On Hybrid TCP Congestion Control

Jiro Katto, Kazumine Ogura, Tomoki Fujikawa, Kazumi Kaneko and Zhou Su
Dept of Computer Science, Waseda University.
E-Mail: katto@waseda.jp

***Abstract* -** This paper presents several considerations on the hybrid TCP congestion control, in which loss-based schemes like TCP-Reno and delay-based schemes like TCP-Vegas are combined in a smart way. We firstly introduce simple analytical models of loss-based TCP, delay-based TCP and their hybrid. With support of experimental results, effectiveness of the hybrid TCP is proved. Then, we try to develop a more detailed model of TCP-Fusion, which we had proposed and is one of hybrid TCPs. We furthermore present delay-based TCP friendly rate control as a by-product of the analytical modeling, which has a potential to surpass the classical TFRC.

Keywords: Congestion Control, TCP-Reno, TCP-Vegas, TCP-Westwood, TFRC.

1 Introduction

TCP (Transmission Control Protocol) is commonly used in the current Internet, which provides end-to-end reliable data communication. It has been pointed out by many researchers that performance of TCP-Reno [1], which is widely utilized over the current Internet, deteriorates in high-speed networks and also in wireless networks. This is primary due to its congestion control mechanism, in which congestion window (cwnd) is increased by one Maximum Segment Size (MSS) every RTT (Round Trip Time) and halved upon packet losses regardless of network condition. To overcome this problem, a number of TCP variants have been proposed. They can be classified into three categories; loss-based (window-based), delay-based (rate-based) and their hybrid (mixed loss-delay-based).

Loss-based schemes modify the AIMD (Additive Increase Multiplicative Decrease) mechanism of TCP-Reno to quickly increase and slowly decrease the cwnd than TCP-Reno when packet losses happen. Examples are High-speed TCP, Scalable TCP, BIC-TCP, CUBIC-TCP, H-TCP and early versions of TCP-Westwood [2-4]. Delay-based schemes exploit RTT to predict network congestion before packet losses happen and shows excellent performance in efficiency and fairness. Examples are TCP-Vegas [5] and FAST-TCP [6]. However, it is well known that their performances are severely degraded when competing with loss-based TCP flows.

Hybrid methods which combine loss-based features and delay-based features have been proposed recently. Examples are TCP-Adaptive Reno (ARENO) [7], Compound TCP (CTCP) [8], YeAH-TCP, TCP-Illinois and our TCP-Fusion [9]. They adaptively switch their congestion control modes (loss-based and delay-based) according to congestion level measurement estimated from RTT. By doing this, they can provide high throughput efficiency of delay-based schemes when network is not congested, and also provide friendliness to loss-based schemes when network is congested.

Since there had been few performance analysis on the hybrid TCP schemes, we had proposed simple analytical models of the hybrid TCP and clarified its effectiveness with experimental results [10]. However, since we did not consider detailed behaviors of the hybrid TCP in the reference, in this paper, we propose more detailed performance models of TCP-Fusion. We furthermore present delay-based TCP friendly rate control as a by-product of the analytical modeling, which has a potential to surpass the classical TFRC (TCP Friendly Rate Control) [11-13].

2 Hybrid TCP Congestion Control

Concept of the hybrid TCP congestion control is quite simple. When network is underutilized, it operates in a delay mode and exploits residual capacity of the bandwidth. When network becomes congested, it moves to a loss mode and behaves as TCP-Reno until next packet losses happen. Mode switching is generally carried out according to observed RTT values. Some of hybrid TCP methods utilize TCP-Westwood's mechanisms of the bandwidth estimation (TCPW-BE) and the achievable rate estimation (TCPW-RE). Accordingly, congestion window increase and decrease are executed in a smart way than legacy TCP methods.

2.1 TCP-Fusion [9]

We introduce basic behaviors of TCP Fusion. Behaviors of other hybrid schemes are similar.

A. Window Decrease after Packet Loss

When packet losses happen, TCP-Fusion reduces cwnd by

$$cwnd_{new} = \max\left(\frac{RTT_{\min}}{RTT} cwnd_{last}, \frac{cwnd_{last}}{2} \right) \tag{1}$$

where $cwnd_{new}$ and $cwnd_{last}$ are congestion window sizes right after and before the packet losses, respectively, and RTT_{min} the minimum observed RTT. When the buffer capacity is smaller than the BDP (bandwidth-delay product), multiplying 1/2 causes vacant capacity in the bottleneck. Therefore, instead of halving cwnd, multiplying RTT_{min}/RTT has an effect just to clear the router buffer, which contributed to high throughput efficiency.

B. Smart Window Control

When TCP-Fusion operates in steady state, it applies TCP-Vegas like window control mechanism. It estimates the number of packets stored in the bottleneck queue (*diff*), which is given by

$$diff = cwnd \cdot \frac{RTT - RTT_{\min}}{RTT} \tag{2}$$

Similar to CTCP, TCP-Fusion has another congestion windows (*reno_cwnd*) which emulates TCP-Reno's behavior. Then, window control of TCP-Fusion is carried out by

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

where $cwnd_{new}$ and $cwnd_{last}$ are congestion window sizes after and before the update. Parameter α is a threshold to switch three phases (increase/decrease/steady) and W_{inc} is a parameter to increase cwnd rapidly when the link is underutilized. When *reno_cwnd* surpasses current cwnd, TCP-Fusion switches to loss-based mode and behaves as TCP-Reno until next packet losses happen.

C. Parameter Setting

We discuss parameter settings of α and W_{inc}. Firstly, we assume that the bottleneck router can store at least G packets, which corresponds to queuing delay $D_{\min}$ (= RTT-RTT_{min}) that is detectable by a sender (i.e. TCP timer granularity). This G is given by

$$G = \frac{B * D_{\min}}{PS} \tag{4}$$

where B is bandwidth of the bottleneck link (estimated by TCPW-BE) and PS is a packet size.

Second, we assume that there are coexisting N TCP flows and they fill the buffer. In order to avoid buffer overflow in the steady state, the router has to be able to store (N*α) packets that is equal to G packets. Therefore, α can be given by

$$\alpha = \frac{G}{N} = \frac{(B/N) * D_{\min}}{PS} \approx \frac{RE * D_{\min}}{PS}. \tag{5}$$

where RE is a fair share rate (estimated by TCPW-RE).

On the other hand, TCP-Fusion will possibly inject W_{inc} packets into the buffer to fill a pipe. To avoid buffer overflow, W_{inc} should be upper bounded by G,

$$W_{inc} \le G = \frac{B * D_{\min}}{PS}. \tag{6}$$

2.2 Analytical Models [10]

We introduce simple analytical models of loss-based TCP (Reno), delay-based TCP and their hybrid. Let w denote cwnd when packet losses happen. In the case of loss-based TCP, it is well known that packet loss probability p is given by

$$p = \frac{8}{3w^2} \tag{7}$$

[11,12]. It is suggested in [11] that this equation can hold in the random drop case in addition to the congestion drop case. Furthermore, let W represent cwnd size which just fills a pipe, which is equivalent to BDP.

A. Single Flow Case

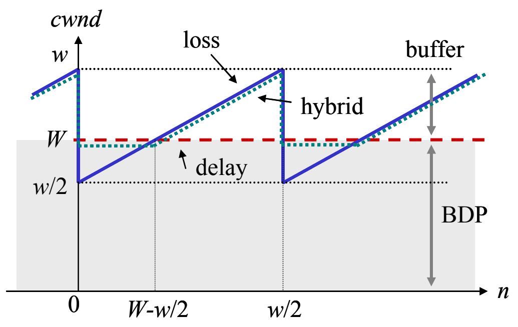
Figure 1(a) shows ideal behavior models of loss-based, delay-based and hybrid TCPs when there are no competing flows. This is the case that router buffer size is smaller than BDP or packet loss rate is medium (Case 2). We omit the case when the buffer size is larger than BDP (Case 1), or packet loss rate is high (Case 3). This classification is done according to relationship between w and W. Horizontal axis means the number of RTT rounds, and vertical axis does cwnd size.

We can recognize that loss-mode causes vacant capacity from 0 to *W*-*w*/2 after packet losses. Delay-mode always fills a pipe. Hybrid performs as delay-mode from 0 to *W*-*w*/2, and performs as loss-mode from *W*-*w*/2 to next packet loss (*w*/2).

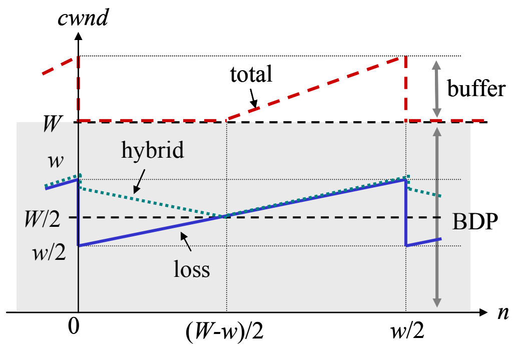
B. Two Flow Case

Figure 1(b) shows ideal behavior models when loss-based TCP and hybrid TCP are competing. This is also the case that router buffer size is smaller than BDP or packet loss rate is medium (Case 2). We omit Case 1 and Case 3 similar to previous subsection. We also omit delay-based TCP because our focus is on the hybrid.

We can recognize that loss-mode reduces its cwnd below *W*/2 after packet losses. This causes vacant capacity which hybrid TCP can exploit. At *n*=(*W*-*w*)/2, loss-based TCP reaches *W*/2 and packet buffering starts. Hybrid TCP performs as delay-mode (filling a pipe) from 0 to (*W*-*w*)/2, and performs as loss-mode from (*W*-*w*)/2 to next packet loss (*w*/2).



(a) Single flow case.



(b) Two flow case.

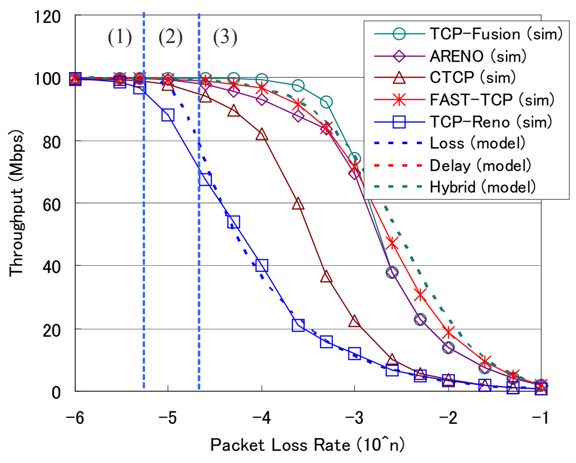
Figure 1: Ideal behavior models of TCPs.

C. Evaluations

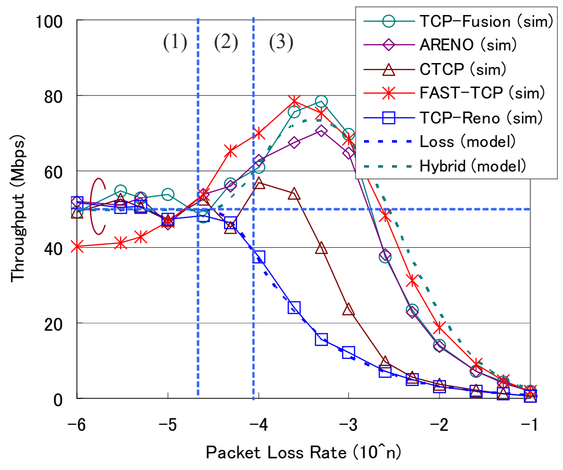
We can calculate the number of transmitted packets and elapsed time of single congestion avoidance round (from 0 to *w*/2) from Figure 1. We can also append timeout penalty as a function of packet loss probability *p*. We then calculate estimated throughputs of each TCP by

$$\frac{\textit{transmitted packets}}{\textit{elapsed time} + \textit{timeout penalty}} \tag{8}$$

Figure 2 shows evaluation results for single flow case and two flow case, in the case that bottleneck bandwidth is 100Mbps and end-to-end RTT_{min} is 40ms. (1)(2)(3) means Cases 1,2 and 3. We plot analytical values as well as those of simulation experiments by ns-2 [14]. We can recognize hybrid TCP performs well except CTCP. This is simply because CTCP halves cwnd after packet losses. Delay-based FAST-TCP also performs well but it loses friendliness when competing TCP-Reno is sufficiently buffered (Case 1 of Figure 2(b)). Among all, TCP-Fusion seems to perform the best.



(a) Single flow case.



(b) Two flow case.

Figure 2: Performance evaluations.

3 Model Improvement

Previous models didn't incorporate effects of parameters α and W_{inc} of TCP-Fusion. In this section, we develop more specific behavior models of TCP-Fusion.

A. Models for TCP-Fusion

Figure 3(a) shows an enlarged view of Case 2 of single flow model. We plot cwnd behaviors of loss-based, hybrid and TCP-Fusion. TCP-Fusion's action is summarized as follows. After a packet loss happens, single RTT round is consumed for lost packet retransmission. Then, congestion avoidance is restarted by setting cwnd=W which is equivalent to BDP. TCP-Fusion increases cwnd by W_{inc} per RTT round until RTT increase is observed. When applying Eqs (5) and (6), W_{inc} is equal to α in a single flow case and steady condition is immediately satisfied (i.e. $a=\alpha/W_{inc}=1$). At $n=(W-w/2+\alpha)$, loss-based window reaches delay-based window, and loss-based mode is triggered until next packet loss. Though TCP-Fusion injects extra packets more than ideal hybrid, this injection only causes RTT increase and throughput performance is nearly the same.

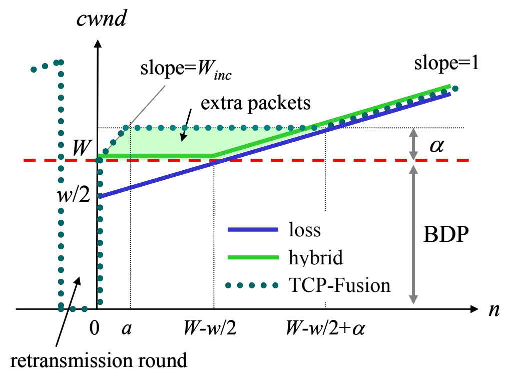
Figure 3(b) gives an enlarged view of case 2 of the two flow model, in which loss-based TCP and TCP-Fusion compete. Behavior details are as follows. After a packet loss happens, congestion avoidance phase is restarted by setting cwnd = $w/2$ for the loss-based TCP and cwnd = $W/2$ (half of BDP) for the TCP-fusion. TCP-Fusion increases the window by W_{inc} per RTT round until α packets are stored in the router buffer until $n=c$. TCP-Fusion gradually decreases cwnd to avoid RTT increase in response to cwnd increase of the loss-based TCP. At $n=d$, loss-based window reaches delay-based window and loss-based mode is triggered. Parameters b, c, d are given by

$$b=\frac{W-w}{2(1+W_{inc})},\ c=\frac{W-w+2\alpha}{2(1+W_{inc})},\ d=\frac{W-w+\alpha}{2} \qquad (9)$$

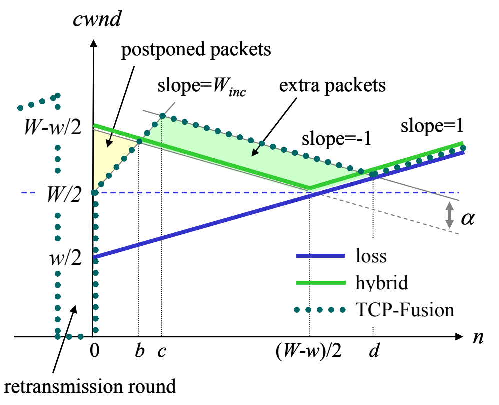
In Figure 3(b), extra packets and postponed (not sending) packets are plotted. They force four new timing parameters a, b, c, d into the model (a is for case 1), which split three cases of Figure 2 into seven cases although their details are omitted.

B. Evaluations

Figure 4 shows effects of parameters α and W_{inc} on throughputs for the two flow model. In Figure 4(a), we change α by multiplying 2^k (k=-3 to 3) to α. We plot both of analytical results and simulation results. This figure proves that parameter α should be kept small, otherwise, friendliness to loss-based TCP will be lost (loss-based TCP will be expelled). In Figure 4(b), we change W_{inc} by multiplying 2^k (k=-3 to 3) to W_{inc}. This result suggests that larger W_{inc}, does not affect the performance but rather smaller W_{inc} causes degradation. This is because too slow cwnd increase in delay-mode loses the chance to exploit residual capacity.

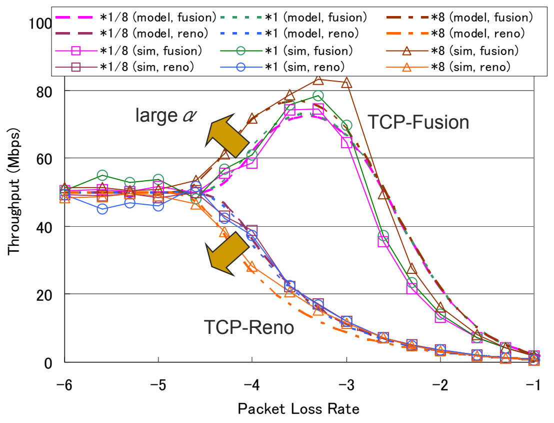


(a) Enlarged view of single flow model.

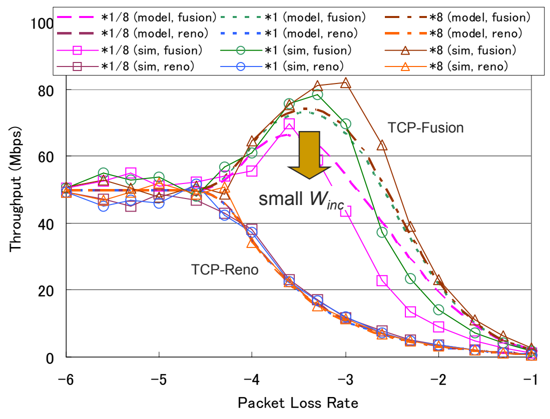


(b) Enlarged view of two flow model.

Figure 3: Microscopic behavior models of TCP-Fusion.



(a) Effect of parameter α.



(b) Effect of parameter W_{inc}.

Figure 4: Performance evaluations.

4 Delay-based TFRC

As a by-product of the analysis, delay-based TCP friendly rate control is derived.

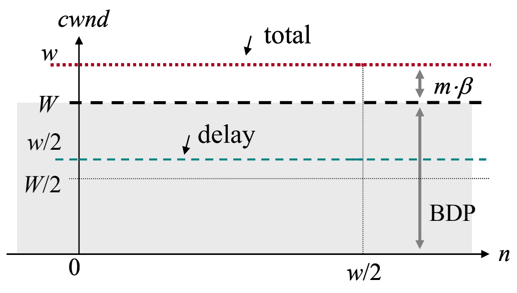


Figure 5: Two flow model of delay-based TCP.

Figure 5 shows two flow model in which two delay-based TCP flows are competing. For generalization, let M denote the number of TCP flows and β represent the number of packets which can be transferred within minimum timer granularity D_{min}. Then, we assume that each flow tries to keep $m/M\cdot\beta$ packets in a buffer, where m is a parameter related to scalability (M=2 and m=1 correspond to Eq.(5)). For one congestion avoidance round of loss-based TCP, estimated throughput of each flow is given by

$$R_{delay}=\frac{\frac{w}{2}\left(\frac{W}{M}+\frac{m}{M}\cdot\beta\right)\cdot PS}{\frac{1}{2}w\left(RTT_{\min}+m\cdot D_{\min}\right)+t_{RTO,delay}}. \tag{10}$$

Timeout penalty is given by

$$t_{RTO,delay}=\frac{K_{delay}}{K_{loss}}\cdot t_{RTO,loss} \tag{11}$$

where

$$\frac{K_{delay}}{K_{loss}}=\frac{W\cdot w/2}{3/8\cdot w^2}=\frac{4W}{3w}. \tag{12}$$

Furthermore, applying various equations such as Eq.(7), $\beta=\frac{B*D_{\min}}{PS}$ and $B=\frac{W*PS}{D_{\min}}$, Eq.(10) is changed to

$$R_{delay}=\frac{\frac{B}{M}\left(1+\frac{m}{M}\cdot\frac{D_{\min}}{RTT_{\min}}\right)}{\left(1+m\cdot\frac{D_{\min}}{RTT_{\min}}\right)+p\cdot\frac{B}{PS}\cdot t_{RTO,loss}}. \tag{13}$$

In case of the ideal delay-based TCP, in which no packet buffering is carried out but the pipe is filled up, terms related to D_{min} can be ignored and a compact form is given by

$$R_{delay}=\frac{B/M}{1+p\cdot B/PS\cdot t_{RTO,loss}}. \tag{14}$$

This equation is composed of bandwidth, the number of flows, packet loss probability and timeout penalty. Bandwidth and the number of flows can be estimated by TCPW-BE and TCPW-RE as before.

Figure 6 presents estimated throughputs of delay-based TFRC with references to those of TCP-Reno and TFRC, the latter of which is given by

$$R_{loss}=\frac{PS}{RTT\sqrt{\frac{2bp}{3}}+t_{RTO,loss}\cdot 3\sqrt{\frac{3bp}{8}}\cdot p(1+32p^2)} \tag{15}$$

where b is the number of packets acknowledged by TCP-ACK [13]. Network parameters are the same as of Figures 2 and 4. This figure suggests that delay-based TFRC would perform better than TCP-Reno and legacy TFRC by utilizing residual capacity when packet loss rate is high.

Figure 6 also shows a comparison result when different values are assigned to parameter m for the M=2 case. When m=1, each flow gently fills a buffer in a scalable manner similar to Eq.(5). However, when m=2, each flow persists in their own control parameter to store constant packets in a buffer. The result suggests that the scalable manner (m=1) will achieve fair and higher rate when packet loss rate is low. This result is similar to the case of Figure 4 and supports our scalable strategy in Eq.(5).

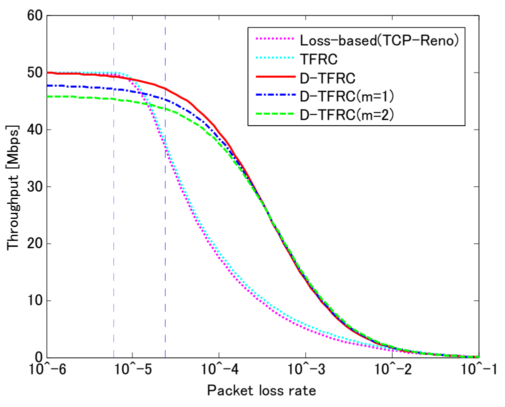


Figure 6: Estimated throughputs of delay-based TFRC when the number of flows is 2 (*M*=2).

5 Conclusions

This paper introduced hybrid TCP, especially our TCP-Fusion, and its performance analysis. Based on the ideal case analysis, we develop more specific models for TCP-Fusion. We also derive delay-based TFRC and suggests its superiority over legacy approach.

References

[1] M.Allman et al: "TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms," RFC 2581, Apr.1999.

[2] C.Casetti et al: "TCP Westwood: Bandwidth Estimation for Enhanced Transport over Wireless Links", ACM Mobicom 2001, Jul.2001.

[3] R.Wang et al: "Efficiency/Friendliness Tradeoffs in TCP Westwood", IEEE SCC 2002. Jul.2002.

[4] R.Wang et al: "Adaptive Bandwidth Share Estimation in TCP Westwood", IEEE Globecom 2002, Nov.2002.

[5] L.S.Brakmo and L.L.Peterson: “TCP Vegas: End-to-End Congestion Avoidance on a Global Internet,” IEEE Journal on Selected Areas in Communications, Vol. 13, No.8, 1995.

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

[7] H.Shimonishi et al: “TCP-Adaptive Reno for Improving Efficiency-Friendliness Tradeoffs of TCP Congestion Control Algorithm”, PFLDnet 2006, Feb.2006.

[8] K.Tan et al: “Compound TCP: A Scalable and TCP-Friendly Congestion Control for High-speed Networks”, PFLDnet 2006, Feb.2006.

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

[10] J.Katto, K,Ogura, Y.Akae, T.Fujikawa, K.Kaneko and Z.Su: “Simple Model Analysis and Performance Tuning of Hybrid TCP Congestion Control”, IEEE Globecom 2008, Dec.2008 (to be published).

[11] M.Mathis et al: "The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm", ACM SIGCOMM CCR, Vol.27, No.3, Jul.1997.

[12] J.Padhye et al: "Modeling TCP Throughput: A Simple Model and its Empirical Validation", ACM SIGCOMM 1998, Oct.1998.

[13] M. Handley et al.: "TCP Friendly Rate Control (TFRC): Protocol Specification", RFC 3448, Jan.2003.

[14] “ns-2 network simulator”, http://www.mash.cs.berkley.edu/ns.