Master's Thesis

Title

Ill-effects of Tampered-TCP Flows and Protection Mechanisms for Well-behaved TCP Flows

Supervisor

Professor Hirotaka Nakano

Author

Junichi Maruyama

February 14th, 2007

Graduate School of Information Science and Technology

Osaka University

Master's Thesis

Ill-effects of Tampered-TCP Flows and Protection Mechanisms for Well-behaved TCP Flows

Junichi Maruyama

Abstract

TCP is a de facto standard transport-layer protocol in the current Internet. Because TCP works at end hosts, malicious users can selfishly modify its behavior. So there are many kind of TCP variants created by malicious users to get higher throughput than the normal TCP. We refer to such modified TCPs as tampered-TCPs. If the number of tampered-TCP connections increases in the network, they may occupy the network bandwidth because of their selfish behavior.

In this thesis, we first investigate the effect of a tampered-TCP through mathematical analysis and simulation experiments, focusing on a tampered-TCP which changes the increase and decrease ratio of the congestion window size during the congestion avoidance phase. By the results of the evaluation, we point out that if tampered-TCP connections with SACK option co-exist with normal TCP connections, the fairness among those connections cannot be kept and the normal TCP connections suffer from low throughput. In a certain network situation, the throughput of the tampered-TCP connection is up to 50 times of that of co-existing TCP Reno connection.

To protect normal TCP connections from tampered-TCP connections, we propose a new mechanism which keeps the fairness among TCP connections at edge routers. The proposed mechanism monitors TCP packets at an edge router and estimates a window size or an average throughput of each TCP connection. By using estimation results, the proposed mechanism assesses whether each TCP connection is tampered or not and drops packets intentionally if necessary to improve the fairness among TCP connections. From the results of simulation experiments, we exhibit that the proposed mechanism can identify tampered-TCP connections at high probability and regulate throughput ratio between tampered-TCP connections and competing TCP Reno connections to around 1.

Keywords

Transmission Control Protocol (TCP)

tampered-TCP

Congestion window

Network Monitoring

Fairness

Contents

List of Figures

List of Tables

1 Introduction

Currently, most Internet traffic is carried by 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 traffic increase. 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 roughly fair throughput despite the minor implementation differences of the protocol in the two operating systems.

However, since TCP works at end hosts, it is easy for users to modify its behavior, especially for those with open source operating systems such as Linux [3, 4]. Thus, it is likely that there exists many kind of TCP variants created by malicious users for higher than normal throughput [5, 6]. In this thesis, we 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 the performance when the proposed TCP and TCP Reno connections share the network bandwidth is also evaluated for assessing the deployment path of the proposed TCP [7–13]. However, malicious users can selfishly modify TCP behavior, focusing only on increasing their own throughput. When the population of tampered-TCP connections increases in a network, therefore, these tampered-TCP connections may unfairly occupy network bandwidth, causing the normal TCP connections 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 may *self-destruct*, when its behavior causes it to send data packets more aggressively than normal TCP Reno connections.

In this thesis, we first investigate the effects of a tampered-TCP on a network shared with normal TCP Reno connections to determine whether the tampered-TCP exhibits self-destructive behavior or not under various situations. In addition, proofs showing that the low cost modification of TCPs without SACK option [14] 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 [15, 16].

We employ mathematical analysis and simulation experiments to evaluate tampered-TCP characteristics. For the mathematical analysis, the analysis presented in [15] is extended to derive the average throughput of the TCP Reno and tampered-TCP connections when they share a bottleneck link, and, hence, explain how the parameters of the tampered-TCP affect its performance and fairness against normal TCP Reno connections. The accuracy of the mathematical analysis is confirmed using simulation experiments. Based on the results of the analysis and simulation experiments, the following characteristics of the tampered-TCP are 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 effects of lowering the decrease ratio is less than ill-effects encountered when the increase ratio is augmented. As the result, we conclude that little region exists where the tampered-TCP without SACK option can improve the throughput.

However, it is not a reasonable assumption that malicious users do not use SACK option and there are many recent OSes that enable SACK option in default settings [17–19]. So we also evaluate the effects of tampered-TCP with SACK option and show that it works quite effectively in large network parameter region. In other words, with SACK option, the tampered-TCP connection can obtain high throughput by depressing the throughput of competing TCP Reno connections. Since tampered-TCPs are TCP variants that are modified at end hosts, we need additional mechanisms in the network, that is, on network routers, for protecting the normal TCP Reno connections from such tampered-TCP connections. In [20], the authors proposed a router mechanism which controls UDP traffic to realize TCP-friendliness [21]. However, this mechanism is not intended to control TCP traffic. Because TCP traffic behaves adaptively in packet loss events, whereas UDP traffic does not change its transmission speed against the network congestion, a new mechanism for controlling TCP traffic is necessary. In addition, the authors of [20] do not specify how to estimate parameters used to calculate estimated throughput.

For the second contribution of this thesis, therefore, we propose a new mechanism to keep the fairness among TCP connections at edge routers, for protecting the normal TCP Reno connections from tampered-TCP connections. There are two reasons that the proposed mechanism is supposed to work not at core routers, but at edge routers. One reason is that the number of TCP connections passing through edge routers is smaller than core routers, which results in lower processing overhead to monitor and control TCP connections. Another reason is that we can avoid too many packets from tampered-TCP connections from being injected into the network.

The proposed mechanism estimates a window size or an average throughput of each TCP connection by monitoring TCP packets at an edge router, and assesses its tampering property based on the estimation results. In case of estimating window size (we call *cwnd-based method* in this thesis), the increase ratio α and decrease ratio β of the congestion window size during the congestion avoidance phase are estimated. If the estimated α and β do not satisfy the conditions for achieving similar throughput to the normal TCP Reno connections, the TCP connection is assessed as a tampered-TCP. On the other hand, in case of estimating throughput (*throughput-based method* in this thesis), we obtain the average throughput of each TCP connections by using the traditional per-flow network monitoring tools such as sFlow [22] and NetFlow [23]. We also monitor packet loss rate, RTT, and other parameters for estimating a throughput when the TCP connection would be a TCP Reno. If the observed throughput is larger than the estimated throughput, the TCP connection is assessed as a tampered-TCP. For both methods, the packets belonging to a tampered-TCP connection are dropped intentionally at the edge router with an appropriate probability to regulate its throughput to the same value as the TCP Reno connections.

We evaluate the proposed mechanism by simulation experiments using ns-2 [24]. We employ the throughput ratio as a metric to check the fairness among TCP Reno and tampered-TCP connections. In addition, we use following three metrics to examine the performance of the proposed mechanism: false negative ratio, detection time of tampered-TCP connections, and false positive ratio. By results of the evaluations, we show that the proposed mechanism can identify tampered-TCP connections at high probability and regulate throughput ratio between tampered-TCP connections and competing TCP Reno connections to around 1.

The rest of this thesis is organized as follows. In Section 2, we evaluate the effects of a tampered-TCP through the mathematical analysis and simulation experiments. In Section 3, we show the design of the proposed mechanism to keep the fairness among TCP connections at edge routers. We also present the evaluation results of proposed mechanism by simulation experiments. We finally conclude this thesis and present future works in Section 4.

2 Evaluation of the effects of Tampered-TCP

In this section, we investigate the effects of a tampered-TCP, which changes its increase and decrease ratio of the congestion window size during the congestion avoidance phase, through mathematical analysis and simulation experiments. We also examine the effects of tampered-TCP with SACK option through simulation experiments.

2.1 Mathematical Analysis of Tampered-TCP Throughput

2.1.1 Network Model and Evaluation Metric

Figure 1 depicts the network model that is used for mathematical analysis and simulation experiments. The network model consists of sender and receiver hosts using TCP Reno connections, sender and receiver hosts using tampered-TCP connections, two routers (R_{A} and R_{B}) with a drop-tail buffer, and links interconnecting the hosts and routers. The bandwidth of the link between the router R_{A} and the router R_{B} is μ Mbps, the buffer size at the router R_{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 μ_{T} Mbps, and that between the TCP Reno hosts and the routers is μ_{R} Mbps. There are n_{T} tampered-TCP connections and n_{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 effects of the tampered-TCP, the throughput ratio was introduced as an evaluation metric. It is defined as:

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

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

2.1.2 Behavior of TCP Reno and Tampered-TCP connections

When triggered by a packet loss event, TCP Reno will change its congestion window size [15, 16]. Figure 2 shows a typical case in a network where both TCP Reno and tampered-TCP connections co-exist. Here, we denote 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

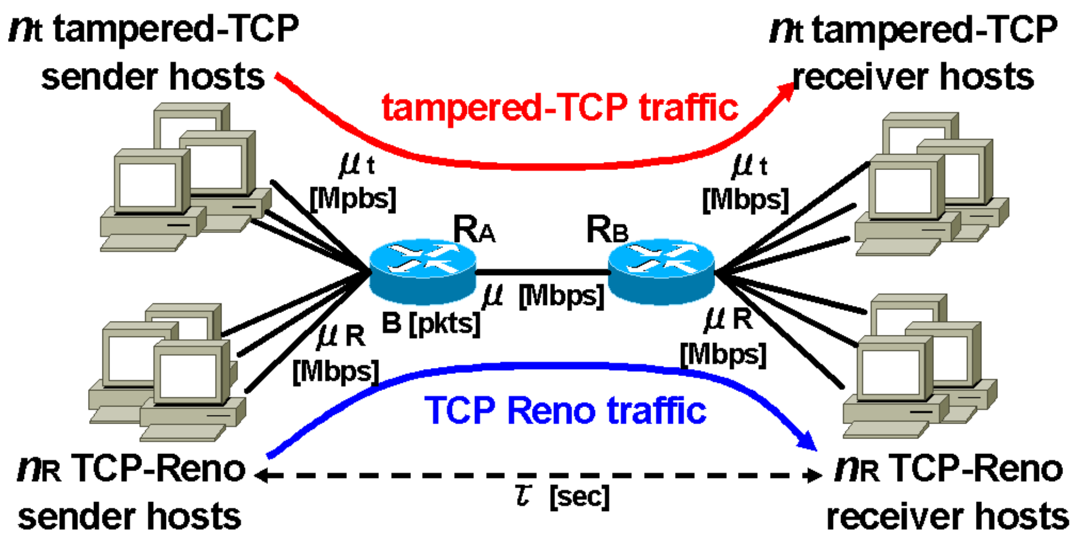


Figure 1: Network model

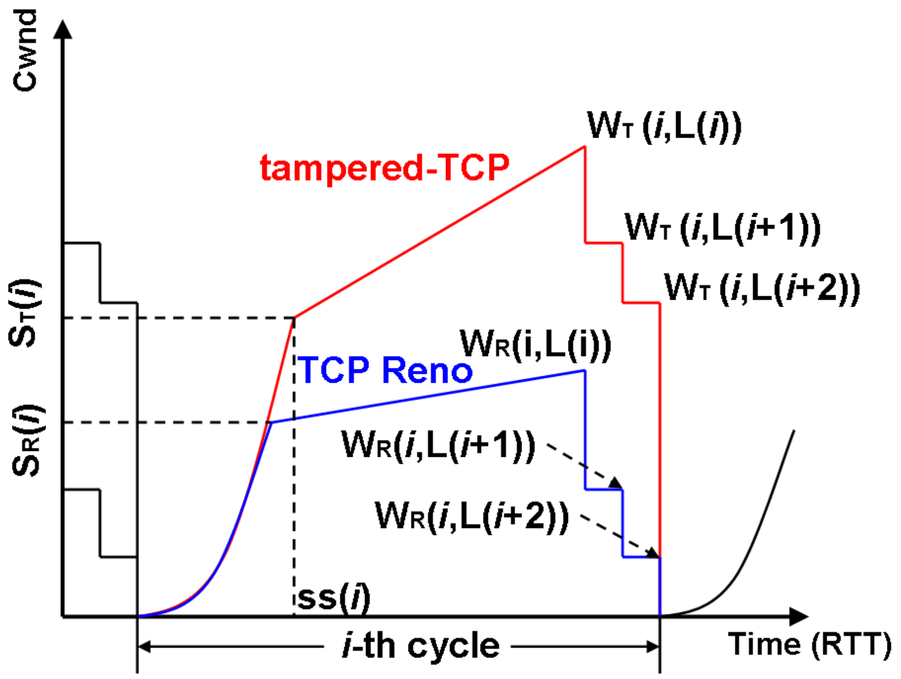


Figure 2: Changes in the congestion window size during the i-th cycle

window size for each RTT. The congestion window size of a TCP Reno connection at the j-th RTT of the i-th cycle is denoted as $W_{\mathrm{R}}(i, j)$.

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 $W_{\mathrm{R}}(i, j)$ by $1/W_{\mathrm{R}}(i, j)$ packets when it receives an ACK packet. Focusing on the change of the congestion window size in every RTT, $W_{\mathrm{R}}(i, j)$ can be derived as follows:

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

where $S_{\mathrm{R}}(i)$ 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 $S_{\mathrm{R}}(i)$ to half of the congestion window size just before the detection of the packet losses.

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 connection at the j-th RTT of the i-th cycle was denoted as $W_{\mathrm{T}}(i, j)$. Then, when a tampered-TCP receives an ACK packet, it increases its congestion window size by $\alpha \cdot 1/W_{\mathrm{T}}(i, j)$, that is α times faster than TCP Reno. This behavior can be described as follows:

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

where $S_{\mathrm{T}}(i)$ 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.

2.1.3 Analysis

In the analysis, the cyclic changes in the congestion window size for TCP Reno and tampered-TCP 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:

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

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

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

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

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:

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

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

Next, the congestion window size of each TCP 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_{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 [25]. 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:

$$\begin{aligned} W_{\mathrm{R}}(i, L(i)+1) &= \tfrac{W_{\mathrm{R}}(i, L(i))}{2} \\ W_{\mathrm{T}}(i, L(i)+1) &= \beta W_{\mathrm{T}}(i, L(i)) \end{aligned}$$

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

$$\begin{aligned} W_{\mathrm{R}}(i, L(i)+2) &= \tfrac{W_{\mathrm{R}}(i, L(i)+1)}{2} \\ W_{\mathrm{T}}(i, L(i)+2) &= \beta W_{\mathrm{T}}(i, L(i)+1) \end{aligned}$$

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

$$\begin{aligned} S_{\mathrm{R}}(i+1) &= \tfrac{W_{\mathrm{R}}(i, L(i)+2)}{2} \\ S_{\mathrm{T}}(i+1) &= \beta W_{\mathrm{T}}(i, L(i)+2) \end{aligned} \tag{6}$$

Let us now consider the congestion window size of the tampered-TCP connection 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 $D_{\mathrm{T}}(i) = \alpha$. 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 connection 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 $D_{\mathrm{T}}(i-1) = \alpha$, the tampered-TCP retransmits α packets. Thus, the congestion window size at the beginning of the i-th cycle, $W_{\mathrm{T}}(i,j)$, is given by:

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

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

$$W_{\mathrm{T}}(i,j) = \beta^{\alpha} W_{\mathrm{T}}(i-1, L(i-1)) + \alpha j \tag{7}$$

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 $D_{\mathrm{T}}(i-1) = \alpha$, 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, $W_{\mathrm{T}}(i,j)$, is derived as follows:

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

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

$$ss(i) = \lfloor \log_2(\beta^2 W_{\mathrm{T}}(i-1, L(i-1))) \rfloor \tag{9}$$

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

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

Finally, the average throughput for TCP Reno and tampered-TCP connections based on the congestion window size evolutions is 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 $max((n_{\mathrm{R}} W_{\mathrm{R}}(i,j) + n_{\mathrm{T}} W_{\mathrm{T}}(i,j) - 2\tau\mu), 0)$, the queuing delay, $Q(i,j)$, is derived as follows:

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

Therefore, the average throughput of TCP Reno connection ρ_{R} and tampered-TCP connection ρ_{T} is:

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

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

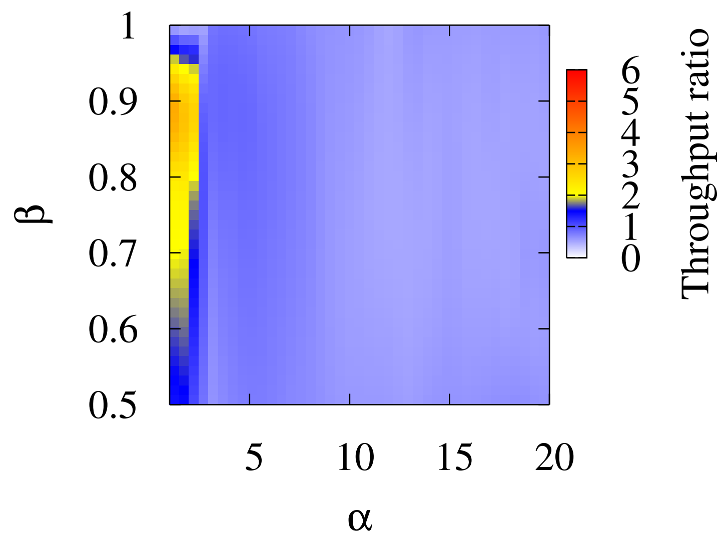
2.2 Simulation Experiments and Discussions

In this subsection, first, simulation experiments are presented that confirm the accuracy of the mathematical analysis developed in the previous subsection 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 we set $\mu_{\mathrm{R}} = \mu_{\mathrm{T}} = 100$ Mbps, $\tau = 20$ msec, the buffer size of the bottleneck link is twice the bandwidth-delay product between the sender and receiver hosts, and the packet size is 1500 bytes. The simulation time was 60 seconds. α, the increase ratio of the congestion window size of the tampered-TCP connection, is changed in [1,20], and β, the decrease ratio of the congestion window size of the tampered-TCP connection, is changed in [0.5,1.0]. We use a ns-2 [24] for the simulation experiments.

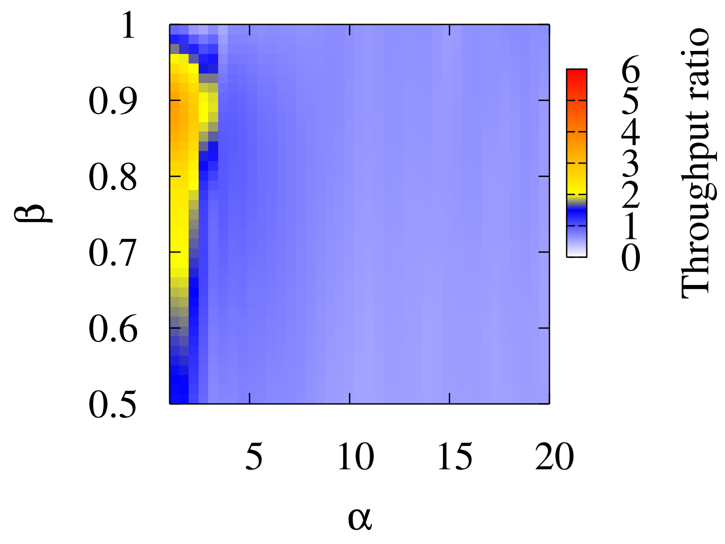
2.2.1 Confirmation of Analysis Results

Figure 3 shows the change in the throughput ratio, defined by Equation (1), as a function of α and β, where we set $n_{\mathrm{R}} = n_{\mathrm{T}} = 1$ 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 connection fails to obtain a large throughput compared with the normal TCP Reno connection 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 connection 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.

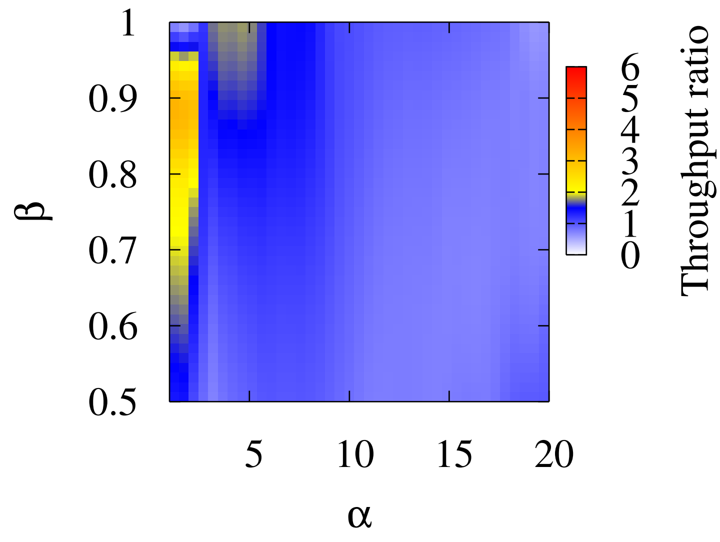


(a) Analysis

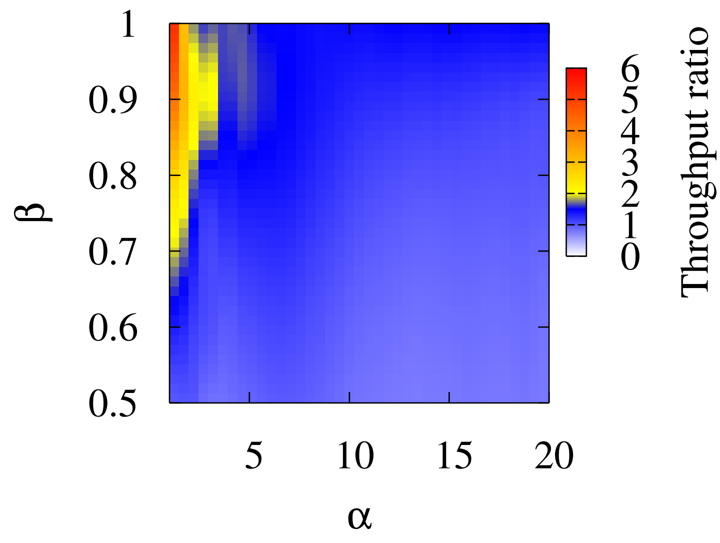


(b) Simulation

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)

2.2.2 Characteristics of Tampered-TCP

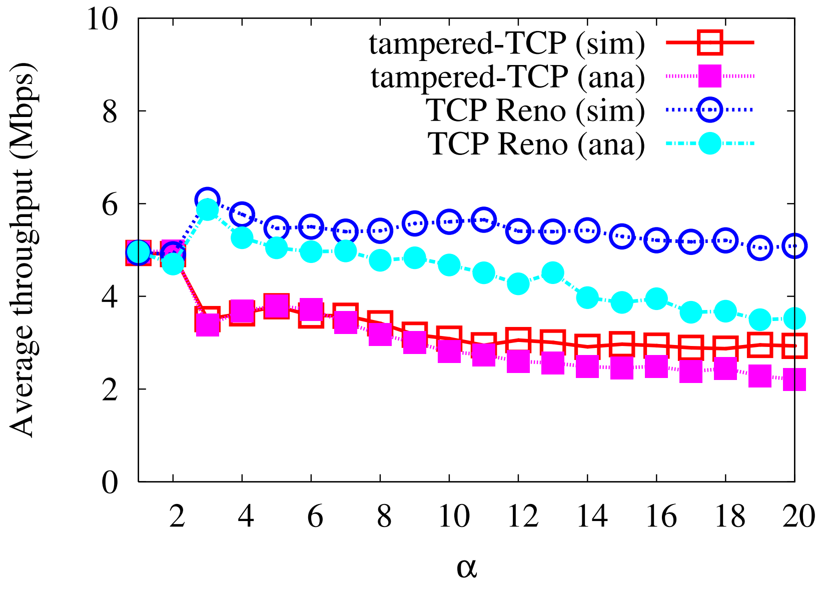
This subsection helps to explain the ineffectiveness of the tampered-TCP by presenting the relevant analytical results, which were confirmed by simulations. Furthermore, the assumptions of the mathematical analysis are validated.

Sensitivity to α and β In this paragraph, 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.

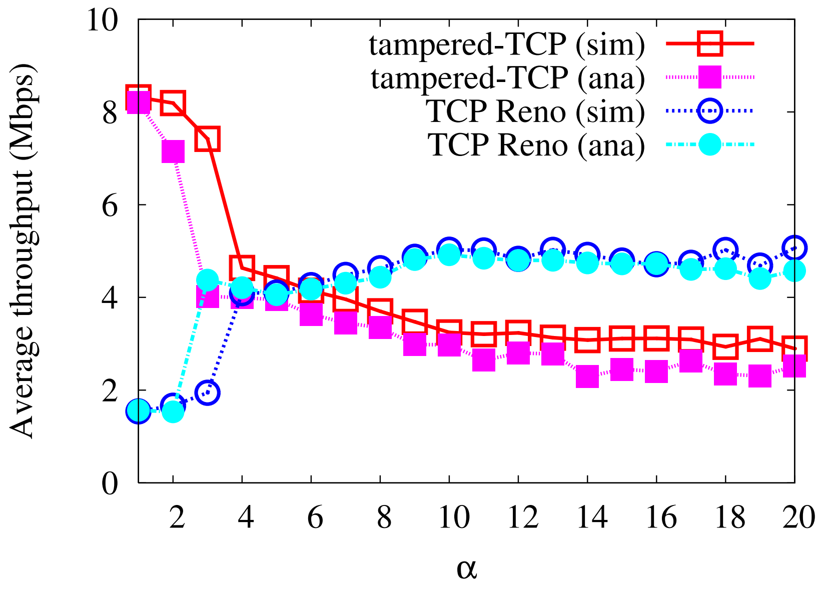
Figure 5(a) plots the change of the average throughput of the TCP Reno and the tampered-TCP connections for both the analytic and simulation cases as a function of the value of α when β is set to 0.5. We show both of analysis and simulation results in this figure. A sharp decrease in throughput occurs for the tampered-TCP connection 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 5(b) shows the results when β is increased to 0.9, which means that the tampered-TCP connection decreases the window size by only 10% when a packet loss occurs. This figure shows that for α smaller than 3, the tampered-TCP connection 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 effects 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 suggests that TCP variants for high-speed and long-distance network such as HSTCP [26] may not work well in such networks. Furthermore, with parameter sets in the effective region, it is obvious that original TCP Reno connections 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.



(a) $\beta = 0.5$



(b) $\beta = 0.9$

Figure 5: Changes in the throughput of TCP Reno and tampered-TCP connections with various α and β

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 paragraph considers the situation where such *well-configured* tampered-TCPs proliferate in a network, and thus diminish its effects.

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 TCP connections is 11 and $\mu = 10$ Mbps, and (2) The total number of TCP 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 paragraph. This figure shows that as the number of tampered-TCP connections increases, the effects sharply diminishes. In the case of 110 TCP connections, the throughput ratio decrease below 1.0, which implies that using a tampered-TCP leads to self-destruction in its performance.

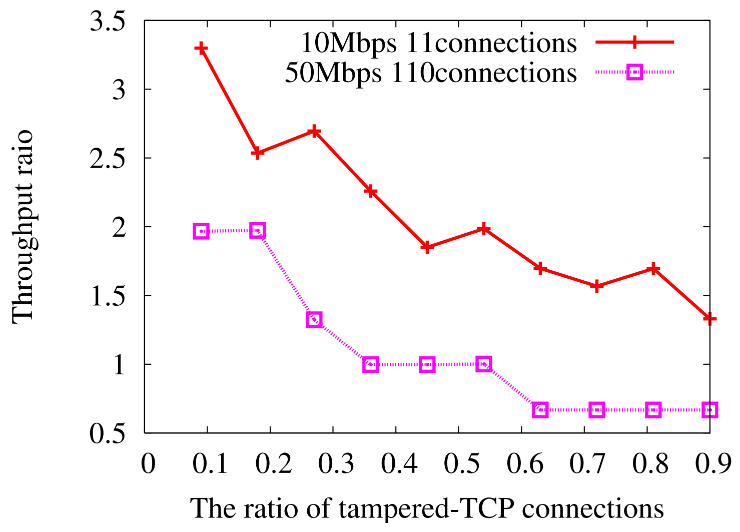


Figure 6: Throughput ratio as a function of the ratio of tampered-TCP connections

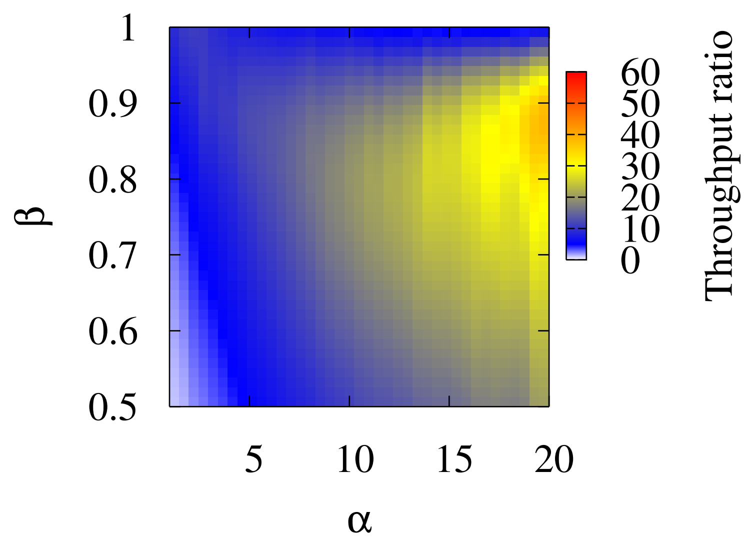
2.2.3 Effects of Tampered-TCP with SACK Option

We have shown in the above subsection that tampered-TCP without SACK option self-destructs in terms of its throughput in many network parameter regions. However, most of current OSes enable SACK option in default settings [17–19] and malicious users must use tampered-TCP with SACK option. In this subsection, therefore, we evaluate the effects of tampered-TCP with SACK option by simulation experiments.

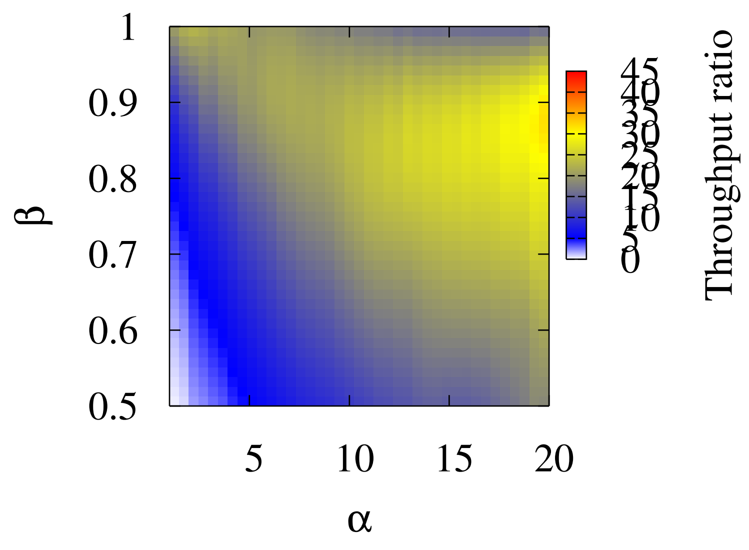
We focus on the tampered-TCP which changes the increase ratio α of the congestion window size with SACK option and keeps the decrease ratio β to 0.5. We use the network model shown in Figure 1 where we set $\mu_{\mathrm{R}} = \mu_{\mathrm{T}} = 100$ Mbps, $\mu = 100$ Mbps, $\tau = 20$ msec, B = 667 packets, and the packet size is 1500 bytes. The simulation time is 60 seconds.

Figure 7 shows the change in the throughput ratio as a function of α and β, when the number of TCP connection is 10 and 30. For each case, 10 % of all the TCP connections are tampered-TCP. This figure shows that the tampered-TCP connection with SACK option does not self-destruct, but unfairly occupies the network bandwidth. As a result, TCP Reno connections suffer from quite low throughput. This means that the fairness among TCP connections can not be kept because of existence of the tampered-TCP connection with SACK option.

The tampered-TCP is modified by malicious users at end hosts. Therefore, we believe that we need a mechanism to protect normal TCP connections from the tampered-TCP connection in the network, that is, on network routers.



(a) 10 connections



(b) 30 connections

Figure 7: Changes in throughput ratio between tampered-TCP with SACK option and TCP Reno connections

3 Protection Mechanism for Well-behaved TCP Flows

3.1 Design of Proposed Mechanism

Figure 8 depicts the overall behavior of the proposed mechanism. By monitoring TCP packets at an edge router, the proposed mechanism detects tampered-TCP connections using estimation results. To protect TCP Reno connections, the proposed mechanism intentionally drops packets of tampered-TCP connections at an appropriate probability that is set to regulate its throughput to become identical to that of the normal TCP Reno connections.

We propose two methods which differ in the metric for assessing the tampering property of TCP connections: a congestion window size and an average throughput. We refer them to *cwnd-based method* and *throughput-based method*, respectively.

In the following subsections, we show the detailed mechanisms in both methods, in terms of estimation mechanism of window size and average throughput, conditions for assessing tampering property, and algorithms for determining the target packet discarding probabilities.

3.1.1 Cwnd-based Method

The cwnd-based method monitors TCP packets passing through the edge router and estimates the window size of each TCP connection continuously. In addition, the increase ratio α and decrease ratio β for the TCP connection for changing the congestion window size during the congestion avoidance phase are estimated based on changes in estimated window sizes. If the estimated α and β indicate that a TCP connection unfairly obtains higher throughput than competing TCP Reno connections, the TCP connection is assessed as a tampered-TCP connection and regulated its throughput based on an appropriate packet discarding probability.

(1) Estimating window size of a TCP connection

Generally, TCP sends packets in a window in bursty fashion. Therefore, as shown in Figure 9, the interval between the last packet of a window and the first packet of the next window is longer than intervals of packets in a burst. By detecting a boundary of two windows divided by such a long interval, the proposed mechanism counts the number of packets sent by sender TCP in each window and estimates the change in the window size.

For that purpose, the proposed mechanism records arrival intervals of every successive

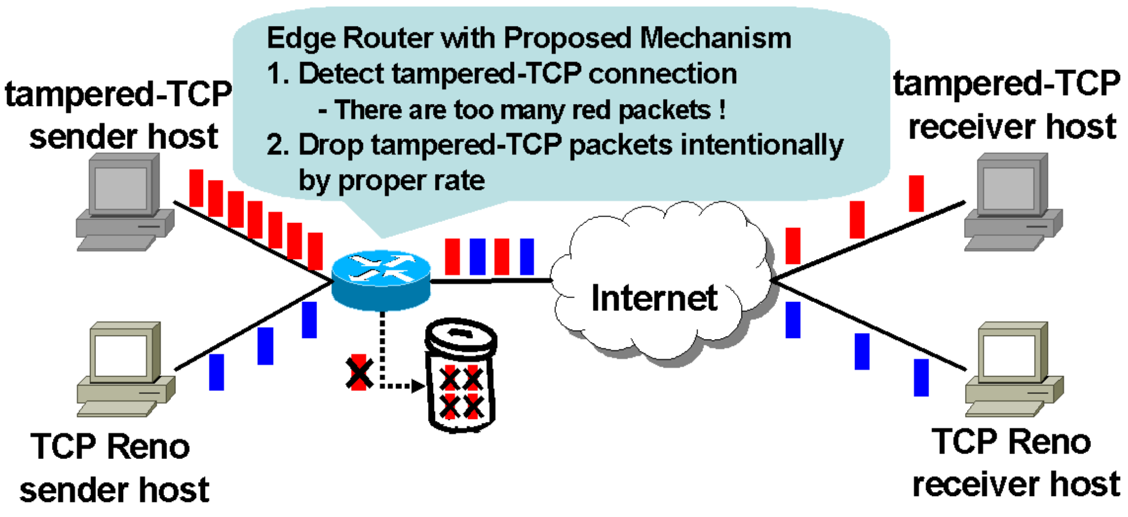

Edge Router with Proposed Mechanism
1. Detect tampered-TCP connection
- There are too many red packets !
2. Drop tampered-TCP packets intentionally by proper rate
tampered-TCP sender host
tampered-TCP receiver host
Internet
TCP Reno sender host
TCP Reno receiver host

Figure 8: Overview of the proposed mechanism

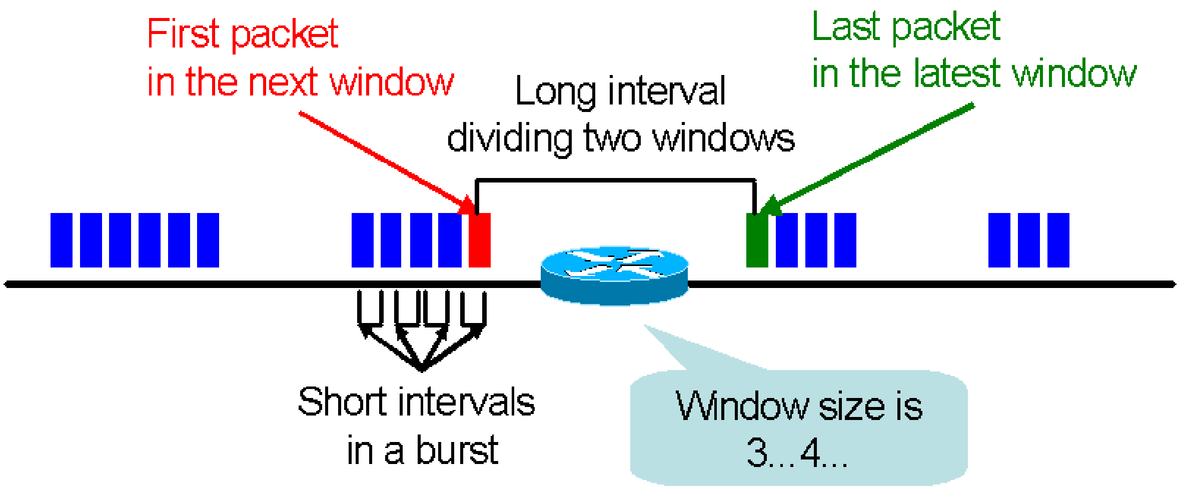

First packet in the next window
Long interval dividing two windows
Last packet in the latest window
Short intervals in a burst
Window size is 3...4...

Figure 9: Detecting the boundary of two successive windows

two packets in a TCP connection and observes the change in the arrival intervals. To observe the change in the arrival intervals, we use an algorithm in [27], which proposes a general method to detect an abrupt change in observed values, as shown in the following equation:

$$g_k = (1 - \delta)g_{k-1} + \delta(y_k - \bar{y})^2$$

This equation calculates the exponential moving average of the squared value of difference between the latest observed value y_k and its average $\bar{y}$ using a smoothing parameter δ $(0 \leq \delta \leq 1)$. If this value is larger than a threshold h, we can determine that an abrupt change happens. In the proposed mechanism, y_k corresponds to be the k-th arrival interval and $\bar{y}$ corresponds to be the average value of the arrival intervals. Detecting the abrupt change in the arrival intervals, an estimated value of the window size can be derived. By this mechanism, we can obtain roughly one estimation result of the window size of a TCP connection per RTT.

On the other hand, [28] proposes a different mechanism to estimate the RTT of TCP connections at routers in the network. By estimating the RTT based on this mechanism, a router counts the number of arriving packets in an RTT and estimates the window size of each TCP connection. However, we do no utilize this method because of its necessity to have the precise RTT value for each TCP connection. One of the advantages of the cwnd-based method we propose here is that there is no need to estimate the RTT of each TCP connection.

(2) Estimating α and β

If the window size of a TCP sender decreases after a packet loss event, the estimated window size at the edge router also decreases. Here, we denote the interval from just after a decrease of the estimated window size caused by a packet loss event to just before the decrease of the estimated window size caused by the next packet loss event as a cycle, as shown in Figure 10. The estimated window size at the j-th RTT of the c-th cycle is denoted as $W_e(c, j)$.

To obtain α, we calculate $\alpha_e(c, j)$, which is a difference between two successive estimated window sizes as follows:

$$\alpha_e(c, j) = W_e(c, j) - W_e(c, j - 1)$$

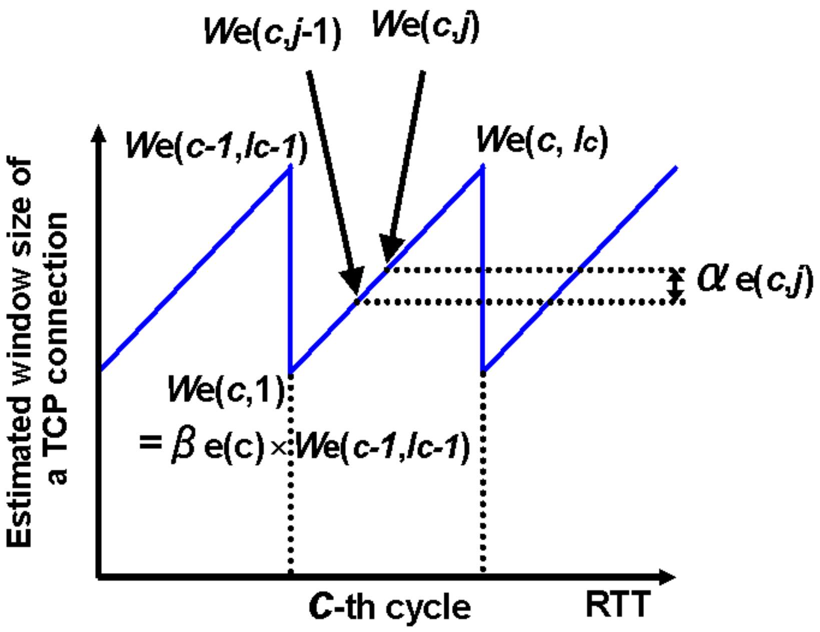


Figure 10: Cyclic changes in the estimated window size of a TCP connection at the edge router

At the end of each cycle, we derive the average value of $\alpha_e(c, j)$ as follows:

$$\overline{\alpha_e}(c) = \frac{\sum_{j=1}^{l(c)} \alpha_e(c, j)}{l(c)}$$

where $l(c)$ is the number of samples of the estimated window size in the c-th cycle. For the current estimation value of α, we derive the exponentially weighted moving average (EWMA) of $\overline{\alpha_e}(c)$, which is denoted as $\overline{\alpha_e}$, as follows:

$$\overline{\alpha_e} = (1 - \gamma_\alpha)\overline{\alpha_e} + \gamma_\alpha \overline{\alpha_e}(c)$$

where γ_α is a smoothing parameter.

For β, we calculate $\beta_e(c)$, which is the estimated value of β in the c-th cycle, from the decrease degree of the window size on a packet loss event:

$$\beta_e(c) = \frac{W_e(c, 1)}{W_e(c - 1, l(c - 1))}$$

We then derive the EWMA of $\beta_e(c)$ values as a current value of β_e:

$$\overline{\beta_e} = (1 - \gamma_\beta)\overline{\beta_e} + \gamma_\beta \beta_e(c)$$

where γ_β is a smoothing parameter.

(3) Estimating packet loss rate

The cwnd-based method estimates a packet loss rate using the information administrated by Management Information Base (MIB) [29] at the edge router. MIB normally stores the number of packets passed through the router and the number of dropped packets at the router. Therefore, by assuming that the edge router implementing the proposed mechanism be a bottleneck, the packet loss rate derived from the MIB information is roughly the same as the packet loss rate which TCP connections passing through the router actually experience. Note that when the different router in the network is the bottleneck, this method underestimates a packet loss rate of TCP connections. This degrades the accuracy of the control mechanism proposed in this subsection. In this case, we should deploy another mechanism such as in [30] for estimating a packet loss rate for each TCP connection, whereas we utilize the MIB-based method for its simplicity. For future work, we plan to compare the MIB-based method and the method in [30] from the perspective of the estimation accuracy and the processing overhead.

When tampered-TCP connections with larger increase ratio of the congestion window size co-exist with normal TCP Reno connections, the packet loss rate at the router increases. As mentioned in Section 2, the number of dropped packets in a tampered-TCP connection is proportional to its increase ratio, α, of the congestion window size. Therefore, the proposed mechanism should estimate the packet loss rate when all the TCP connections passing through the router are supposed to be TCP Reno. We can then calculate the target packet discarding probability for tampered-TCP connections.

We denote the number of dropped packets at the router as n_d, the number of all the packets passed through the router as n_a, and the average value of α_e for all the TCP connections passing through the router as $\bar{A}_e$. Then we estimate the packet loss rate, p, as follows:

$$p = \frac{\frac{n_d}{n_a}}{\bar{A}_e}$$

For averaging p, we utilize the following EWMA calculation:

$$\overline{p} = (1 - \gamma_d)\overline{p} + \gamma_d p$$

where γ_d is a smoothing parameter. Note that we derive new values of p and $\bar{p}$ every when a new value for the target packet discarding probability is calculated.

(4) Assessing tampering property

In [7], the authors extended the equation in [16] for an average throughput of a TCP connection for arbitrary values of α and β. They also derived that when the following equation is satisfied, the TCP connection obtains as the same throughput as a normal TCP Reno connections:

$$\alpha = \frac{4(1-\beta^2)}{3}$$

By using the above equation, we assess a TCP connection is a tampered-TCP when its $\overline{\alpha_e}$ and $\overline{\beta_e}$ satisfy the following equation:

$$\frac{4(1-\overline{\beta_e}^2)}{3\overline{\alpha_e}} < (1-\gamma_w) \tag{11}$$

where γ_w $(0 < \gamma_w < 1)$ is a parameter to allow the estimation error of $\overline{\alpha_e}$ and $\overline{\beta_e}$. Note that the above assessment of tampering property of the TCP connection is repeated when every r_w packets of the TCP connection arrives at the router. r_w is given by:

$$r_w = \frac{k_w}{\overline{p}}$$

where k_w is a positive integer parameter.

(5) Setting the target packet discarding probability

The proposed mechanism sets a target packet discarding probability p' for each TCP connection assessed as tampered-TCP to regulate its throughput to roughly the same as TCP Reno connections. In setting p', we focus on the change in the congestion window size of a TCP Reno connection in the situation where all the TCP connections passing through the router are supposed to be TCP Reno. Here, the TCP Reno connection in such situation is called as a pseudo TCP Reno connection. We calculate p' as to equalize the throughput of the pseudo TCP Reno connection with that of the regulated tampered-TCP connection.

Figure 11 shows the typical changes in the congestion window sizes of the pseudo TCP Reno connection and the tampered-TCP connection with the target packet discarding probability. The number of the packets that a pseudo TCP Reno sender sends in a cycle is $\frac{1}{\overline{p}}$. Because this value is equal to the shadow area in Figure 11(a), the following equation is satisfied:

$$\frac{1}{2} \cdot (W_R + \frac{1}{2}W_R) \cdot \frac{1}{2}W_R = \frac{1}{\overline{p}} \tag{12}$$

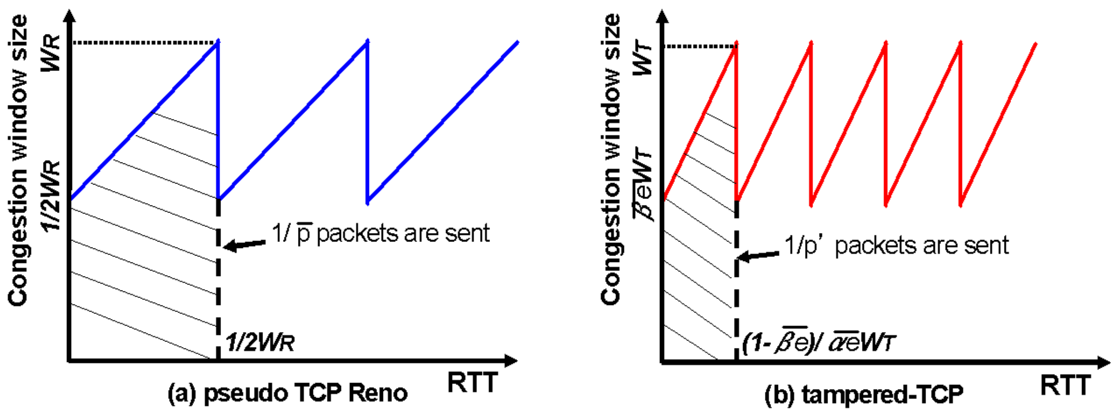


Figure 11: Setting the target packet discarding probability in the cwnd-based method

where W_R is the estimated window size of the pseudo TCP Reno connection at the beginning of the cycle. For the tampered-TCP connection, the similar equation is satisfied:

$$\frac{1}{2} \cdot (W_T + \overline{\beta_e} W_T) \cdot \frac{(1 - \overline{\beta_e})}{\overline{\alpha_e}} W_T = \frac{1}{p'} \tag{13}$$

where W_T is the estimated window size of the tampered-TCP connection at the beginning of the cycle. Therefore, when the throughput of the tampered-TCP connection is identical to the pseudo TCP Reno connection, we obtain the following equation:

$$\frac{\frac{1}{\overline{p}}}{\frac{1}{2} W_R} = \frac{\frac{1}{p'}}{\frac{(1-\overline{\beta_e})}{\overline{\alpha_e}} W_T} \tag{14}$$

From Equations (12)-(14), the target packet discarding probability can be obtained as follows:

$$p' = \frac{(1 + \overline{\beta_e})}{3(1 - \overline{\beta_e})} \overline{\alpha_e} \overline{p}$$

Note that the target packet discarding probability is calculated every when u_w packets of the TCP connection arrives at the router. u_w is given by:

$$u_w = \frac{1}{p'}$$

3.1.2 Throughput-based Method

The throughput-based method monitors the throughput of each TCP connection and regulate tampered-TCP connections at regular intervals. We denote the interval as a control interval. In each control interval, an *observed throughput* is derived based on the information from traditional traffic monitoring tools like sFlow [22] and NetFlow [23]. In addition, we estimate the network parameters, such as RTT, packet loss ratio, and so on, to determine the throughput when we suppose the TCP connection would be a TCP Reno. We call this throughput as an *estimated throughput*. If the observed throughput is larger than the estimated throughput, we assess that the TCP connection is NOT TCP Reno, but tampered-TCP, and regulate its throughput based on a target packet discarding probability.

(1) Setting the control interval

The control interval is the time for $n_I(i)$ packets arriving at the router. $n_I(i)$ is derived as follows:

$$n_I(i) = \frac{k_t}{p(i)}$$

where $p(i)$ is an estimated packet loss rate at the beginning of the i-th control interval and k_t is a positive integer parameter.

(2) Calculating the observed throughput

The traffic monitoring tools generally store the total bytes of packets passed through the router and the traffic monitoring time for each flow passing through the router. We denote the total number of bytes in the i-th control interval as $b(i)$, the length of the i-th control interval as $t(i)$ and the observed throughput in the i-th control interval as $T_o(i)$. Then $T_o(i)$ is given by the following equation:

$$T_o(i) = \frac{b(i)}{t(i)}$$

(3) Calculating the estimated throughput

The equation proposed in [16] which estimates the throughput of a TCP connection has the following parameters: packet size, delayed ACK option value, RTT, retransmission timeout, and packet loss rate. To calculate the estimated throughput when we suppose the TCP connection would be a TCP Reno, all the parameters are estimated as follows:

- Packet size

  The traffic monitoring tools store amounts of traffic arrived at the router in both units of packets and bytes. We denote the total number of packets in the i-th control interval as $n(i)$ and the estimated packet size as $s_e(i)$. Then $s_e(i)$ can be calculated as follows:

$$s_e(i) = \frac{b(i)}{n(i)}$$

- The delayed ACK option value

  We denote the ACK sequence number of the j-th ACK packet as $a(i, j)$. Using the difference between these two ACK sequence numbers, the estimated value of the delayed ACK option $del_e(i, j)$ is given by:

$$del_e(i, j) = a(i, j) - a(i, j - 1)$$

  The average number of the $del_e(i, j)$ in the i-th control interval is denoted as $\overline{del_e}(i)$. $\overline{del_e}(i)$ is derived as follows:

$$\overline{del_e}(i) = \frac{\sum_{j=1}^{n_b} del_e(i, j)}{n_b}$$

  where n_b is the number of samples of the estimated delayed ACK option values in the i-th control interval. Here, we ignore duplicate ACK packets and ACK packets just after the duplicate ACK packets for calculation, because the ACK sequence numbers of such ACK packets is not appropriate for determining the delayed ACK option value.

- RTT

  Though many kind of mechanisms are proposed to estimate the RTT in past papers [31–33], we choose the mechanism proposed in [28] which utilizes TCP's timestamp option [34]. This mechanism estimates the RTT as follows. Figure 12 depicts the time chart of packet transfers between a sender and a receiver via a router. The sender transmits a TCP data packet dp_1 with timestamp ts_1. It arrives at the router at time m_1. The receiver responds with an ACK packet ap_1 with timestamp ts_2 and the echo ts_1. The router recognizes ts_1 in both the packet dp_1 and ap_1, then makes an association between the two packets. On receiving the ACK packet ap_1, the sender

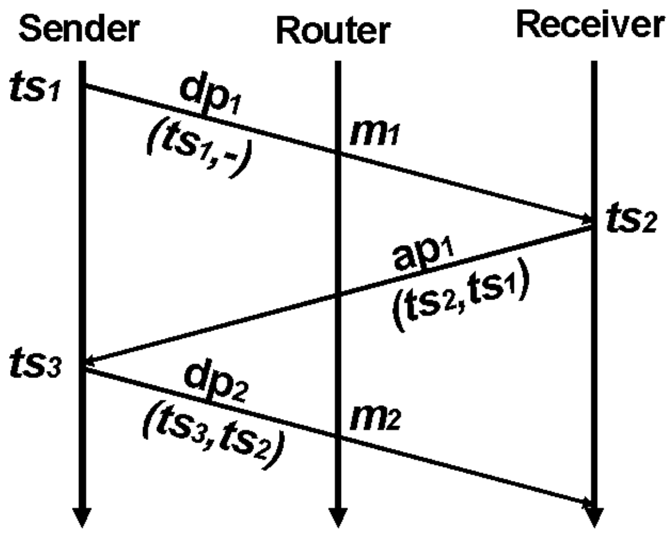


Figure 12: RTT estimation using timestamp values

transmits a new data packet dp_2 with timestamp ts_3 and the echo ts_2. The router receives the packet dp_2 at time m_2 and recognizes ts_2 in both the packet ap_1 and dp_2, then makes an association between the packet $ap1$ and $dp2$. Having three associated packets, the router estimates the RTT using m_1 and m_2. The j-th estimated RTT in the i-th control interval $rtt_e(i,j)$ is given by:

$$rtt_e(i,j) = m_2 - m_1$$

The average value of the $rtt_e(i,j)$ in the i-th control interval is derived as follows:

$$\overline{rtt_e}(i) = \frac{\sum_{j=1}^{n_r} rtt_e(i,j)}{n_r}$$

where n_r is the number of samples of the estimated RTTs in the i-th control interval.

- Retransmission timeout

  [35] recommends that 4 times of the RTT is used as an estimated value of the retransmission timeout. We utilize the same method for estimating the retransmission timeout $rto_e(i)$:

$$rto_e(i) = 4\overline{rtt_e}(i)$$

- packet loss rate

  The estimated packet loss rate is derived almost the same as the cwnd-based method.

However, because the throughput-based method does not estimate the increase ratio α of the congestion window size of each TCP connection, the packet loss rate observed when all the TCP connections passing through the router are supposed to be TCP Reno can not be estimated. So We simply calculate the packet loss rate $p(i)$ from $n_d(i)$ and $n_a(i)$ as follows:

$$p(i) = \frac{n_d(i)}{n_a(i)}$$

In case of the number of co-existing TCP connections is small, this equation overestimates a packet loss rate of TCP connections. However, the number of TCP connections passing through the router increases and the ratio of tampered-TCP connections decreases relatively, the effect of the overestimation becomes small. The estimated packet loss rate is smoothed according to the following EWMA calculation:

$$\overline{p}(i) = (1 - \gamma_l)\overline{p}(i-1) + \gamma_l p(i)$$

We finally derive the estimated throughput $T_e(i)$ in the i-th control interval as follows:

$$T_e(i) = \frac{s_e(i)}{\overline{rtt_e}(i)\sqrt{\frac{2\overline{del_e}(i)\overline{p}(i)}{3}} + rto_e(i)\min\left(1, 3\sqrt{\frac{3\overline{del_e}(i)\overline{p}(i)}{8}}\right)\overline{p}(i)(1 + 32\overline{p}(i)^2)}$$

.

(4) Assessing tampering property

The throughput-based method assesses a TCP connection as tampered-TCP if its $T_o(i)$ and $T_e(i)$ in the i-th control interval satisfy the following equation:

$$\frac{T_o(i)}{T_e(i)} > (1 + \gamma_t) \tag{15}$$

where γ_t $(0 < \gamma_t)$ is a parameter to allow the estimation error of $T_o(i)$ and $T_e(i)$. Note that the above assessment of tampering property of the TCP connection is repeated every control interval, which reduces the effect of assessment miss.

(5) Setting the target packet discarding probability

For setting the target packet discarding probability $p'(i)$ in the i-th control interval, we use the property of TCP throughput that the throughput of a TCP connection is proportional to the inverse of the square root of the packet loss rate [21]. Based on this property, $p'(i)$ is given by:

$$p'(i) = \left(\frac{T_o(i-1)}{T_e(i-1)}\right)^2 p'(i-1)$$

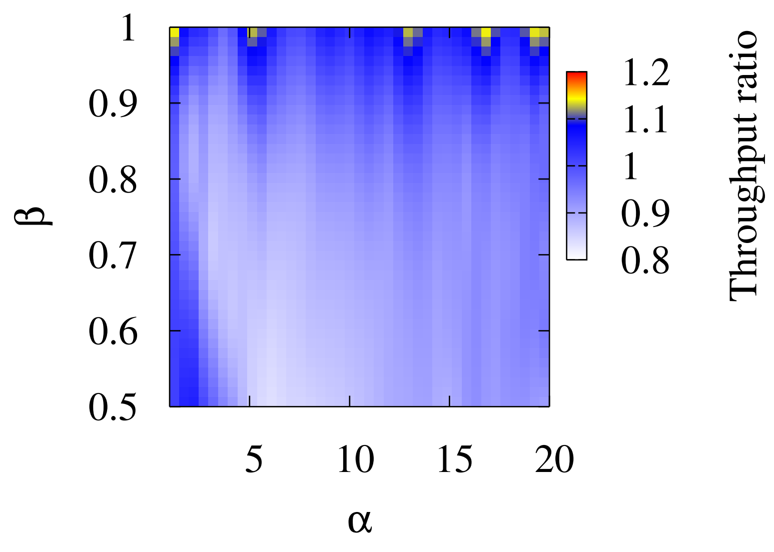
3.2 Simulation Experiments of Proposed Mechanism

In this subsection, we present simulation results to evaluate the performance of the proposed mechanism described in Subsection 3.1. The control parameters for the cwnd-based method are set as $\delta = 0.6$, $h = 0.0001$, $\gamma_\alpha = 0.6$, $\gamma_\beta = 0.6$, $\gamma_d = 0.6$, $\gamma_w = 0.1$, and $k_w = 4$. The control parameters for the throughput-based method are set as $\gamma_l = 0.6$, $\gamma_t = 2$, and $k_t = 4$.

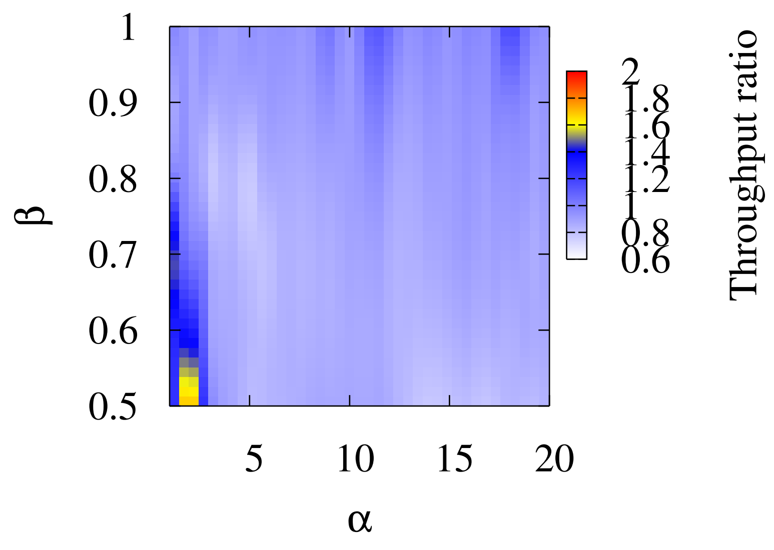
The simulation model is shown in Figure 1 where we set μ_{R} = μ_{T} = 100 Mbps, $\mu = 50$ Mbps, $\tau = 20$ msec, $B = 333$ packets, $n_T = 1$, $n_R = 20$, and the packet size is set to 1500 bytes. The simulation time is 60 seconds. We evaluate the performance of the proposed mechanism when α, the increase ratio of the congestion window size of the tampered-TCP connections, is changed in [1,20] and β, the decrease ratio of the congestion window size of the tampered-TCP connection, is changed in β [0.5,1.0]. We use the throughput ratio, false negative ratio, detection time of tampered-TCP connections, and false positive ratio for the evaluation metrics. The detection time is the time that the proposed mechanism takes to detect tampered-TCP connections.

3.2.1 Throughput Ratio

Figure 13 plots the change in the throughput ratio of the cwnd-based method and throughput-based method. Figure 13(a) shows that the cwnd-based method keeps the throughput ratio around 1 for almost all the parameters. Figure 13(b) shows that when using the throughput-based method, the throughput ratio is larger than 1 around the point $(\alpha, \beta) = (1, 0.5)$, where the tampering property of tampered-TCP connections is weak. This is because the parameter γ_t is used to allow the estimation error in Equation (15), causing that tampered-TCP connections in this region are sometimes assessed as normal TCP Reno. However, the throughput ratio is kept around 1 in other region.



(a) Cwnd-based method



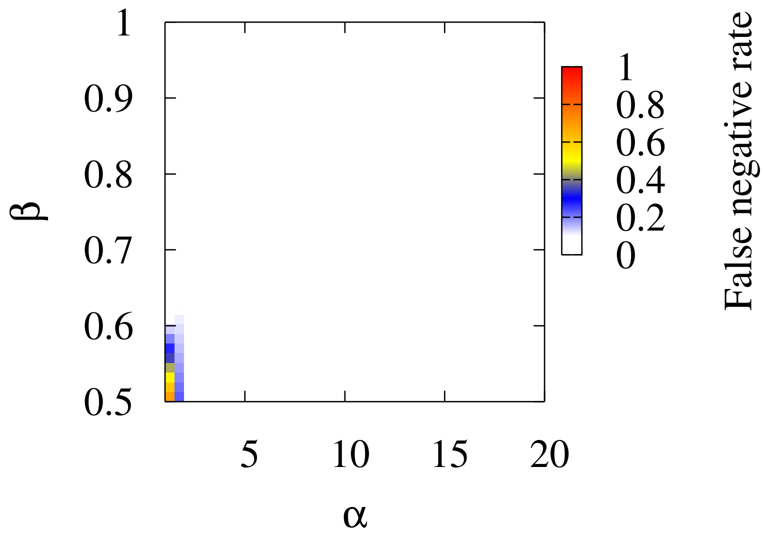
(b) Throughput-based method

Figure 13: Changes in throughput ratio when using the proposed mechanism

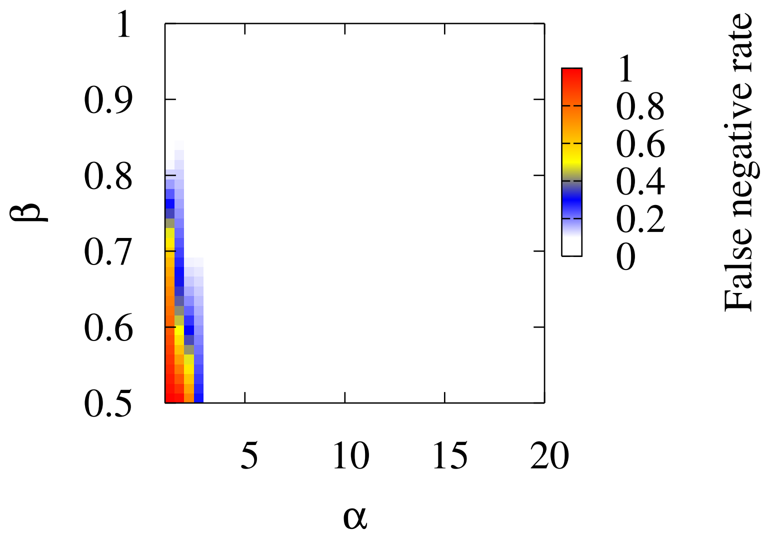
3.2.2 False Negative Ratio and Detection Time

Figures 14 and 15 show changes in false negative ratio and detection time of the cwnd-based method and throughput-based method. Figure 14 shows that the both method have almost the same trend of the false negative ratio. In the region around $(\alpha, \beta) = (1, 0.5)$, which corresponds to TCP Reno's increase and decrease ratio of the congestion window size, the false negative ratio is nearly 0. This means that the proposed mechanism does not assess a normal TCP Reno connection as a tampered-TCP connection. In addition, in the region where tampering property of the tampered-TCP connections is weak, the false negative ratio becomes high. This is because the parameter γ_w and γ_t are used to allow the estimation error in Equations (11) and (15), tampered-TCP connections in this region are sometimes assessed as normal TCP Reno. However, tampered-TCP connections are detected at almost 100 % in other region.

Figure 15 shows that the cwnd-based method takes about 2 seconds to detect the tampered-TCP connections, and about 5 seconds for the the throughput-based method. Though it takes a little longer to detect tampered-TCP connections in the region where tampering property of tampered-TCP connections is weak, even in that case it takes only about 10 seconds. Currently, when ISP monitors and detects connections that uses large bandwidth, the MIB information is mainly used, and the typical update interval of the MIB information is 5 minutes, so we can say that the proposed mechanism detects tampered-TCP connections in quite a short time.

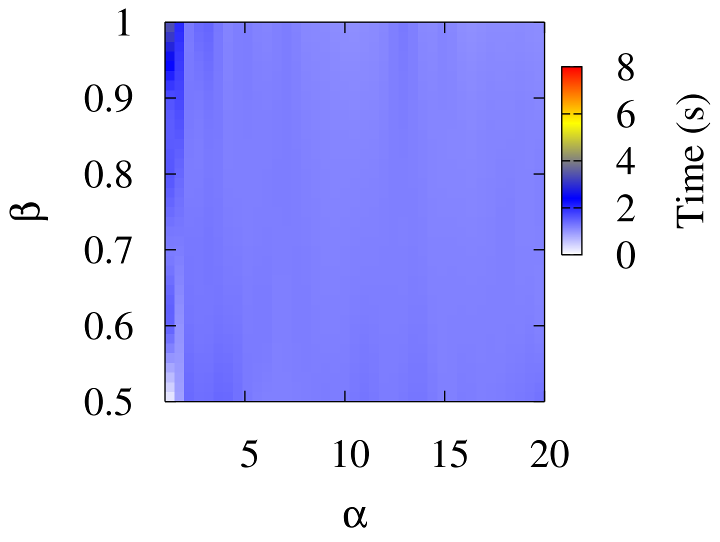


(a) Cwnd-based method

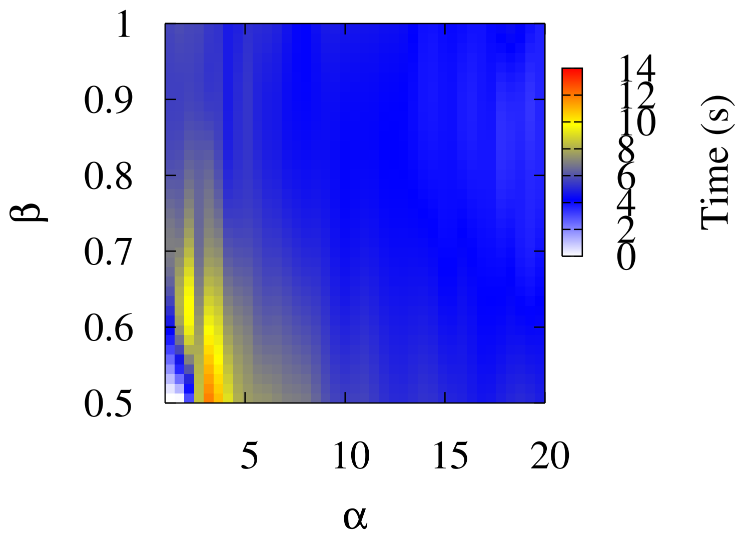


(b) Throughput-based method

Figure 14: False negative ratio of tampered-TCP connections



(a) Cwnd-based method



(b) Throughput-based method

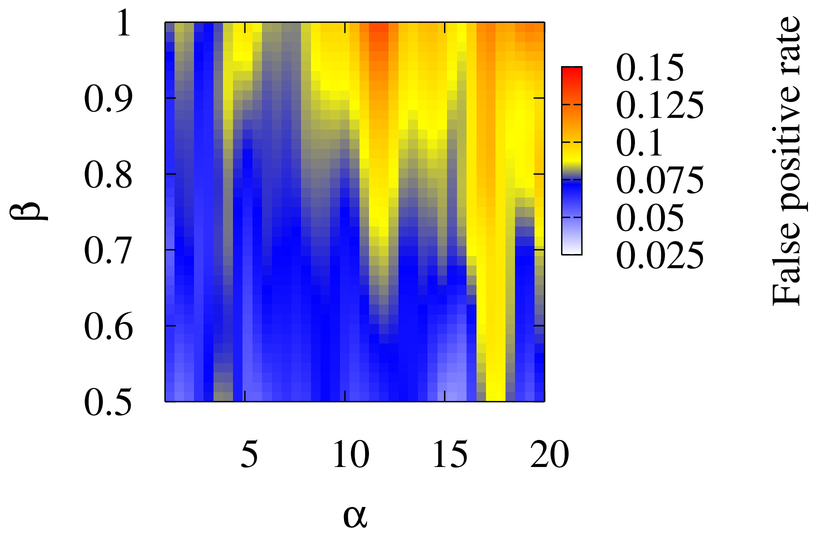
Figure 15: Detection time of tampered-TCP connections

3.2.3 False Positive Ratio

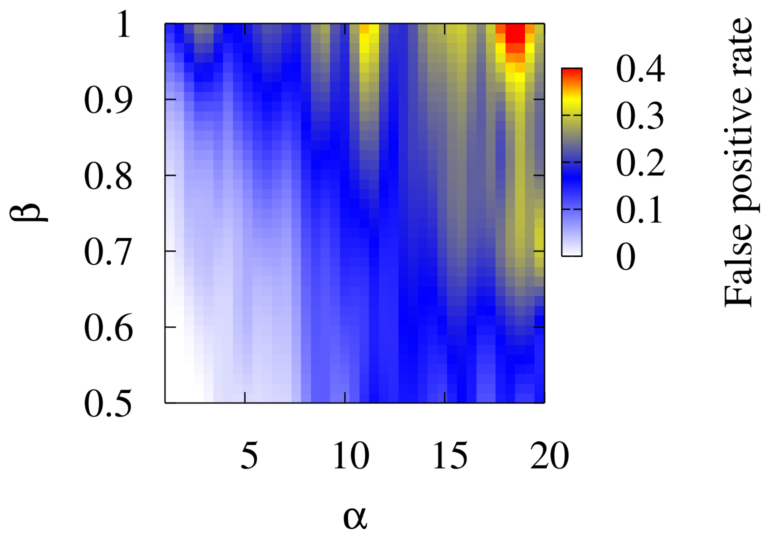
Figure 16 depicts the false positive ratio of the cwnd-based method and throughput-based method. This figure shows that the false positive ratio of both methods increases as the tampering property of tampered-TCP connections becomes stronger. This can be explained as follows. The tampered-TCP in this thesis increases its congestion window size rapidly as its tampering property becomes stronger, which leads the unstable changes in the congestion window size and throughput of the competing TCP Reno connections. This causes the estimation error of the α and β in the cwnd-based method, and the observed throughput and the estimated throughput in the throughput-based method.

However, in case of false positive error, the throughput of miss-assessed TCP Reno connection does not decrease so largely. This is shown by Table 1, which presents the throughput and throughput ratio of miss-assessed TCP Reno connections and successfully-assessed TCP Reno connections.

Therefore, we can say that though the proposed mechanism sometimes miss-assesses TCP Reno connections as tampered-TCP, the effects of miss-assessment is little.



(a) Cwnd-based method



(b) Throughput-based method

Figure 16: False positive ratio of TCP Reno connections

Table 1: Throughput and throughput ratio of miss-assessed TCP Reno and successfully-assessed TCP Reno connections

Cwnd-based method				
α	β	Miss-assessed Reno (Mbps)	Reno (Mbps)	Throughput ratio
10	0.7	2.614	2.329	1.122
20	0.9	2.330	2.391	0.975

Throughput-based method				
α	β	Miss-assessed Reno (Mbps)	Reno (Mbps)	Throughput ratio
10	0.7	2.464	2.358	1.050
20	0.9	2.209	2.405	0.926

3.3 Comparison of Cwnd-based Method with Throughput-based Method

We presented that both of the cwnd-based method and throughput-based method can detect tampered-TCP connections in a few seconds at high probability and keep throughput ratio at around 1. In this subsection, we discuss the characteristics of these two methods and propose the guideline for selecting the methods for given network situations.

3.3.1 Characteristics of Cwnd-based Method

The cwnd-based method estimates α and β, assesses the tampering property, and sets the target packet discarding probability, based on the estimation of the window size of TCP connections. So, in the ideal case, tampered-TCP connections are detected in one cycle. In addition, since this method maintains the estimated α values, the packet loss rate when all the TCPs connections passing through the router are supposed to be TCP Reno can be estimated. Consequently, the appropriate target packet discarding probability can be set in the situation where the number of co-existing TCP connections is small.

On the other hand, to estimate the window size, all arriving packets have to be monitored, which leads to the large processing overhead especially when many TCP connections are passing through the router. In addition, if the boundary of two successive windows can not be detected, α will be overestimated and β will be underestimated, which causes the miss-assessment. This trend will be strong in the situation where RTT of the TCP connections is small and the jitter in RTT is large.

3.3.2 Characteristics of Throughput-based Method

The throughput-based method can reduce the processing overhead by adjusting the number of samples used in estimating parameters. In addition the effect of estimation errors is generally less than the cwnd-based method.

On the other hand, the throughput-based method takes at least one control interval to detect tampered-TCP connections, which is about 3-4 times larger than one cycle in the cwnd-based method. In addition, because it does not estimate α, the packet loss rate when all the TCP connections passing through the router are supposed to be TCP Reno can not be estimated. Then, the target packet discarding probability can not be set appropriately when there are small number of co-existing TCP connections.

3.3.3 Appropriate Network Environments for Both Methods

Table 2 summarize the characteristics of both methods. From this table, we can conclude that the cwnd-based method is suitable when the number of competing TCP connections is small and when the amount of short-lived flows is large. On the other hand, the throughput-based method is good at the situations where there are many TCP connections passing through the router and when the network is comparatively unstable. Therefore, we can conclude that these two methods are complemental to each other.

Table 2: Comparison of cwnd-based method and throughput-based method

	Detection time	Suitable number of connections	Tolerance to estimation error	Processing overhead
Cwnd	short	small	low	large
Throughput	a little long	large	high	small

4 Conclusion

In this thesis, we focused on tampered-TCP, which changes the increase and decrease ratio of the congestion window size during the congestion avoidance phase, and evaluated its effects by mathematical analysis and simulation experiments. For the tampered-TCP without SACK option, the following characteristics was presented: when the increase ratio is larger than 2 packets per RTT, TCP retransmission timeouts occur frequently and the throughput of tampered-TCP diminishes sharply. Lowering the decrease ratio is beneficial whenever the increase ratio is smaller than 3. Futhermore the effect rapidly decreases if there are too many tampered-TCP connections, even when well-configured parameters are used.

However, it is difficult to assume that malicious users do not use SACK option. In addition, there are many OSes that enable SACK option in default settings recently. So we investigated the effects of the tampered-TCP with SACK option. We presented that the tampered-TCP connections with SACK option unfairly occupy the network bandwidth causing normal TCP Reno connections suffer from the low throughput. Because these tampered-TCPs are modified at end host, we pointed out the necessity of a new mechanism to keep the fairness among TCP connections in the network.

We then proposed a new mechanism at edge routers to protect normal TCP connections from tampered-TCP connections. The proposed mechanism estimates a window size or an average throughput of each TCP connection passing through the edge router by monitoring TCP packets, and assesses its tampering property based on the estimation results, and regulates tampered-TCP connections by dropping incoming packets at an appropriate probability.

We evaluated the proposed mechanism by simulation experiments using ns-2. By results of the evaluations, we presented that the proposed mechanism keeps the throughput ratio around 1 and achieve the fairness among TCP connections. We also showed that the proposed mechanism detects tampered-TCP connections at almost 100 % in large parameter regions. The detection time was only about 2 seconds in the cwnd-based method and 5 seconds in the throughput-based method. This was short enough compared with the update interval of MIB information which ISP uses to monitor and control connections. Though the false positive rate of the proposed mechanism was a little high, we explained that its effect is quite a little by showing the throughput and throughput ratio between the miss-assessed and rightly-assessed TCP Reno connections.

For future work, we want to examine the implementation method to routers. We also plan to investigate the performance of the proposed mechanism in the actual Internet environment.

Acknowledgements

My deepest gratitude goes to Professor Hirotaka Nakano of Osaka University for his supervision of this thesis. I greatly appreciate the continuous support and expensive advice he has given to me.

I would like to express my gratitude to Professor Masayuki Murata of Osaka University. In addition to giving me good insights into this thesis, his constant encouragement helped me very much.

I also wish to express my sincere appreciation to Associate Professor Go Hasegawa of Osaka University for his appropriate guidance and invaluable direct advice. All works of this thesis would not have been possible without his support. In addition, he gave me everyday life wisdom as well as many technical advice. I can’t express how grateful I am.

I am most grateful to Professors Koso Murakami, Makoto Imase, Teruo Higashino, and Tetsuji Satoh of Osaka University, for their technical guidance and helpful advice.

I am indebted to Associate Professors Naoki Wakamiya and Research Assistants Shin’ichi Arakawa, Masahiro Sasabe, and Yuichi Oshita of Osaka University, who gave me useful comments and supports.

I also wish to thank Cao Le Thanh Man, Tomoaki Tugawa, and Kana Yamanegi of Osaka University, who gave me variable comments. Their helps greatly contributed to my thesis.

Finally, I want to say thanks to all my friends and colleagues in the Department of Information Networking of the Graduate School of Information Science and Technology of Osaka University for their support. Our conversations and work together have greatly influenced this thesis.

References

[1] M. Fomenkov, K. Keys, D. Moore, and K. Claffy, "Longitudinal study of Internet traffic from 1998-2003," in *Proceedings of WISICT 2004*, Jan. 2004.

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

[3] S. Bokhari, "The Linux operating system," *IEEE Computer*, vol. 28(8), pp. 74–79, Aug. 1995.

[4] I. Phillips and J. Crowcroft, *TCP/IP and Linux Protocol Implementation: Systems Code for the Linux Internet (Networking Council Series)*. John Wiley and Sons Inc, 2001.

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

[6] M. Baldi, Y. Ofek, and M. Yung, "Idiosyncratic signatures for authenticated execution of management code," in *Proceedings of DSOM 2003*, Oct. 2003.

[7] Y. R. Yang and S. S. Lam, "General AIMD congestion control," in *Proceedings of ICNP 2000*, Nov. 2000.

[8] L. Mamatas and V. Tsaoussidis, "Protocol behavior : More effort, more gains?," in *Proceedings of PIMRC 2004*, Sept. 2004.

[9] 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*, Aug. 1994.

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

[11] Z. Zhang, G. Hasegawa, and M. Murata, "Analysis and improvement of HighSpeed TCP with TailDrop/RED routers," in *Proceedings of MASCOTS 2004*, Oct. 2004.

[12] K. Tan, J. Song, Q. Zhang, and M. Sridharan, "Compound TCP: A scalable and TCP-friendly congestion control for high-speed networks," in *Proceedings of PFLDnet 2006*, Feb. 2006.

[13] H. Shimonishi, M. Sanadidi, and T. Murase, "Assessing interactions among legacy and high-speed TCP protocols," in *Proceedings of PFLDnet 2007*, Feb. 2007.

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

[15] 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*, Mar. 2003.

[16] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, "Modeling TCP throughput: A simple model and its empirical validation," in *Proceedings of ACM SIGCOMM '98*, Sept. 1998.

[17] J. Padhye and S. Floyd, "On inferring TCP behavior," *ACM SIGCOMM Computer Communication Review*, vol. 31(4), pp. 287–298, Aug. 2001.

[18] K. Pentikousis and H. Badr, "Quantifying the deployment of TCP options - a comparative study," *IEEE Communications Letters*, vol. 8(10), pp. 647–649, Oct. 2004.

[19] M. Mellia, R. L. Cigno, and F. Neri, "Measuring IP and TCP behavior on edge nodes with Tstat," *Computer Networks*, vol. 47(1), pp. 1–21, Jan. 2005.

[20] S. Floyd and K. Fall, "Router mechanisms to support End-to-End congestion control," *Technical report, Lawrence Berkeley Laboratory, Berkeley, CA*, Feb. 1997.

[21] J. Padhye, J. Kurose, D. Towsley, and R. Koodi, "Model based TCP-friendly rate control protocol," in *Proceedings of NOSSDAV' 99*, June 1999.

[22] P. Phaal, S. Panchen, and N. McKee, "InMon corporation's sFlow: A method for monitoring traffic in switched and routed networks," *RFC 3176*, Sept. 2001.

[23] "NetFlow." available at `http://www.cisco.com/japanese/warp/public/3/jp/product/hs/ios/nmp/prodlit/pdf/iosnf_ds.pdf`.

[24] "The Network Simulator - ns-2." available at `http://www.isi.edu/nsnam/ns/`.

[25] K. Fall and S. Floyd, "Simulation-based Comparisons of Tahoe, Reno, and SACK TCP," *ACM SIGCOMM Computer Communication Review*, vol. 26(3), pp. 5–21, July 1996.

[26] S. Floyd, “HighSpeed TCP for large congestion windows,” *RFC 3649,* Dec. 2003.

[27] M. Basseville and I. Nikiforov, *Detection of abrupt changes: Theory and application*. Prentice-Hall,Inc, 1993.

[28] B. Veal, K. Li, and D. K. Lowenthal, “New methods for passive estimation of TCP round-trip times,” in *Proceedings of PAM 2005*, pp. 121–134, Mar. 2005.

[29] K. McCloghrie and M. Rose, “Management information base for network management of TCP/IP-based Internets: MIB-II,” *RFC1213,* Mar. 1991.

[30] P. Benko and A. Veres, “A passive method for estimating end-to-end TCP packet loss,” in *Proceedings of IEEE GLOBECOM 2002*, Nov. 2002.

[31] H. Jiang and C. Dovrolis, “Passive estimation of TCP round-trip times,” *ACM Computer Communication Review*, vol. 32(3), pp. 75–88, Aug. 2002.

[32] G. Lu and X. Li, “On the correspondency between TCP acknowledgment packet and data packet,” in *Proceedings of IMC 2003*, Oct. 2003.

[33] S. Jaiswal, G. Iannaccone, C. Diot, J. Kurose, and D. Towsley, “Inferring TCP connection characteristics through passive measurements,” in *Proceedings of INFOCOM 2004*, Mar. 2004.

[34] V. Jacobson, R. Braden, and D. Borman, “TCP extensions for high performance,” *RFC1323*, May 1992.

[35] M. Handley, S. Floyd, J. Pahdye, and J. Widmer, “TCP friendly rate control (TFRC),” *RFC3448*, Jan. 2003.