Is tampered-TCP really effective for getting higher throughput in the Internet?

Junichi Maruyama Go Hasegawa Masayuki Murata
Graduate School of Information Science and Technology, Osaka University
1-5 Yamadaoka Suita, Osaka 565-0871, Japan
{j-maruyama, hasegawa, murata}@ist.osaka-u.ac.jp

Abstract

This paper examines the effectiveness of tampered-TCP whose congestion control mechanism has been modified by malicious users for higher than the normal TCP throughput. In this paper, We focus on a tampered-TCP in which the increase and decrease ratio of the congestion window size were changed during the congestion avoidance phase. The performance of the tampered-TCP was examined based on a mathematical analysis and simulation given the coexisting of tampered-TCP and TCP Reno connections in the network. The following characteristics of the tampered-TCP were found: (1) when the increase ratio is larger than 2 packets per RTT, the throughput is significantly degraded due to many timeouts; (2) lowering the decrease ratio is effective for the throughput improvement; and (3) the effectiveness of lowering the decrease ratio is less than the ill-effects encountered when the increase ratio is augmented. We conclude that the tampered-TCP would self-destruct in many parameter regions, and there is few situation where the tampered-TCP obtains much higher throughput.

1. Introduction

Currently, most Internet traffic is carried via the Transmission Control Protocol (TCP) [1]. The congestion control mechanism of TCP allows the Internet to provide fair and unstoppable services without any collapse due to an extreme increase in traffic. The congestion control mechanism of TCP is defined by RFC [2], and its implementation in operating systems is based on this document. Therefore, if two users, with different operating systems, should share a bottleneck link in the network, each user can obtain a fair throughput despite the minor implementation differences of the protocol in the two operating systems. However, since TCP works at the end hosts, it is easy for users to modify its behavior, especially for those with open source operating systems. Thus, it is likely that there exists many kind of TCP variants, created by malicious users for higher than normal throughput [3–5]. In this paper, we will refer to such modified TCPs as *tampered-TCPs*.

Generally, when modifications to TCP congestion control mechanisms are proposed, the effects of those modifications are compared with the original TCP Reno and are evaluated based on the performance where both the original and modified TCPs co-exist on the same network [6–10]. However, malicious users can selfishly modify TCP behavior, focusing only on increasing their own throughput. When the population of tampered-TCPs increases in a network, these tampered-TCP connections may unfairly monopolize network bandwidth, causing the normal TCPs to suffer from low throughput.

On the other hand, such tampered-TCPs may not work well in the actual Internet environment. For example, by augmenting the increase ratio of the congestion window size, the number of packets that are simultaneously injected into the network increases rapidly. This results in increased packet loss due to congestion within the network, which leads to degraded throughput. Thus, a tampered-TCP *self-destructs*, when its behavior causes it to send data packets more aggressively than normal TCP Reno connections.

In this paper, the effects of a tampered-TCP on a network shared with normal TCP Reno connections were sought to determine if the tampered-TCP exhibits self-destructive behavior under various situations. In addition, proofs showing that the low cost modification of TCPs does not work are presented by considering a tampered-TCP, which changes the increase and decrease ratio of the congestion window size during the congestion avoidance phase, and we call such TCP variant just tampered-TCP. There are two reasons for choosing such a tampered-TCP. Firstly, for malicious users, it is comparatively easy to modify the increase and decrease ratio of the congestion window size in the TCP source code. Secondly, it is possible for researchers to investigate the behavior of the modified TCP by mathematical analysis [11, 12].

In this paper, we employed mathematical analysis and simulation experiments to evaluate tampered-TCP characteristics. For the mathematical analysis, the analysis presented in [11] was used to derive the average throughput of the tampered-TCP and TCP Reno connections when they share a bottleneck link, and, hence, explain how the parameters of the tampered-TCP affect its performance and fairness compared to normal TCP Reno connections. The accuracy of the mathematical analysis was confirmed using simulation experiments. Based on the results of the simulation experiments, the following characteristics of the tampered-TCP were identified: (1) when the increase ratio is larger than two packets per Round Trip Time (RTT), the throughput degraded significantly due to many timeouts, (2) lowering the decrease ratio is effective for throughput improvement, (3) the effectiveness of lowering the decrease ratio is less than ill-effects encountered when the increase ratio is augmented. The discussion section shows that little region exists where the tampered-TCP can improve the throughput.

2. Mathematical analysis

2.1 Network model and evaluation metric

Figure 1 depicts the network model that was used for the mathematical analysis and simulation experiments. The network model consists of sender and receiver hosts using TCP Reno connections, sender and receiver hosts using

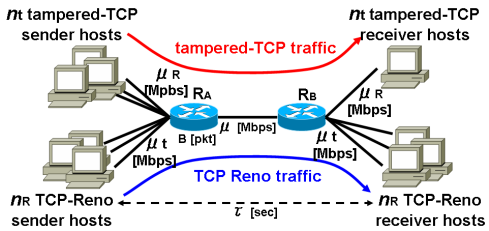


Figure 1. Network model

tampered-TCP connections, two routers ($R_{\rm A}$ and $R_{\rm B}$) with a droptail buffer, and links interconnecting the hosts and routers. The bandwidth of the link between the router $R_{\rm A}$ and the router $R_{\rm B}$ is μ Mbps, the buffer size at the router $R_{\rm A}$ is B packets, the propagation delay between the sender and receiver hosts is τ sec, the bandwidth of the links between the tampered-TCP hosts and routers is $\mu_{\rm t}$ Mbps, and that between the TCP Reno hosts and the routers is $\mu_{\rm R}$ Mbps. There are $n_{\rm t}$ tampered-TCP connections and $n_{\rm R}$ TCP Reno connections. We assume that the sender hosts have an infinite amount of data to send and continue transmitting as much data as is allowed by their congestion window sizes.

To evaluate the effectiveness of the tampered-TCP, the throughput ratio was introduced as an evaluation metric. It is defined as:

$$Throughput\ ratio = \frac{(Throughput\ of\ \text{tampered-TCP})}{(Throughput\ of\ \text{TCP Reno})} \quad (1)$$

When this value is greater than 1, the tampered-TCP is said to work effectively.

2.2 Behavior of TCP Reno and tampered-TCP

When triggered by a packet loss event, TCP Reno will change its congestion window size [11, 12]. Figure 2 shows a typical case in a network where both TCP Reno and tampered-TCP connections co-exist. Here, we define the interval from the (i-1)-th packet loss event to the i-th packet loss event as the i-th *cycle*. We further divide the i-th cycle into RTTs and consider the congestion window size for each RTT. The congestion window size of TCP Reno at the j-th RTT of the i-th cycle is defined as $W_{\rm R}(i,j)$.

The congestion control mechanism of the TCP Reno consists of two phases: the slow start phase and the congestion avoidance phase. For each phase, TCP Reno uses a different algorithm for increasing the congestion window size. In the slow start phase, TCP Reno increases its window size by one packet on receiving an ACK packet. On the other hand, in the congestion avoidance phase, TCP Reno increases its window size $W_{\rm R}(i,j)$ by $1/W_{\rm R}(i,j)$ packets when it receives an ACK packet. Focusing on the change of the congestion window size in every RTT, $W_{\rm R}(i,j)$ can be derived as follows:

$$W_{\rm R}(i,j)=\begin{cases} 2W_{\rm R}(i,j-1), & \text{if } W_{\rm R}(i,j-1) < S_{\rm R}(i) \\ W_{\rm R}(i,j-1)+1, & \text{if } W_{\rm R}(i,j-1) \geq S_{\rm R}(i) \end{cases} \quad (2)$$

where $S_{\rm R}(i)$ is an ssthresh value in the i-th cycle at which TCP Reno changes its phase from the slow start phase to the congestion avoidance phase.

When packet losses occur in the network, TCP Reno detects them either by a retransmission timeout or by receiving triple duplicate ACK packets (three or more ACK packets with the same sequence number) and retransmitting them. If the packet losses are detected by the retransmission timeout, TCP Reno sets its congestion window size to 1 packet and change its phase to the slow start phase. On the other hand, if packet losses are detected by the duplicate ACK packets route, then TCP Reno sets its congestion window size to half of that just before the packet loss. In both cases, TCP Reno sets $S_{\rm R}(i)$ to half of the congestion window size just before the detection of the packet losses.

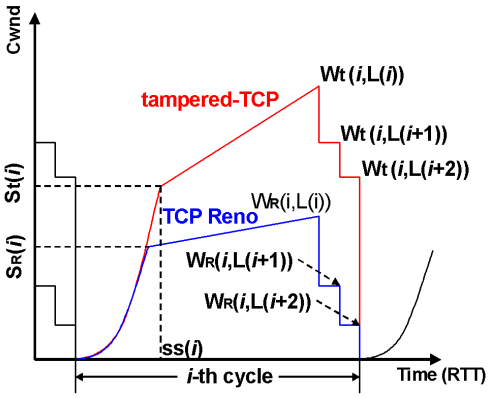


Figure 2. The change in the congestion window size during the i-th cycle

The behavior of the tampered-TCP is almost identical to that of TCP Reno. However, in the congestion avoidance phase, the increase speed of the congestion window size is different than that in a TCP Reno. The congestion window size of a tampered-TCP at the j-th RTT of the i-th cycle was defined as $W_{\rm t}(i,j)$. Then, when a tampered-TCP receives an ACK packet, it increases its congestion window size by $\alpha \cdot 1/W_{\rm t}(i,j)$, that is α times faster than TCP Reno. This behavior can be described as follows:

$$W_{\rm t}(i,j)=\begin{cases} 2W_{\rm t}(i,j-1), & \text{if } W_{\rm t}(i,j-1) < S_{\rm t}(i) \\ W_{\rm t}(i,j-1)+\alpha, & \text{if } W_{\rm t}(i,j-1) \geq S_{\rm t}(i) \end{cases} \quad (3)$$

where $S_{\rm t}(i)$ is an ssthresh value of the tampered-TCP in the i-th cycle. When packet losses occur in the network, the tampered-TCP detects and retransmits them in the same way as a TCP Reno. However, when packet losses are detected by duplicate ACK packets, the tampered-TCP sets its congestion window size to β ($0.5 \leq \beta \leq 1$) times of that just before the packet loss. In both cases, the tampered-TCP sets $S_{\rm t}(i)$ to β times of the congestion window size just before the detection of the packet losses.

2.3 Analysis

In the analysis, the cyclic changes in the congestion window size for tampered-TCP and TCP Reno connections were modeled as being triggered by packet loss events (Figure 2). Thus, the average throughput values can be calculated. It was assumed that $n_{\rm R}$ TCP Reno connections behave identically, and that $n_{\rm t}$ tampered-TCP connections also behave identically. Note that this assumption is reasonable when a droptail buffer is used at the bottleneck link.

The congestion window sizes at the beginning of the i-th cycle, corresponding to $W_{\rm R}(i,1)$ and $W_{\rm t}(i,1)$, are equal

to those at the end of the (i-1)-th cycle. The congestion window sizes of both connections grow according to Equations (2) and (3). When the sum of the congestion window sizes becomes larger than the bandwidth-delay product of the network ($2\tau\mu$ here), then the excess packets begin to accumulate at the router buffer. Finally, packet losses occur when the buffer is fully utilized. Assuming that the packet losses occur at the $L(i)$-th RTT of the i-th cycle, the following equations are satisfied:

$$n_{\rm R}W_{\rm R}(i,L(i)-1)+n_{\rm t}W_{\rm t}(i,L(i)-1)\leq 2\tau\mu+B$$

$$n_{\rm R}W_{\rm R}(i,L(i))+n_{\rm t}W_{\rm t}(i,L(i))>2\tau\mu+B \qquad (4)$$

Then, $L(i)$ is given by:

$$L(i)=\frac{(2\tau\mu+B)-n_{\rm R}W_{\rm R}(i,1)+n_{\rm t}W_{\rm t}(i,1)}{n_{\rm t}\alpha+n_{\rm R}} \qquad (5)$$

Since $D(i)$ denotes the number of dropped packets due to buffer overflow at the end of the i-th cycle, then $D(i)$ is given by:

$$D(i)=n_{\rm R}W_{\rm R}(i,L(i))+n_{\rm t}W_{\rm t}(i,L(i))-(2\tau\mu+B)$$

Furthermore, the number of dropped packets in each TCP Reno connection is denoted by $D_{\rm R}(i)$ and in each tampered-TCP connection by $D_{\rm t}(i)$. By assuming that the ratio of $D_{\rm R}(i)$ and $D_{\rm t}(i)$ is equal to the ratio of their congestion window sizes at the packet loss events, the following equations can be derived:

$$D_{\rm R}(i)=\frac{W_{\rm R}(i,L(i))}{n_{\rm R}W_{\rm R}(i,L(i))+n_{\rm t}W_{\rm t}(i,L(i))}D(i)$$

$$D_{\rm t}(i)=\frac{W_{\rm t}(i,L(i))}{n_{\rm R}W_{\rm R}(i,L(i))+n_{\rm t}W_{\rm t}(i,L(i))}D(i)$$

Next, the congestion window size of each connection just after the packet losses was derived. In this analysis, since it can be assumed that droptail routers are used, the packets are dropped due to buffer overflow at the router $R_{\rm A}$ in a bursty fashion. Thus, we assume D(i) >1. When three or more packets are dropped within a TCP connection window, the first two packets are transmitted by the fast retransmit algorithm, followed by a timeout and then the retransmission of the remaining packets [13]. Since the tampered-TCP behaves the same as TCP Reno during a packet loss event, the congestion window sizes of TCP Reno and tampered-TCP connections after the first packet retransmission are determined by:

$$W_{\rm R}(i,L(i)+1)=\tfrac{W_{\rm R}(i,L(i))}{2},W_{\rm t}(i,L(i)+1)=\beta W_{\rm t}(i,L(i))$$

Similarly, the congestion window sizes after the second retransmission are determined by:

$$W_{\rm R}(i,L(i)+2)=\tfrac{W_{\rm R}(i,L(i)+1)}{2},W_{\rm t}(i,L(i)+2)=\beta W_{\rm t}(i,L(i)+1)$$

After the second retransmission, the retransmission timeout occurs, each connection sets its congestion window size to 1, and the values of ssthresh are updated as follows:

$$S_{\rm R}(i+1)=\tfrac{W_{\rm R}(i,L(i)+2)}{2},S_{\rm t}(i+1)=\beta W_{\rm t}(i,L(i)+2) \qquad (6)$$

Let us now consider the congestion window size of the tampered-TCP during a packet loss event. From empirical results, the average number of dropped packets in a tampered-TCP connection at the end of the i-th cycle is approximated as $D_{\rm t}(i)=\alpha$. This equation means that for a tampered-TCP connection, timeout never occurs when $\alpha<3$, while timeout occurs whenever $\alpha\geq 3$. Thus, we derive the evolution of the congestion window size of a tampered-TCP in the i-th cycle for the two cases where the $(i-1)$-th cycle ends with and without a retransmission timeout.

In case of no timeout ($\alpha<3$) In this case, the i-th cycle begins with a congestion avoidance phase. At the (i-1)-th cycle, since the number of dropped packets in the tampered-TCP connection is $D_{\rm t}(i-1)=\alpha$, the tampered-TCP retransmits α packets. Thus, the congestion window size at the beginning of the i-th cycle, $W_{\rm t}(i,j)$, is given by:

$$W_{\rm t}(i,1)=\beta^{\alpha}W_{\rm t}(i-1,L(i-1))$$

From Equation (3), $W_{\rm t}(i,j)$ is derived as follows:

$$W_{\rm t}(i,j)=\beta^{\alpha}W_{\rm t}(i-1,L(i-1))+\alpha j \qquad (7)$$

In case of timeout ($\alpha\geq 3$) In this case, since timeout occurred at the end of the (i-1)-th cycle, the i-th cycle begins with a slow start phase. Since the number of the dropped packets in a tampered-TCP connection at the end of the (i-1)-th cycle is $D_{\rm t}(i-1)=\alpha$, the tampered-TCP retransmits the first 2 packets by detecting duplicate ACKs. Thus, the ssthresh value is given by Equation (6), and the congestion window size at the beginning of the i-th cycle is 1.

By assuming that the slow start phase of the i-th cycle ends at the $ss(i)$-th RTT, the congestion window size at the j-th RTT of the i-th cycle, $W_{\rm t}(i,j)$, is derived as follows:

$$W_{\rm t}(i,j)=\begin{cases}2^{j}, & \text{if } j<ss(i)\\ 2^{ss(i)}+\alpha(j-ss(i)), & \text{if } j\geq ss(i)\end{cases} \qquad (8)$$

Here, $ss(i)$ can be calculated from Equation (3) as follows:

$$ss(i)=\lfloor\log_2(\beta^2W_{\rm t}(i-1,L(i-1)))\rfloor \qquad (9)$$

From Equations (7)-(9), $W_{\rm t}(i,j)$ can be determined as follows:

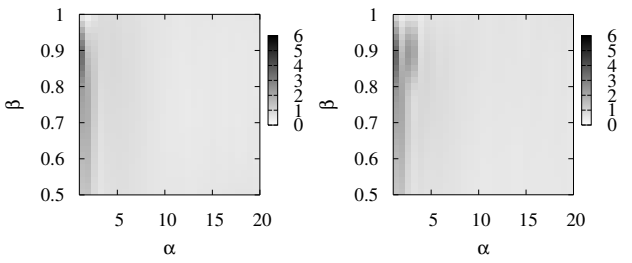
$$W_{\rm t}(i,j)=\begin{cases}\beta^{\alpha}W_{\rm t}(i-1,L(i-1))+\alpha j, & \text{if } \alpha<3\\ \beta^2W_{\rm t}(i-1,L(i-1))+\alpha[j-ss(i)], & \text{if } \alpha\geq 3\end{cases} \qquad (10)$$

Finally, the average throughput for TCP Reno and tampered-TCP connections based on the congestion window size evolutions was derived. In order to determine this, the queuing delay at the bottleneck link buffer is needed to obtain the precise value of RTTs in the TCP connections. Since the number of stored packets in the buffer at the j-th RTT of the i-th cycle is given by $max((n_{\rm R}W_{\rm R}(i,j)+n_{\rm t}W_{\rm t}(i,j)-2\tau\mu),0)$, the queuing delay, $Q(i,j)$, is derived as follows:

$$Q(i,j)=\frac{max((n_{\rm R}W_{\rm R}(i,j)+n_{\rm t}W_{\rm t}(i,j)-2\tau\mu),0)}{\mu}$$

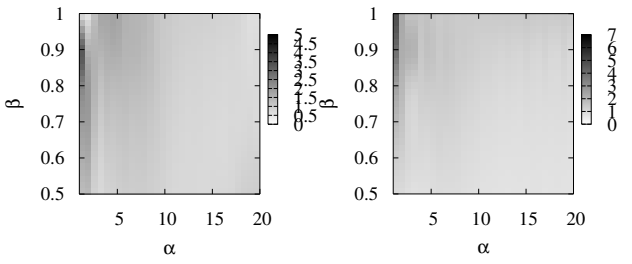
Therefore, the average throughput of TCP Reno $\rho_{\rm R}$ and tampered-TCP $\rho_{\rm t}$ is:

$$\rho_{\rm R}=\frac{\sum_{i=1}^{\infty}\sum_{j=1}^{L(i)}W_{\rm R}(i,j)}{\sum_{i=1}^{\infty}\sum_{j=1}^{L(i)}(Q(i,j)+2\tau)},\ \rho_{\rm t}=\frac{\sum_{i=1}^{\infty}\sum_{j=1}^{L(i)}W_{\rm t}(i,j)}{\sum_{i=1}^{\infty}\sum_{j=1}^{L(i)}(Q(i,j)+2\tau)}$$



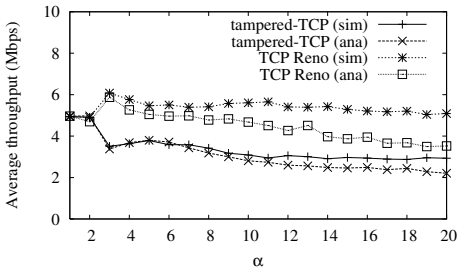
(a) analysis (b) simulation

Figure 3. Analysis and simulation results for throughput ratio ($n_{\mathrm{R}} = 1$, $n_{\mathrm{t}} = 1$, $\mu = 10$Mbps)

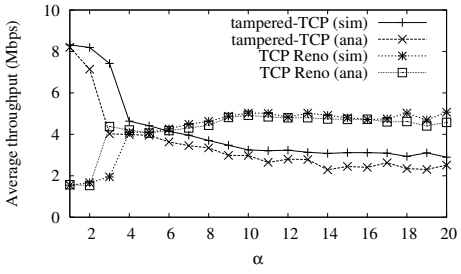


(a) analysis (b) simulation

Figure 4. Analysis and simulation results for throughput ratio ($n_{\mathrm{R}} = 10$,$n_{\mathrm{t}} = 1$, $\mu = 50$Mbps)



(a) $\beta = 0.5$



(b) $\beta = 0.9$

Figure 5. Throughput change of TCP Reno and tampered-TCP with various α and β

3 Simulation experiments and discussions

In this section, first, simulation experiments are presented that confirm the accuracy of the mathematical analysis developed in the previous section and, then, the characteristics of the tampered-TCP based on the mathematical analysis and the simulation results are discussed. In the simulation experiments, the network model shown in Figure 1, where $\mu_{\mathrm{R}} = \mu_{\mathrm{t}} = 100$ Mbps, the propagation delays of the links between the sender/receiver hosts and routers are all set to 5 msec, the propagation delay of the link between the routers is 10 msec, the packet size is 1500 Bytes, and the buffer size of the bottleneck link is twice the bandwidth-delay product between the sender and receiver hosts. α, the increase ratio of the congestion window size of tampered-TCP, was varied between 1 and 20, while β, the decrease ratio of the congestion window size of the tampered-TCP, was varied between 0.5 and 1. The simulation time was 60 seconds. We used a ns-2 [14] for the simulation experiments.

3.1 Confirmation of analysis results

Figure 3 shows the change in the throughput ratio, defined by Equation (1), as a function of α and β, where $n_{\mathrm{R}} = n_{\mathrm{t}} = 1$ and $\mu = 10$ Mbps. Both the analytical and simulation results were plotted. This figure confirms that the mathematical analysis presented in the previous section gives a precise throughput ratio estimation. As well, it can be seen that the tampered-TCP fails to obtain a large throughput compared with the normal TCP Reno for almost all of the parameter region (α, β), except for the case when α is smaller than 3 and β is around 0.9.

Figure 4 shows the results when n_{R} is increased to 10 and μ is increased to 50 Mbps. This setting is more realistic since it assumes that there are many normal TCP Reno connections and relatively few tampered-TCP connections. Once more, the analytical results are almost the same as the simulation results. As before, the tampered-TCP does not work for most of the (α, β) parameter region. The next subsection based on the analytical and simulation results explains the behavior of the tampered-TCP in more detail and reveals why the tampered-TCP is so ineffective.

3.2 Characteristics of tampered-TCP

This section helps to explain the ineffectiveness of the tampered-TCP by presenting the relevant analytical results, which were confirmed by simulations. Furthermore, the assumptions of the mathematical analysis were validated.

3.2.1 Sensitivity to α and β

In this subsection, a single TCP Reno connection co-exists with a single tampered-TCP connection ($n_{\mathrm{R}} = n_{\mathrm{t}} = 1$), and the bottleneck link bandwidth μ is set to 10 Mbps.

Figure 5(a) plots the change of the average throughput of the TCP Reno and the tampered-TCP connection for both the analytic and simulation cases as a function of the value of α when β is set to 0.5. We show both of simulation and analysis results in this figure. A sharp decrease in throughput occurs for the tampered-TCP when α is larger than 2. Furthermore, further increases in α cause the throughput of the tampered-TCP connection to gradually degrade, due to an increase in the number of dropped packets as α increases.

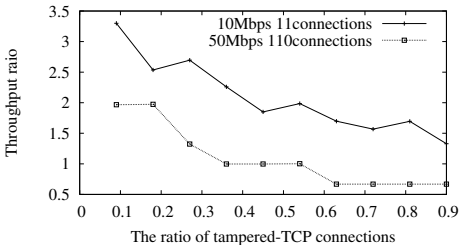


Figure 6. Throughput ratio as a function of the ratio of tampered-TCP connections

Figure 5(b) shows the results when β is increased to 0.9, which means that the tampered-TCP decreases the window size by only 10% when a packet loss occurs. This figure shows that for α smaller than 3, the tampered-TCP achieves larger throughput than the TCP Reno connection. However, as α increases above 3, a situation with a cause similar to that seen when in $\beta = 0.5$ occurs. Thus, the results suggest that increasing β is effective in increasing the performance of tampered-TCP. However, any α greater than 3 cancels the effectiveness that may have been gained from an increase in β.

By comparing Figures 3 and 4, the parameter region where the tampered-TCP is effective does not become so larger when the link bandwidth becomes larger. This suggest that TCP variants for high-speed and long-distance network such as HSTCP [15] may not work well in such networks. Furthermore, with parameter sets in the effective region, it is obvious that original TCP Reno flows suffer from low throughput when co-existing such high-speed TCP variants. Therefore, we need to consider fairness property of such TCP variants and original TCP Reno when we deploy high-speed TCP variants in the actual networks.

3.2.2 Effect on the throughput ratio for tampered-TCP connections

It has been shown that a tampered-TCP is not effective in most of the parameter region (α, β). However, the results from the previous subsections suggest that when α is around 2 and β is increased to about 0.9, higher throughput is obtained by the tampered-TCP. This subsection considers the situation where such *well-configured* tampered-TCPs proliferate in a network, and thus diminish its effectiveness.

Figure 6 shows the change in the throughput ratio when there is an increase in the ratio of the number of tampered-TCP connections to the total number of TCP connections in the network. The results are plotted for the following 2 cases: (1) The total number of connections is 11 and $\mu = 10$ Mbps, and (2) The total number of connections is 110 and $\mu = 50$ Mbps. $\alpha = 2$ and $\beta = 0.9$ were used for tampered-TCP connections, which are the best values determined in the previous subsections. This figure shows that as the number of tampered-TCP connections increases, the effectiveness sharply diminishes. In the case of 110 connections, the throughput ratio decrease below 1.0, which implies that using a tampered-TCP leads to self-destruction in its performance.

4 Conclusion and future works

In this paper, the performance of the tampered-TCP, which change the increase and decrease ratio of the congestion window size, was evaluated. It was shown that when the increase ratio is larger than 2 packets per RTT, TCP retransmission timeouts occur frequently, and the throughput diminishes sharply. It was also established that lowering the decrease ratio is beneficial whenever the increase ratio is smaller than 3. However, it was demonstrated that the effectiveness rapidly decreases if there are too many tampered-TCP connections, which even use the well-configured parameters.

One possible way to increase the throughput of the tampered-TCP is to enable the TCP's SACK option [16]. Thus, we are now evaluating the effectiveness of the SACK option. Preliminary results show that it can effectively increase the performance of the tampered-TCP. Therefore, network mechanisms, and not endhost mechanisms, need to be introduced in order to maintain the fairness property currently found on the Internet. The search for this mechanism will form the basis of our future work. We also plan to investigate the performance of tampered-TCP in the actual Internet environment.

References

[1] M. Fomenkov, K. Keys, D. Moore, and K. Claffy, "Longitudinal study of Internet traffic from 1998-2003," in *Proceedings of Winter International Symposium on Information and Communication Technologies (WISICT 2004)*, Jan. 2004.

[2] M. Allman, V. Paxson, and W. Stevens, "TCP congestion control," *RFC2581*, Apr. 1999.

[3] S. Savage, N. Cardwell, D. Wetherall, and T. Anderson, "TCP congestion control with a misbehaving receiver," *ACM Computer Communications Review, 29(5):71-78*, Oct. 1999.

[4] S. Floyd and K. Fall, "Router mechanisms to support end-to-end congestion control," tech. rep., Network Research Group at LBNL, Feb. 1997.

[5] M. Baldi, Y. Ofek, and M. Yung, "Idiosyncratic signatures for authenticated execution of management code," in *Proceedings of 14th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management (DSOM 2003), Heidelberg, Germany*, Oct. 2003.

[6] Y. R. Yang and S. S. Lam, "General AIMD congestion control," in *Proceedings of the IEEE International Conference on Network Protocols*, Nov. 2000.

[7] L. Mamatas and V. Tsaoussidis, "Protocol behavior : More effort, more gains?," in *Proceedings of Personal, Indoor and Mobile Radio Communications, 2004. PIMRC 2004. 15th IEEE International Symposium, 125- 129 Vol.1*, Sept. 2004.

[8] L.S.Brakmo, S.W.O'Malley, and L.L.Peterson, "TCP Vegas: New techniques for congestion detection and avoidance," in *Proceedings of ACM SIGCOMM '94*, 1994.

[9] T. Kelly, "Scalable TCP: Improving performance in. highspeed wide area networks," in *Proceedings of PFLDnet*, 2003.

[10] Z. Zhang, G. Hasegawa, and M. Murata, "Analysis and improvement of HighSpeed TCP with TailDrop/RED routers," in *Proceedings of IEEE/ACM International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, Oct. 2004.

[11] K. Tokuda, G. Hasegawa, and M. Murata, "Performance analysis of HighSpeed TCP and its improvement for high throughput and fairness against TCP Reno connections," in *Proceedings of IEEE High Speed Network Workshop 2003 (HSN '03), (San Francisco)*, Mar. 2003.

[12] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, "Modeling TCP throughput: A Simple Model and it Empirical Validation," in *Proceedings of ACM SIGCOMM '98, Vancouver, B.C.*, Sept. 1998.

[13] K. Fall and S. Floyd, "Simulation-based Comparisons of Tahoe, Reno, and SACK TCP," *Computre Communication Review, vol. 26, pp. 5-21*, July 1996.

[14] "NS simulator." available at `http://www.isi.edu/nsnam/ns/`.

[15] S. Floyd, "HighSpeed TCP for large congestion windows," *RFC 3649*, Dec. 2003.

[16] E. Blanton, M. Allman, K. Fall, and L. Wang, "A conservative selective acknowledgment (SACK)-based loss recovery algorithm for TCP," *RFC3517*, Apr. 2003.