (before buffering) where there are residual capacity and no delay by buffering. In *Model I*, window size is always less than BDP (Bandwidth Delay Product). The second part is *Model II* (after buffering) where the bottleneck link capacity is fully utilized and there is delay by buffering. We consider this process until packet losses happen due to overflow. We evaluate the window size behavior from two points of view in the following subsections.

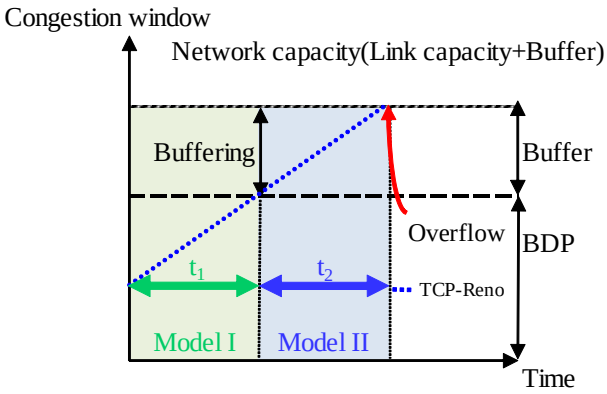

Figure 1: The window size behavior of TCP-Reno.

3.1 *Model I*: before buffering

There are residual capacity on the bottleneck link and no buffering delay in this *Model I*. Thereby, we can set RTT to RTT_{min}. We consider a case that the window size grows up to BDP, which is calculated by

$$BDP_{RTT} = RTT_{min}\cdot B \quad (6)$$

It is clear that BDP depends on RTT value. We define x_{RTT} as the increment amount of the congestion window per RTT. Table 1 shows x_{RTT} of each protocol. For example, TCP-Reno updates its window size by 1 per RTT.

Table 1: Increment amount of each protocol

	TCP-Reno	TCP-Libra	TCP-Alpha
x_{RTT}	1	$k*RTT^2$	$k'*RTT$

We assume the case where there is residual capacity R [pkt/s]. Let t_1 represent the time until the bottleneck link is fully utilized. Relationship between t_1 and R is expressed by

$$\frac{t_1}{RTT_{min}}\cdot x_{RTT} = R\cdot RTT_{min} \quad (7)$$

where the left term means that the number of RTT rounds ($=t_1/RTT_{min}$) is multiplied by increment amount($=x_{RTT}$) and the right term represents the number of packets to fill residual capacity.

From Eq.(7), t_1 is given by

$$t_1 = \frac{R\cdot RTT_{min}^2}{x_{RTT}} \quad (8)$$

Table 2: t_1 of each protocol

	TCP-Reno	TCP-Libra	TCP-Alpha
t_1	$R \cdot RTT^2{}_{min}$	$\frac{R}{k}$	$\frac{R \cdot RTT_{min}}{k'}$

Table 2 shows t_1 calculated by using x_{RTT} in Table 1. In this table, since k and k' of Eqs.(3) and (5) assume the target rate of TCP-Reno having $RTT_{Reno\ min}$[s] which means the minimum RTT value of TCP-Reno without buffering delay, each parameter is given by below;

$$k = \frac{1}{RTT^2} = \frac{1}{RTT_{\mathrm{Re}no\ min}{}^2} \tag{9}$$

$$k' = \frac{1}{RTT} = \frac{1}{RTT_{\mathrm{Re}no\ min}} \tag{10}$$

These parameters are constant values in particular.

As a result, in *Model I*, the time until link capacity is fully utilized is constant when TCP-Libra is implemented. The larger RTT a flow has, the longer time it takes to fill residual capacity in both case of TCP-Reno and TCP-Alpha.

To evaluate our model, using elapsed time t[s], congestion window size(*cwnd*) is calculated by;

$$cwnd = (B - R) \cdot RTT_{min} + \frac{t}{RTT_{min}} \cdot x_{RTT} \tag{11}$$

3.2 Model II: after buffering

We consider the process from packets buffering start to overflow at the bottleneck router buffer. There are extra delay by buffering and we can express $RTT=RTT_{min}+\alpha$. In our model, we use *average_RTT* instead of *RTT*. *average_RTT* is calculated as below by

$$average\_RTT = \frac{(RTT_{min} + \frac{x_{RTT}}{B}) + (RTT_{min} + \frac{2 \cdot x_{RTT}}{B}) + \cdots + (RTT_{min} + \frac{k \cdot x_{RTT}}{B})}{k} \tag{12}$$

First, buffer size is given by S [pkt] and let t_2 represent the time from buffering start to overflow happened. There is a relationship below right before overflow;

$$\frac{t_2}{average\_RTT} \cdot x_{RTT} = S \tag{13}$$

where left function means that the number of RTT rounds ($=t_2/average\_RTT$) is multiplied by increment amount x_{RTT}. The function expresses the buffering size right before overflow happened.

From Eq.(13), t_2 is given by

$$t_2 = \frac{average\_RTT \cdot S}{x_{RTT}} \tag{14}$$

Since we treat *RTT* and *average_RTT* as the same value in our model, t_2 can be expressed shown in Table 3 using Table 2. Let k and k' in this table also assume the target rate of TCP-Reno having RTT_{Reno}, we set $RTT_{Reno}=RTT_{Reno\ min}+\alpha$ in Eqs. (3) and (5) to trace TCP-Reno behavior considering delay by buffering;

$$k = \frac{1}{RTT^2} = \frac{1}{\left(RTT_{\mathrm{Re}no\ min} + \alpha\right)^2} \tag{15}$$

$$k' = \frac{1}{RTT} = \frac{1}{RTT_{\mathrm{Re}no\ min} + \alpha} \tag{16}$$

Table 3: t_2 of each protocol

	TCP-Reno	TCP-Libra	TCP-Alpha
t_2	$S \cdot RTT$	$\frac{S}{k \cdot RTT}$	$\frac{S}{k'}$

Table 3 shows that the larger RTT value a flow using TCP-Reno has, the longer time it takes to saturate the buffer space. In contrast, in case of TCP-Libra, the trend is the reverse. A flow using TCP-Alpha takes constant t_2 regardless of having any RTT values.

To evaluate our model, using elapsed time t[s] from buffering start, congestion window size (*cwnd*) is calculated by

$$cwnd = B \cdot RTT_{min} + \frac{t}{average\_RTT} \cdot x_{RTT} \tag{17}$$

4 PROPOSAL

In this section, we present our proposal which achieves RTT-fairness along with improving throughput efficiency and keeping friendliness with conventional method (=TCP-Reno). Our proposal is composed of three functions. Two of them are in congestion avoidance and the other function is used when packet loss is detected. Since we would like to allocate common resources fairly to achieve RTT-fairness, our proposal switches adaptively Libra mode and Alpha mode according to observed RTT (i.e. packet buffering status). Using TCP-Westwood algorithm in window reduction phase at a packet loss, our proposal keeps efficiency (i.e. clear the router buffer) even if we use it on wireless networks. Three functions are expressed as follows;

$$cwnd += k \cdot RTT^2 \quad if\ there\ is\ no\ delay \tag{18}$$

$$cwnd += k' \cdot RTT \quad if\ the\ buffering\ starts \tag{19}$$

$$cwnd = max\left(\frac{RTT_{min}}{RTT} \cdot cwnd, \frac{cwnd}{2}\right) \quad if\ the\ packet\ loss\ is\ detected \tag{20}$$

where k and k' are the parameters of RTT which determine the standard target rate of this proposal. At first, the proposal updates its window size same as TCP-Libra using Eq.(18) when residual capacity exists (*Model I*). Next, when RTT increase is observed (*Model II*), the proposal switches the mode to update its window size similar to TCP-Alpha using Eq.(19). Finally, when a packet loss happens, the proposal decreases its window size similar to TCP-Westwood using Eq.(20) and returns to the Libra mode or the Alpha mode according to network status. Fig. 2 shows the total behavior.

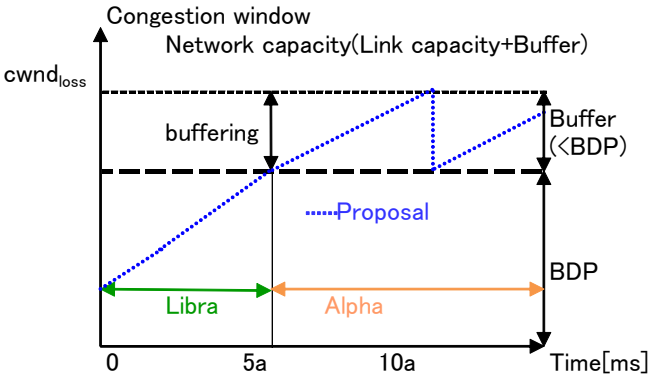

Figure 2: The proposal congestion window behavior.

5 SIMULATIONS

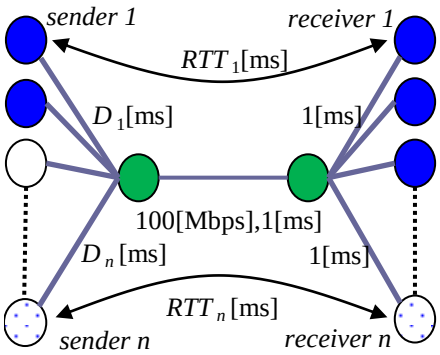

Figure 3: Simulation topology.

We carried out simulation evaluations using ns-2[13]. Fig. 3 shows simulation topology used in our experiments. There are *n*-flows competing on the bottleneck link. Sender *i* communicates with receiver *i* (*i*=1,2,⋯,*n*). Each sender is connected to 1Gbps link of which propagation delay is D_i which is varied according to RTT_i. Each receiver is connected to 1Gbps link with 1ms propagation delay. Link speed and propagation delay of the shared (bottleneck) link are 100Mbps and 1ms, respectively. In this simulation, parameters *k* and *k'* which assume RTT of a competing TCP-Reno flow is $40*10^{-3}$[s]. We use the DropTail buffer management at the bottleneck router. Random packet losses also happen in the bottleneck link.

5.1 Simulation Evaluations

We evaluate the validity of our model introduced in section 3. First, we demonstrate the model case using ns2 simulation and prove the validity of our model by comparing the simulation results with our model. For this purpose, we consider the behavior of two flows having different RTTs. RTT of one flow is $40*10^{-3}$[s] and that of the other is $80*10^{-3}$[s]. We set *B* as follows;

$$B = \frac{100 \cdot 10^6}{1500 \cdot 8} = 8333.3\dot{3}[pkt/s]$$

where we assumes 1500[byte] as the packet size. Let *S* be equal to 500[pkt]. We use three TCP protocols, TCP-Reno, TCP-Libra, and TCP-Alpha.

5.2 Model I Validation

When we compare Eq.(11) and simulation result, let *R* be equal to *B*/2 [pkt/s]. Namely, at the starting time, a flow gets half utilization of bottleneck link capacity. In the end time, window size is equal to BDP. The window size behavior in this period is shown in Fig.4 using ns2 simulation and in Fig.5 calculated by Eq.(11) of our model. We add the vertical line (t_1) in these graphs which means the time when the flow having smaller RTT (=$40*10^{-3}$[s]) fills up the residual capacity on the bottleneck link. In *Model I*, *k* and *k'* in Eqs.(9) and (10) are set to the RTT of TCP-Reno having $40*10^{-3}$[s]. In Fig.4, all protocols' flows with $40*10^{-3}$[s] behave the same due to (9) and (10) which trace TCP-Reno having $40*10^{-3}$[s]. t_1 of these flows are about 6.5[s]. However, window size behaviors of flows with $80*10^{-3}$[s] are different from the others. They show that t_1 of TCP-Reno is approximately 4 times longer. Since Alpha increases its window size in the same rate of flows with $40*10^{-3}$[s], it takes two times to fill up the residual capacity compared to flows with $40*10^{-3}$[s]. Finally, t_1 of TCP-Libra is the same as flows with $40*10^{-3}$[s]. Fig. 4 obviously suggests that the time of TCP-Libra until full utilization is constant regardless of RTT values in *Model I*. Compared to Fig. 5, these graphs reveal the same behaviors, in other word, validity of *Model I* is proved.

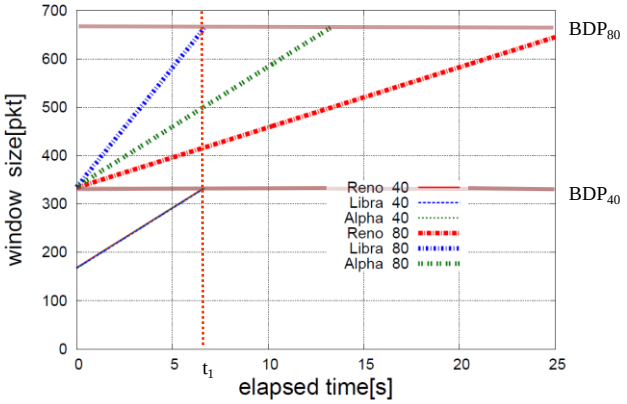

Figure 4: Congestion window size behaviors in simulation.

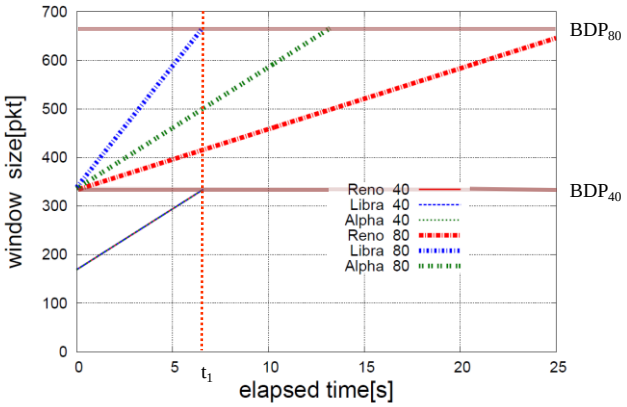

Figure 5: Congestion window size behaviors in our model (*Model I*).

5.3 Model II Validation

Congestion window size behaviors from buffering start to overflow are shown in Fig.6 using ns2 simulator and in Fig.7 using Eq.(17). The window size right before overflow is *BDP*+*S*[pkt]. In *Model II*, k and k' in Eqs.(15) and (16) are set to the RTT of TCP-Reno with $40*10^{-3}+\alpha$[s] including delay by buffering. We add the vertical line (t_2) in these graphs which means the time when the flow having smaller RTT ($40*10^{-3}$[s]) reaches overflow from the buffering start.

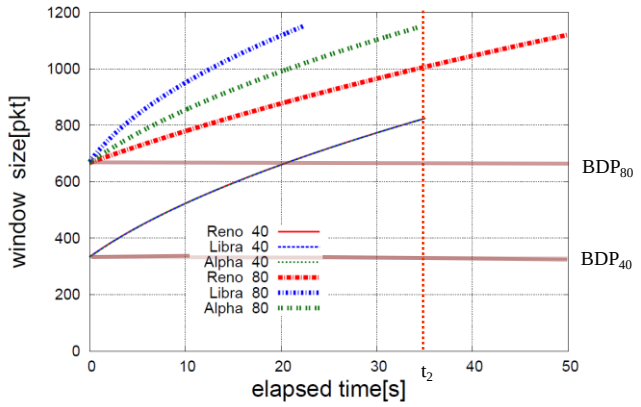

Figure 6: Congestion window size behaviors in simulation.

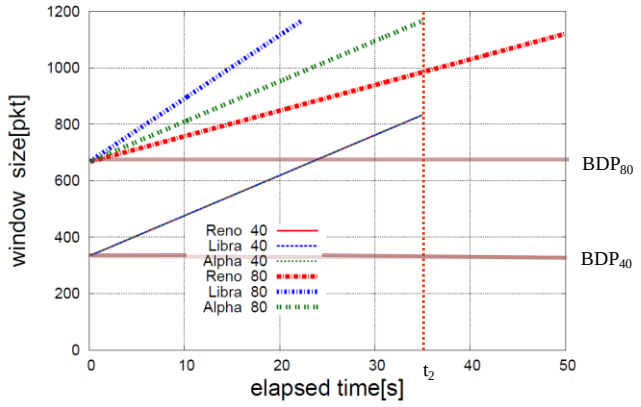

Figure 7: Congestion window size behaviors in our model (*Model II*).

In Fig.6, congestion window size behaviors show curved lines because increment amounts in one second become slow as the number of buffering packets increase and *RTTs* increase. All protocol flows with $40*10^{-3}$[s] behave the same. Each protocol flows with $80*10^{-3}$[s] are different. A flow using TCP-Reno takes longer time to fill the buffer space. TCP-Libra rapidly fills the buffer space compared to the others. In case of TCP-Alpha, since the increment rate of $80*10^{-3}$[s] is the same as that of $40*10^{-3}$[s], it shows that t_2 is constant regardless of different RTTs in *Model II*.

Compared to our model Fig.7, there are no large differences except the linearity of lines. Therefore, simulation results support the validity of *Model II*.

As a result, from the verifications using our models and simulations, it is clearly observed that TCP-Libra would bring fair allocation of residual capacity and TCP-Alpha would bring fair allocation of buffer space.

5.4 RTT-Fairness

The purpose of this subsection is to inspect RTT-fairness. We evaluate RTT-fairness from three points of view. In all cases, two flows of the same protocol (Proposal, Reno, Libra, Alpha) run in the simulation topology shown in Fig. 3.

Firstly, we consider the network status changing packet loss rates between 10^{-2} and 10^{-6} and analyze throughput ratios of two flows. One flow has smaller RTT ($=40*10^{-3}$[s]) and the other flow has larger RTT ($=120*10^{-3}$[s]). The buffer size of the bottleneck router is 500[pkt]. Fig. 8 shows the ratio of throughputs which is calculated by dividing throughput of the larger RTT flow by throughput of the smaller RTT flow. When packet loss rate is high, there is residual capacity and packets buffering at the bottleneck router rarely occurs. Due to this reason, network condition with high packet loss rates is approximately the same as *Model I* in section 3. Similarly, network condition with low packet loss rates is the same as *Model II* where the bottleneck link capacity is fully used and packets buffering starts. In Fig.8, two Proposal flows equally share link capacity in almost cases. It is observed that TCP-Reno is the most unfair protocol in high packet loss rates (*Model I*). On the contrary, TCP-Libra flows can share bottleneck link capacity fairly in high packet loss rates. However, for low loss rates (*Model II*), larger RTT flow gets more utilization than the smaller RTT flow. Using TCP-Alpha, smaller RTT flow gets about half utilization compared to larger RTT flow. Throughput rate of TCP-Alpha approaches to fair sharing in low packet loss rates. It can be also observed in TCP-Reno. If we assume wireless networks, typical packet loss rates are around 10^{-6}~10^{-3}[14]. Therefore, our proposal is expected to bring good performance also in wireless networks.

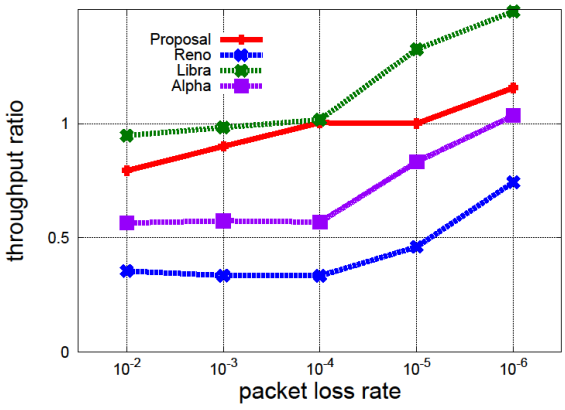

Figure 8: Throughput ratios when packet loss rate varies.

Secondly, we focus on throughput fluctuations when RTT value is changed. One flow has constant RTT ($=40*10^{-3}$[s]). RTT of the other flow varies between $20*10^{-3}$[s] and $120*10^{-3}$[s]. The ratio of throughputs of various RTTs to throughputs of constant RTT is shown in Fig. 9. Buffer size at the router is equal to 500[pkt]. Link packet loss rate is equal to 10^{-5}. Fig. 9 indicates that TCP-Reno and Alpha utilize larger bandwidth as the flow has smaller RTTs. The smaller RTT these protocols have, the faster they increase their window sizes. TCP-Libra has the opposite trend. In

case of our Proposal, it can be noted that as varied RTT gets larger RTT, Proposal keeps fair share. However, when we set $20*10^{-3}$[s] to the value of varied RTT, Proposal behave the same as rather TCP-Alpha than TCP-Libra. In this scenario, since buffer size is high for small RTT, it would be almost the same as *Model II*.

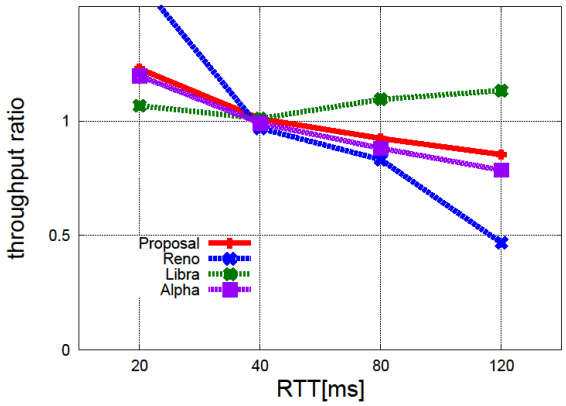

Figure 9: The ratios of throughputs of varied RTT flows to throughputs of constant RTT flows (=40[ms]).

Thirdly, we evaluate RTT-fairness when changing buffer sizes at a bottleneck router. We fixed 10^{-5} to the packet loss rate. Two flows of smaller RTT (=$40*10^{-3}$[s]) and larger RTT ($120*10^{-3}$[s]) compete. In Fig.10, throughput ratio is calculated by dividing throughput of the flow with larger RTT by that of the flow with smaller RTT. When the buffer size is small, the network state approaches to *Model I*. The network state approaches to *Model II* when the buffer size becomes large. First, it is indicated from this graph that throughput ratio of TCP-Reno and TCP-Alpha is almost constant and the ratio is lower than 1. Second, the ratio of throughput using TCP-Libra is more than 1. Finally, Proposal keeps middle positions between TCP-Libra and TCP-Alpha. Thus, Proposal keeps fair allocation of the network resources. On the other hand, TCP-Libra in large buffer size approaches to fair rate. The trend is similar in previous simulations. This is because the RTT value is small compared to the delay value caused by buffering so that total delay of each flow is not so different relatively.

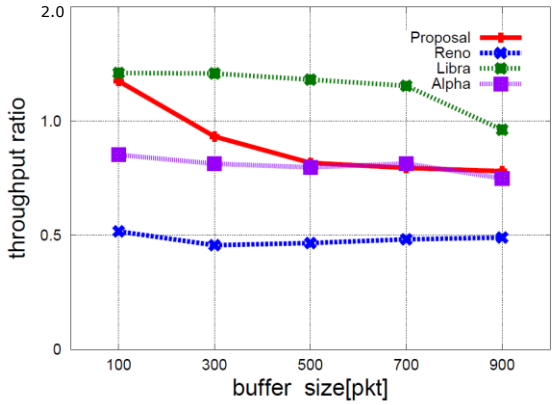

Figure 10: Throughput ratios when buffer size varies.

5.5 Throughput Efficiency

In Fig. 11, throughputs of a single TCP flow are shown along with TCP-Westwood. For the network simulation setting, RTT is varied between $40*10^{-3}$[s] and $200*10^{-3}$[s] and random packet loss rate is set to 10^{-6}. Buffer size at the router is constant (500[pkt]). Bottleneck bandwidth is 1[Gbps] and the other links are 2[Gbps].

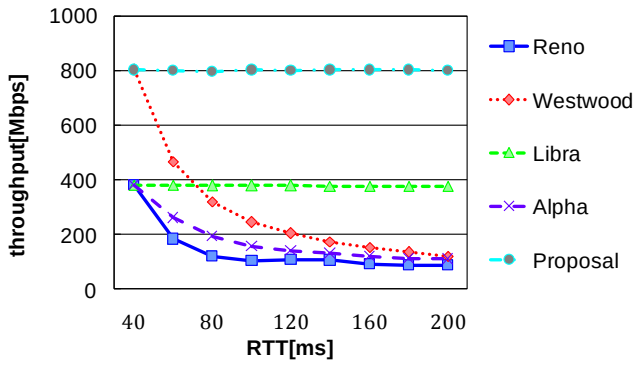

Figure 11: Throughputs of various TCPs when RTT values are varied.

The result shows that Proposal proves its high speed efficiency and robustness in high speed and long delay networks. Due to its window decrement mechanism which halves window sizes upon a packet loss, TCP-Reno and TCP-Alpha decrease their link utilization as soon as RTTs become larger and cannot increase their window sizes quickly. In high speed and long delay networks, there are severe effects for them. On the other hand, when RTT is lower than $60*10^{-3}$[s], Proposal and TCP-Westwood achieve more bandwidth than the others because both of them are robust due to their window decrement mechanism when packet loss is detected. But, in case of TCP-Westwood, its throughput decreases suddenly when RTT is larger because of its conservative window increment mechanism. TCP-Libra keeps its throughput around 400[Mbps] in spite of RTT values.

5.6 Friendliness *(Inter-Protocol Fairness)*

This subsection focuses on inter-protocol fairness with TCP-Reno. The throughputs of TCP-Reno having constant RTT (=$40*10^{-3}$[s]) are shown in Fig. 12 and Fig. 13 along with those of the competing flows (Reno or Proposal) having different RTTs. Buffer size is equal to BDP of which RTT is between $20*10^{-3}$[s] and $60*10^{-3}$[s]. Link packet loss rate is equal to 10^{-4}.

Comparing Fig. 12 with Fig. 13, these graphs show that the total throughput in Fig. 13 is larger than that in Fig. 12 because Proposal is efficient. Fig. 12 shows that TCP-Reno of RTT=$20*10^{-3}$[s] expel the bandwidth of the competing flow. On the other hand, TCP-Reno of RTT=$60*10^{-3}$[s] cannot utilize bandwidth sufficiently. The throughputs of TCP-Reno competing with Proposal are almost constant in spite of varying RTTs. It can be said that Proposal utilizes bandwidth efficiently without disturbing TCP-Reno regardless of changing RTT. Namely, Proposal achieves friendliness.

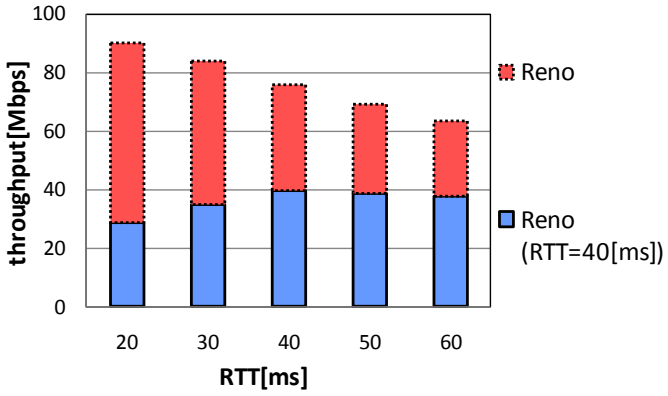

Figure 12: Throughputs of a TCP-Reno flow when it competes with TCP-Reno having different RTTs.

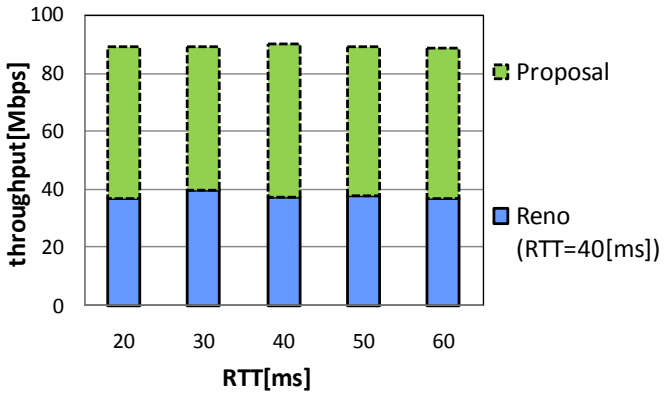

Figure 13: Throughputs of a TCP-Reno flow when it competes with the proposal flow having different RTTs.

6 CONCLUSION

In this paper, we analyze the sharing process of common resources and present new congestion control algorithm which achieves RTT-fairness. Then we prove that our proposal achieves RTT-fairness, throughput efficiency, and friendliness with TCP-Reno. However, in case of wireless networks, we should consider several effects (random packet loss, RTT variation, handover) for applying our proposal into implementation in these networks. Practically, since our proposal requires precise RTT estimates to switch the modes and to update its window size, it sometimes suffers from the environments where RTT frequently changes. As future work, we will experiment in wireless networks and try to develop an automatic estimation method of RTTs of competing TCP-Reno flows (or other TCP congestion control flows) to set adaptive k and k' in (3) and (5).

REFERENCES

[1] G. Huston: "TCP in a wireless world," IEEE Internet Computing, Vol. 5, No. 2, pp. 82–84, 2001.

[2] H. Balakrishnan, V. Padmanabhan, S. Sesha, and R. Katz: "A Comparison of Mechanisms for Improving TCP Performance over Wireless Links," IEEE/ACM Trans. on Networking, December 1997.

[3] W. Richard Stevens: "TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms," IETF RFC 2581, 1997.

[4] S.Floyd and K.Fall, "Promoting the Use of End-to-end Congestion Control in the Internet," IEEE/ACM Trans. on Networking, Vol. 6, Aug.1999.

[5] J.Padhye, V.Firoiu, D.Towsley, and J.Kurose: "Modeling TCPThroughput: A Simple Model and its Empirical Validation," ACM SIGCOMM 1998, Sept. 1998.

[6] H. Hisamatu, H. Ohsaki, and M. Murata. "Steady State Analysis of TCP Connections with Different Propagation Delays", IEICE Tech. Report, IN2002-97, Oct.2002 (in Japanese).

[7] L.S.Brakmo and L.L.Peterson: "TCP Vegas: End-to-End Congestion Avoidance on a Global Internet," IEEE Journal on Selected Areas in Commun., Vol. 13, No.8, pp.1465-1480, Oct.1995.

[8] C.Jin, D.X.Wei and S.H.Low: "FAST TCP: Motivation, Architecture, Algorithms, Performance", IEEE INFOCOM 2004, Mar.2004.

[9] G.Marfia et al.: "TCP-Libra: Exploring RTT Fairness for TCP," UCLA Comp. Science Dept. Tech. Report # UCLA-CSD TR-050037.

[10] C.Casetti, M.Gerla, S.Mascolo, M.Y.Sanadidi, and R.Wang: "TCP Westwood: Bandwidth Estimation for Enhanced Transport over Wireless Links", In proc. of ACM Mobicom 2001, Jul.2001.

[11] L. Xu, K. Harfoush and I. Rhee: "Binary Increase Congestion Control (BIC) for Fast, Long Distance Networks", in Proc. of INFOCOM 2004.

[12] S. Floyd, V.Jacobson. On Traffic Phase Effects in Packet-switched Gateways. ACM SIGCOMM Computer Communication Review, 21(2):26-42, 1991.

[13] "ns-2 network simulator(ver.2)," http://www.mash.cs.berkley.edu/

[14] L. K. Tee: "Packet Error Rate and Latency Requirements for a Mobile Wireless Access System in an IP Network" IEEE VETECF 2007, Oct.2007