Improvement of RTT-Fairness in Hybrid TCP Congestion Control

Kazumine OGURA, Tomoki FUJIKAWA, Su ZHOU, and Jiro KATTO
Dept. of Computer Science Waseda University
3-4-1 Okubo, Shinjuku-ku, Tokyo, 169-8555 Japan
{ogura, katto}@katto.comm.waseda.ac.jp

***Abstract*—This paper presents TCP-Fusion supporting RTT (Round Trip Time) fairness in addition to throughput efficiency and friendliness to TCP-Reno. When multiple TCP flows having different RTT values compete, more bandwidth is unfairly allocated to the flow having smaller RTT. 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 and throughput efficiency are evolving actively. An example for RTT fairness is TCP-Libra and an example for throughput efficiency is Hybrid TCP congestion control. This paper focuses on Hybrid TCP (exploiting residual link capacity when TCP-Reno drops its rate) and improves its RTT fairness by incorporating the ideas of TCP-Libra (congestion window increase in proportion to square of RTT) to its loss mode. Experiments are carried out to validate the proposed method and much better performances in RTT fairness and throughput are provided against conventional methods.**

Keywords-component; TCP; RTT-fairness; Hybrid

I. INTRODUCTION

TCP (Transmission Control Protocol) is widely used in current network and provides end-to-end, reliable congestion control. The majority of data services from web surfing to HTTP multimedia streaming (like YouTube) in the Internet are carried by TCP. In principle, an AIMD (Additive Increase and Multiplicative Decrease) behavior of TCP-Reno's [1] congestion avoidance mechanism is widely adopted, of which equivalent rate can be estimated from observable information (RTT and packet loss rate) [2,3].

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. Namely, when multiple flows having different RTT values compete, fair share of bandwidth is impossible because they increase their congestion windows by their different paces [3]. As a result, a user joining longer RTT session may not be able to obtain sufficient bandwidth, resulting that the user can not connect internet comfortably.

RTT fairness had been focused in many TCP papers such as TCP-Vegas [4], FAST-TCP [5], CUBIC-TCP [6], TCP-Hybla [7] and TCP-Libra [8].

On the other hand, joint improvement of throughput efficiency and inter-protocol friendliness to TCP-Reno had been focused in Compound-TCP [9], Adaptive-Reno [10], TCP-Illinois [11], YeAH-TCP [12] and our TCP-Fusion [13], some of which also refer to RTT fairness. They are called Hybrid TCP congestion control because they switch two modes, loss-based mode and delay-based mode, according to network conditions.

In this paper, we try to develop congestion control algorithms supporting both of RTT-fairness and throughput efficiency. Initially to improve throughput efficiency, we apply the idea of TCP-Fusion congestion control we had proposed [13]. Then, to achieve RTT fairness, we focus on incorporation of the idea of TCP-Libra to manage its congestion window increase in the loss mode.

This paper is organized as follows: Section II presents research backgrounds. Section III presents our proposal. Section IV demonstrates experimental results. Finally, Section V provides conclusions of this paper.

II. RESEARCH BACKGROUNDS

Firstly, the AIMD congestion control is introduced as a typical congestion control algorithm. Secondly, we introduce our TCP-Fusion as an example of the Hybrid TCP. Finally, we describe TCP-Libra which fulfills RTT-fairness by its window increase mechanism.

A. AIMD Congestion Control

A window increase rate of the AIMD congestion controls based on TCP-Reno is proportional to RTT values in principle. For example, [14] provides an analytical result of RTT unfairness of the AIMD congestion controls, in which throughput ratio of two TCP flows having different RTT values is given by

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

where w_i is an average congestion window size 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).

B. *TCP-Fusion*

TCP-Fusion [13] is one of Hybrid TCP, which has been originally proposed to achieve higher efficiency in fast long-distance network while still maintaining inter-protocol friendliness to TCP-Reno.

1) *Congestion Window Reduction.*

TCP-Fusion adopts optimization of the decrease parameter based on TCPW-RE (Rate Estimation) [15] to improve efficiency particularly in the leaky pipe. In TCPW-RE, the decrease parameter after a loss can be expressed as RTT_{min}/RTT [16], where RTT_{min} and RTT are the minimum RTT and the RTT right before the packet loss, respectively. This equation indicates that TCPW-RE reduces its congestion window size to AR (Achieved Rate) [17] to clear the router buffer and, as a result, it improves throughput efficiency against TCP-Reno. Thus, congestion window reduction of TCP-Fusion is implemented as follows;

$$cwnd_{new} = \max\left(\frac{RTT_{\min}}{RTT} cwnd_{last}, \frac{cwnd_{last}}{2}\right) \tag{2}$$

where $cwnd_{new}$ and $cwnd_{last}$ are congestion window sizes right after and before the packet loss, respectively.

2) *Congestion Window Increase.*

Similar to TCP-Vegas, TCP-Fusion has three phases; increase phase, decrease phase, and steady phase, which are switched by a number of packets in the bottleneck queue (*diff*). The *diff* can be estimated as;

$$diff = cwnd \frac{(RTT - RTT_{\min})}{RTT} \tag{3}$$

Using the *diff*, congestion window increase of TCP-Fusion is carried out by

$$\begin{aligned} &cwnd_{new} = \\ &\begin{cases} cwnd_{last} + W_{inc} / cwnd_{last}, & \text{if } diff < \alpha \\ cwnd_{last} + (-diff + \alpha) / cwnd_{last}, & \text{if } diff > 3 * \alpha \\ cwnd_{last}, & \text{otherwise} \end{cases} \\ &cwnd_{\text{new}} = reno\_cwnd, \ \text{if } cwnd_{new} < reno\_cwnd \end{aligned} \tag{4}$$

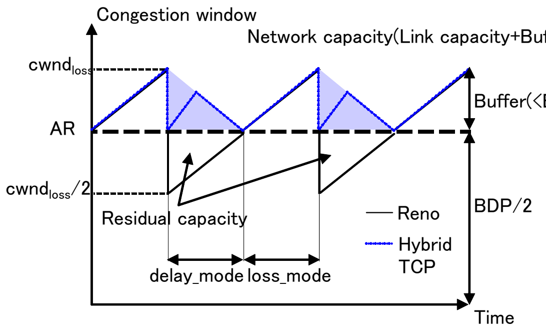


Figure 1. Comparison of congestion window behaviors of TCP-Reno and TCP-Fusion

where $cwnd_{new}$, $cwnd_{last}$ and *reno_cwnd* are the congestion window sizes after and before update and of an equivalent to TCP-Reno, respectively. α is the *lower bound threshold* to switch three phases. W_{inc} is the increment parameter to increase congestion window size rapidly. The congestion window behavior of TCP-Fusion is like Fig. 1 when TCP-Fusion and TCP-Reno compete with a buffer size less than BDP.

C. *TCP-Libra*

Let α and β be window increase rate and window decrease rate, respectively. Then, an average sending rate of the conventional AIMD protocol like TCP-Reno is given by

$$R_{AIMD} = \frac{1}{RTT} \sqrt{\frac{\alpha}{\beta} \cdot \frac{1-p}{p}} \tag{5}$$

where R is an average sending rate and p is an average packet loss rate (α is 1 and β is 0.5 for TCP-Reno). TCP-Libra [8] defines RTT_0 and RTT_1 to be constants which satisfy (RTT_0, $RTT_1 >> RTT$) and γ to be a parameter of RTT (which is assumed to be a constant in this paper), and let α and β be

$$\alpha = \gamma \cdot \frac{RTT^2}{RTT + RTT_0} \approx \gamma \cdot \frac{RTT^2}{RTT_0} \tag{6}$$

$$\beta = \frac{RTT_1}{2(RTT + RTT_0)} \approx \frac{1}{2} \tag{7}$$

Different from the conventional AIMD mechanism, TCP-Libra weights α to be proportional to square of RTT. Then, an average sending rate of TCP Libra is provided by

$$R_{Libra} = \sqrt{\frac{2\gamma}{RTT_0} \cdot \frac{1-p}{p}} \tag{8}$$

Since RTT_0 is a constant, R becomes constant irrespective of different RTT values (i.e. RTT fairness).

III. Proposals

We propose TCP-Fusion with RTT fairness capability by applying TCP-Libra's window increase mechanism into the TCP-Fusion's loss_mode which operates when there is no residual capacity as mentioned above. Except for this, proposed congestion control is equal to TCP-Fusion.

A. *TCP-Fusion with RTT Fairness*

We consider the case that a TCP-Reno flow and a proposed TCP flow having different RTT values are competing. When there exists residual capacity before the TCP-Reno flow reaches *AR*, the proposed TCP flow operates in *delay_mode* and puts the constant number of packets into a router buffer which is expected to be RTT fair (as long as estimated *AR* is RTT fair). Therefore, we can focus on *loss_mode* when packet buffering starts. In this phase, original TCP-Fusion carries out sending rate increase by *1MSS/RTT* which is same as TCP-Reno. Since TCP-Libra modifies its window increase rate α to be proportional to RTT^2, we can modify it to $k \cdot RTT(i)^2$ and achieve

$$R(i+1)=\frac{\left(R(i)\times RTT(i)+k\cdot RTT(i)^2\right)}{\left(RTT(i)+\Delta RTT(i)\right)} = \frac{\left(R(i)+k\cdot RTT(i)\right)}{\left(1+\Delta RTT(i)/RTT(i)\right)} \quad (9)$$

where $R(i)$ is a sending rate of the i-th RTT round, and k is a tunable parameter corresponding to RTT of the competing TCP-Reno flow as discussed later. Since we can assume $RTT(i) >> \Delta RTT(i)$, (9) can be approximated by

$$R(i+1)\approx R(i)+k\cdot RTT(i) \quad (10)$$

This indicates that the rate is increased by $k \cdot RTT(i)$ per $RTT(i)$ second, which means that $k \cdot RTT(i) / RTT(i) = k$ packets are sent per unit time without depending on RTT.

IV. Experiments

We carried out simulation evaluations using ns-2 [18]. Fig. 2 shows simulation topology used in the experiments. There are n-flows shared in the bottleneck bandwidth. Sender i communicates with receiver i (i=1,2,⋯,n). Each sender is connected to 1Gbps link of which propagation delay is D_i. D_i is varied respectively according to RTT_i. Each receiver is connected to 1Gbps link with 1ms propagation delay. Link speed and propagation delay of a shared (bottleneck) link are 100Mbps and 1ms, respectively. The router buffer size is equal to the BDP. In our proposal, parameter k in (9) is set to $1/(0.04)^2$ which assumes RTT of a competing TCP-Reno flow is 40ms.

A. Efficiency

Fig. 3 shows the throughput of a single TCP flow. For the network simulation setting, RTT is 40ms and random loss rate is varied from 10^{-6} to 10^{-1}. Our proposal is same as TCP-Fusion in this situation. All kind of TCP variant flows can utilize nearly the link bandwidth when the loss rate is smaller than 10^{-5} and degrade its throughput as the loss rate increases. Among of them, Proposal, ARENO, and FAST are more efficient and robust in this lossy link than others. This is because they can update their windows aggressively when there exists residual capacity caused by packet loss. Incidentally, in case of TCP-Libra which we tentatively set γ assuming the parameter of RTT as 40[ms], TCP-Libra behaves same as TCP-Reno.

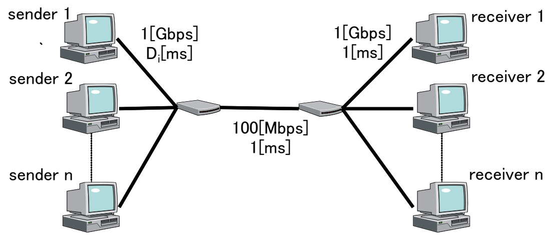


Figure 2. Simulation topology

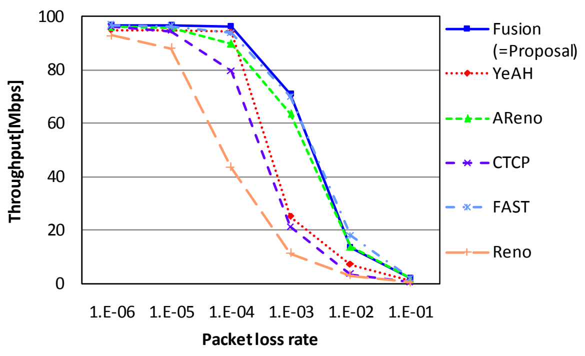


Figure 3. Throughput of a single TCP flow with different packet loss rates

B. Intra-Protocol Fairness

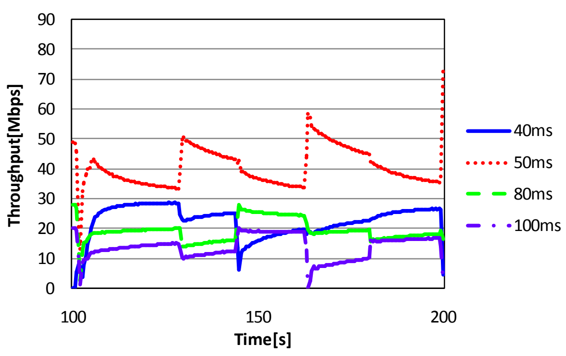
In this subsection, we focus on intra-protocol fairness when multiple flows with different RTTs are competing, and evaluate performance differences of two queuing mechanisms at a router; Drop Tail (DT) and Random Early Detection (RED) [19]. RED can detect congestion before the buffer overflows by dropping or marking packets with a probability that increases with the queue length. The objectives are an equitable distribution of packet loss and reduction of delay variation.

For the experiments, we simulate four TCP flows of the same congestion control having different RTTs and evaluate their RTT fairness. Compared congestion controls are TCP-Reno, TCP-Fusion and our Proposal. We set RTT_i as shown in Table I. The router buffer size in this simulation is equal to BDP of setting 100ms RTT.

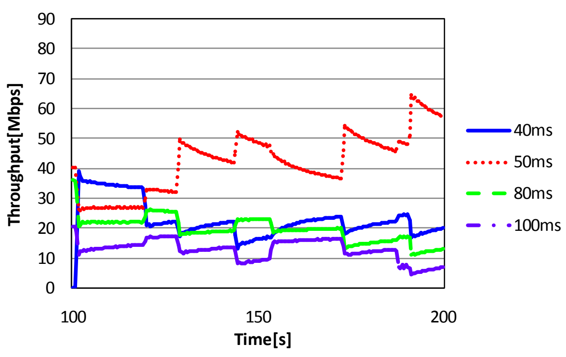
Fig. 4 shows throughput behaviors of each protocol in case of DT. This figure indicates that, in case of TCP-Reno and TCP-Fusion, smaller RTT flows (RTT_1 and RTT_2) utilize the link capacity larger than longer RTT flows (RTT_3 and RTT_4) as expected. However, in case of our Proposal, only RTT_2 flows dominate the link capacity, and the others show similar behaviors each other. Different from expectation, the relation between RTT values and bandwidth utilization is out of order. We then evaluate the number of packets lost during this simulation, which is shown in Table II. This table indicates that the dominating flow (i.e. 50ms flow) seldom suffers from packet losses compared to the others. This reason is not hard to see. Once one or more flows dominate the link capacity, it is too hard for other flows to take the link capacity as long as router buffer size is large and DT policy is applied (few opportunities to steal the bandwidth).

TABLE I. RTT PARAMETERS USED IN THE SIMULATION

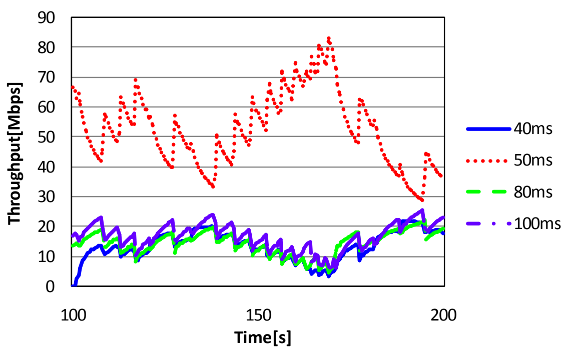
RTT_i	RTT_1	RTT_2	RTT_3	RTT_4
RTT[ms]	40	50	80	100



(a) TCP-Reno



(b) TCP-Fusion



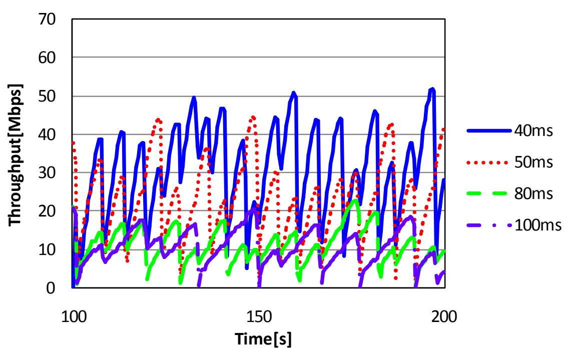
(C) Proposal

Figure 4. Throughput behaviors in the DT case when four TCP flows of different RTT values share a bottleneck link

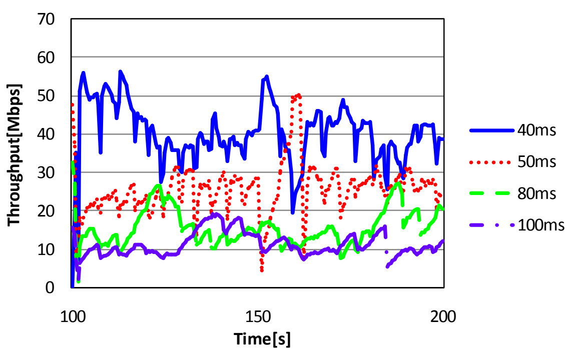
TABLE II. PACKET LOSS COUNTS IN THE DT CASE WHEN FOUR TCP FLOWS OF DIFFERENT RTT VALUES SHARE A BOTTLENECK LINK

RTT[ms]	40	50	80	100	Total
Proposal	79	6	70	135	290
Fusion	14	1	7	6	28
Reno	7	2	4	9	22

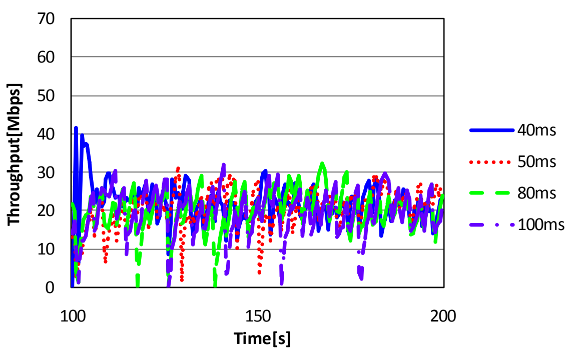
On the other hand, Fig. 5 shows throughput behaviors of each protocol when RED is applied at a router. This figure indicates that, in case of TCP-Reno and TCP-Fusion, smaller RTT flows utilize larger bandwidth than longer RTT flows. The relation between RTT values and bandwidth utilization is also in order. Then, in case of our Proposal, all flows with different RTTs approximately utilize the link capacity in the same rate. Table III shows that the situation like Table II does not happened. This is because RED provides more opportunities for non-dominant flows to steal bandwidth from dominant flows than DT when multiple flows with different RTTs are competing.



(a) TCP-Reno



(b) TCP-Fusion



(c) Proposal

Figure 5. Throughput behaviors in the RED case when four flows of different RTT values share a bottleneck link

We use the variance to evaluation of the RTT-fairness degree, which is calculated by

$$Variance = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2 \tag{11}$$

where x_i is the throughput of the i-th flow and n is the number of flows(is equal to 4 in this case). $\bar{x}$ is the average throughput taken by the four flows. The result is presented in

Fig. 6 which shows the statistical result changing the start time of each flow. As this figure indicates, our proposal is much better than the other protocols in respect of RTT-fairness because the variance of proposal is lower than the others.

TABLE III. PACKET LOSS COUNTS IN THE RED CASE WHEN FOUR TCP FLOWS OF DIFFERENT RTT VALUES SHARE A BOTTLENECK LINK

RTT[ms]	40	50	80	100	Total
Proposal	124	131	149	167	571
Fusion	71	71	40	24	206
Reno	36	35	27	16	114

TABLE IV. FAIRNESS INDEX IN THE RED CASE WHEN FOUR TCP FLOWS OF DIFFERENT RTT VALUES SHARE A BOTTLENECK LINK

	Proposal	Fusion	Reno
FI	0.997487	0.811747	0.833822

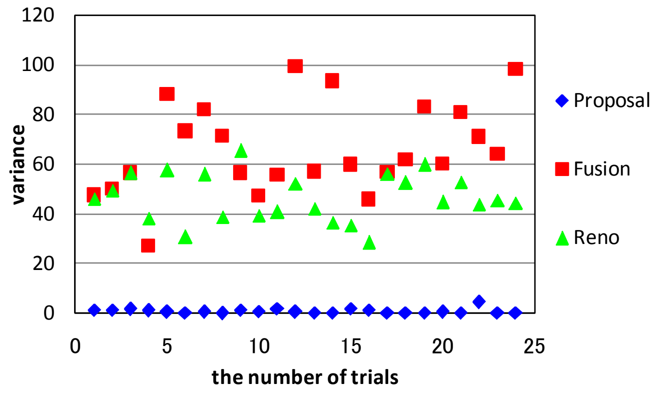


Figure 6. RTT-fairness staticstical anlysis changing the start time of each flow

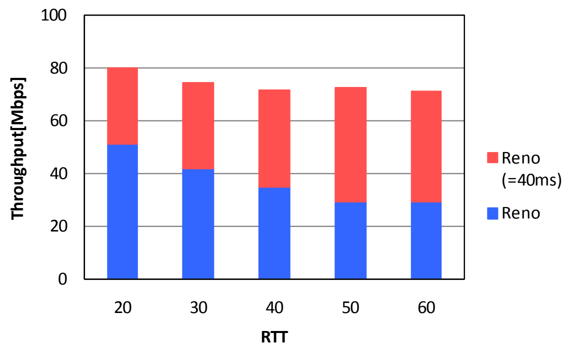
C. *Inter-Protocol Fairness*

The purpose of this subsection is to inspect inter-protocol fairness with TCP-Reno. The throughputs of TCP-Reno having constant RTT (=40ms) are shown in Fig. 7 along with those of the competing flows (Reno, Fusion or Proposal) having different RTTs. Since the preceding subsection suggests that RED is more suitable than DT to achieve RTT fairness by the Proposal, this experiment also uses RED.

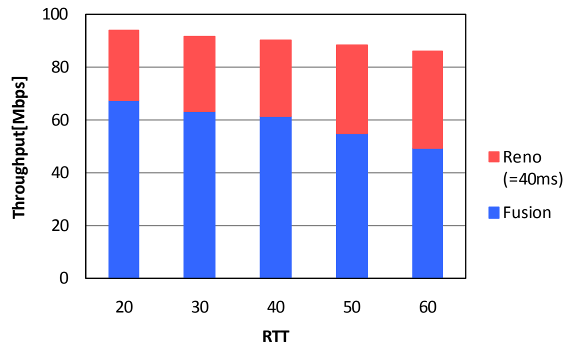
Fig. 7 (a) shows throughputs of two TCP-Reno flows. The figure indicates that the throughput is fair only when both of RTTs are same. When RTT of the competing flow becomes longer, its bandwidth utilization becomes smaller.

In Fig. 7 (b), we notice that TCP-Fusion utilizes larger bandwidth than TCP-Reno because of its throughput efficiency. TCP-Fusion with smaller RTT achieves more bandwidth but this does not mean its unfriendliness. It can be explained that TCP-Fusion in *delay_mode* can utilize residual capacity, which is caused by packet losses from the TCP-Reno flow. Though the router buffer size is equal to BDP, RED contributes to causing residual capacity (packet losses happen before the number of buffered packets reaches BDP). In case of DT, though the results are omitted here, TCP-Fusion doesn't dominate bandwidth when RTTs of TCP-Reno and TCP-Fusion are the same.

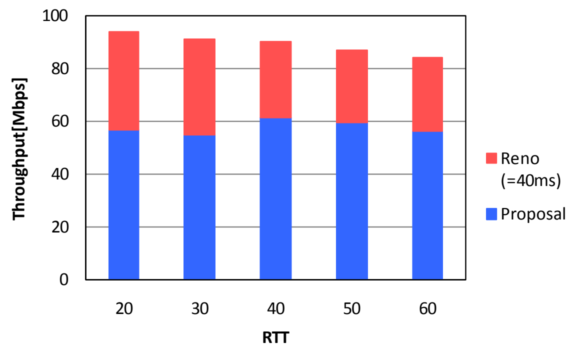
Finally, Fig. 7(c) reveals that throughputs of the proposal is almost constant in spite of varying RTT. There might be a criticism that the proposal provides bandwidth to TCP-Reno when RTT is small and expels TCP-Reno when RTT is large. However, this is not true and can be explained as follows. Our proposal operates in *delay_mode* when residual capacity exists and achieves throughput gains against the competing TCP-Reno flow without causing unfriendliness. When packet buffering starts at a router buffer, our proposal operates in *loss_mode* and shows inter-protocol friendliness.



(a) TCP-Reno



(b) TCP-Fusion



(c) Proposal

Figure 7. Throughputs of a TCP-Reno flow of RTT = 40ms when it competes with other TCP protocol flows having different RTTs.

V. CONCLUSION

This paper presents improvement of TCP-Fusion to achieve RTT fairness while keeping high throughput efficiency and inter-protocol friendliness with TCP-Reno. Simulation results validate our proposal against conventional TCP-Reno and TCP-Fusion. As future work, we will further try to inspect inter-protocol friendliness to TCP-Reno in Fig. 7. In this experiment, we assumes constant *k* in (10). However, this value has to be changed in an adaptive manner according to competing TCP-Reno flows having various RTTs. For this purpose, we should develop an automatic estimation method of RTT of competing TCP-Reno flows.

REFERENCES

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

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

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

[4] 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.

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

[6] I.Rhee and L.Xu: “CUBIC: A New TCP-Friendly High-speed TCP Variant”, PFLDnet 2005, Feb.2005.

[7] C.Caini and R.Firrincieli, “TCP Hybla: A TCP Enhancement for Heterogeneous Networks”, Int. J. Satell. Comm. Network, Vol1.22,pp.547-566, Sep.2004.

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

[9] K.Tan. J.Song, Q.Zhang, and M.Sridharan: “Compound TCP: A Scalable and TCP-Friendly Congestion Control for High-speed Networks”, PFLDnet 2006, Feb.2006.

[10] H.Shimonishi, T.Hama and T.Murase: “TCP-Adaptive Reno for Improving Efficiency-Friendliness Tradeoffs of TCP Congestion Control Algorithm”, PFLDnet 2006, Feb.2006.

[11] S.Liu, T.Başar and R.Srikant. “TCP-Illinois: A Loss and Delay-Based Congestion Control Algorithm for High-Speed Networks”, VALUETOOLS 2006, Oct.2006.

[12] A.Baiocchi, A.P.Castellani and F.Vacirca: “YeAH-TCP: Yet Another Highspeed TCP”, PFLDnet 2007, Feb.2007.

[13] K.Kaneko, T.Fujikawa, S.Zhou and J.Katto: “TCP-Fusion: A Hybrid Congestion Control Algorithm for High-speed Networks”, PFLDnet 2007, Feb.2007.

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

[15] 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.

[16] H. Shimonishi, M. Y. Sanadidi, and M. Gerla: “Improving Efficiency-Friendliness Tradeoffs of TCP in Wired-Wireless Combined Networks”, In proc. of ICC, 2005.

[17] R.Wang, M.Valla, M.Y.Sanadidi, and M.Gerla: “Adaptive Bandwidth Share Estimation in TCP Westwood”, IEEE Globecom 2002, Nov.2002.

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

[19] S.Floyd and V.Jacobson: “Random Early Detection Gateways for Congestion Avoidance”, UEEE/ACM Trans. on Networking, Vol.1, No.4, pp.397-413, Aug.1993.