Protection Mechanisms for Well-behaved TCP Flows from Tampered-TCP at Edge Routers

Junichi Maruyama+* Go Hasegawa+ Masayuki Murata++
Graduate School of Information Science and Technology, Osaka University
1-5 Yamadaoka Suita, Osaka 565-0871, Japan
Phone: +81-6-6850-6863 Fax: +81-6-6850-6868
+ {maruyama, hasegawa}@ane.cmc.osaka-u.ac.jp
++ murata@ist.osaka-u.ac.jp
* Corresponding author

Abstract— **In this paper, we propose a new mechanism which detects tampered-TCP connections at edge routers and protects well-behaved TCP connections from the tampered-TCP connections, resulting in maintaining the fairness amongst TCP connections. The proposed mechanism monitors the TCP packets at an edge router and estimates the window size or the throughput for 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 amongst TCP connections. From the results of simulation experiments, we exhibit that the proposed mechanism can accurately identify tampered-TCP connections. We also show that the proposed mechanism can regulate throughput ratio between tampered-TCP connections and competing TCP Reno connections to about 1.**

Keywords: Transmission Control Protocol (TCP), tampered-TCP, Congestion window, Network Monitoring, Fairness

I. Introduction

Since Transmission Control Protocol (TCP) works at end hosts, it is easy for users to modify its behavior. This is especially the case for users with open source operating systems such as Linux. Thus, there exists many kind of TCP variants created by malicious users that allow for higher than normal throughput [1, 2]. In this paper, such modified TCPs are referred to as *tampered-TCPs*.

Generally, when modifications to TCP congestion control mechanisms are proposed, the effects of these modifications are compared with the original TCP Reno. Furthermore, for assessing the deployment path of the proposed TCP, the performance when the proposed TCP and TCP Reno connections share the network bandwidth is evaluated [3, 4]. 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 normal TCP connections to suffer from low throughput.

In [5], we evaluated the effects of the tampered-TCP on a network shared with normal TCP Reno connections. We focused on a tampered-TCP which changes the increase and decrease ratio of the congestion window size during the congestion avoidance phase without the SACK option [6] and we presented that there exists little region where the tampered-TCP without the SACK option can improve the throughput. However, it is not a reasonable to assume that a malicious user does not use the SACK option, and there are many recent operating systems that enable the SACK option as a default setting [7–9]. Thus, we also evaluate the effects of tampered-TCP with the SACK option and show that it works quite effectively in large network parameter region. Since tampered-TCPs are TCP variants that are modified at the end hosts, additional mechanisms are needed in the network for protecting normal TCP Reno connections from tampered-TCP connections. One such possible location could be on the network routers.

In [10], the authors proposed a router mechanism that controls UDP traffic to realize TCP-friendliness [11]. However, this mechanism is not intended to control TCP traffic. Since 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 [10] do not specify how to estimate parameters used to calculate estimated throughput. On the other hand, [12, 13] guarantee quality of service (QoS) by dropping packets intentionally at routes based on the class or hop-counting. However, these mechanisms doesn't consider the TCP behavior, therefore, they can't control connections adaptively depending on the TCP implementation version.

In this paper, therefore, we propose a new mechanism that maintains the fairness amongst TCP connections at edge routers, which protects the normal TCP Reno connections from tampered-TCP connections. There are two reasons why the proposed mechanism should be located at the edge routers and not at the core routers. The first reason is that the number of TCP connections passing through edge routers is smaller than through core routers, which results in lower processing overhead to monitor and control TCP connections. The second reason is that this prevents too many packets from tampered-TCP connections from entering the network.

The proposed mechanism estimates a window size or an average throughput for each TCP connection by monitoring the TCP packets at an edge router, and assesses its tampering property based on the estimation results. Then, 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 TCP Reno connections.

We evaluate the proposed mechanism by simulation experiments using ns-2 [14]. Based on the results of these evaluations, it is shown that the proposed mechanism can accurately identify tampered-TCP connections and regulate the throughput ratio between tampered-TCP connections and competing TCP Reno connections to about 1.

II. Effects of Tampered-TCP with SACK Option

In this section, we briefly demonstrate the effects of a tampered-TCP with the SACK option.

Figure 1 depicts the network model that is used for simulation experiments with ns-2 [14]. 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 droptail 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

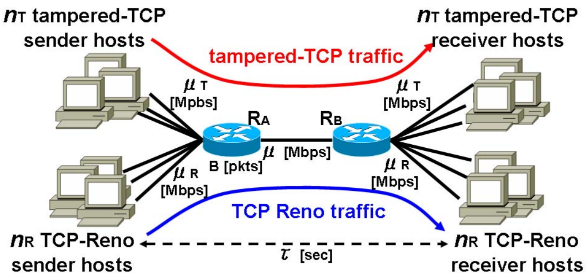
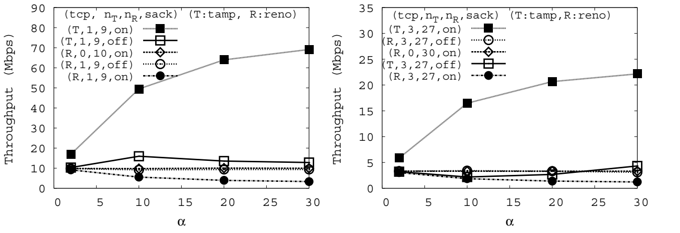


Fig. 1. Network model



(a) 10 connections

(b) 10 connections

Fig. 2. Changes in the throughput of the tampered-TCP with SACK option and TCP Reno connections

$\mu_{\rm T}$ 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. It is assumed 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.

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

Figure 2 shows the change in the throughput as a function of α when the number of TCP connections is set at 10 and 30. For each case, we plot the results of three situations: no tampered-TCP connection, 10 % of all the TCP connections are tampered-TCP without the SACK option, and 10 % of all the TCP connections are tampered-TCP with the SACK option. This figure shows that the fairness is kept in case of no tampered-TCP connection. In addition, the tampered-TCP connections without the SACK option cannot obtain much higher throughput than competing TCP Reno connections. However, the tampered-TCP connections with the SACK option obtain quite a high throughput as α increases which results in depressing the throughput of competing TCP Reno connections.

The tampered-TCP is modified by malicious users at end hosts. Therefore, a mechanism is needed to protect normal TCP connections from tampered-TCP connections in the network. Such a mechanism should be located on the network routers.

III. Design of Proposed Mechanism

Figure 3 depicts the overall behavior of the proposed mechanism. By monitoring the 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 the tampered-TCP connections at an appropriate probability that regulates its throughput to equal that of normal TCP Reno connections.

Note that the proposed mechanism is not based on per-flow queueing. It can be combined with Weighted RED (WRED) mechanism which is equipped in many commercial

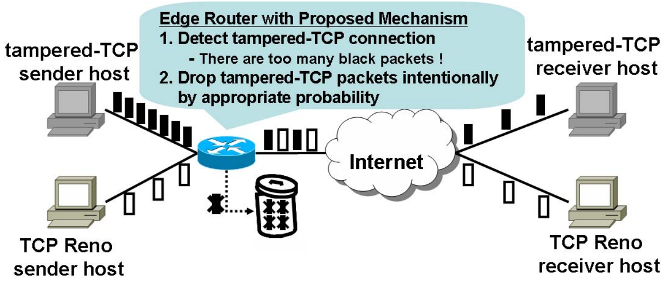

Edge Router with Proposed Mechanism
1. Detect tampered-TCP connection
- There are too many black packets !
2. Drop tampered-TCP packets intentionally by appropriate probability
tampered-TCP sender host
tampered-TCP receiver host
Internet
TCP Reno sender host
TCP Reno receiver host

Fig. 3. Overview of the proposed mechanism

router products, since the proposed mechanism only sets the packet discarding probabilities for tampered-TCP connections to maintain fairness amongst connections. We also note that the regulating mechanism for tampered-TCP connections is only activated on congested routers.

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 to them as *cwnd-based method* and *throughput-based method*, respectively. In the following subsections, a detailed description is given of each method, in terms of estimation mechanism of the window size and the average throughput, conditions for assessing the tampering property, and algorithms for determining the target packet discarding probabilities.

A. Cwnd-based Method

The cwnd-based method monitors the TCP packets passing through the edge router and continuously estimates the window size of each TCP connection. 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 the 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 its throughput is regulated using an appropriate packet discarding probability.

1) Estimating the window size of a TCP connection

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

For that purpose, the proposed mechanism records the arrival intervals of every two successive packets in a TCP connection and observes the change in the arrival intervals. To observe the change in the arrival intervals, algorithm presented in [15], which proposes a general method to detect an abrupt change in observed values is used. This algorithm can be described with 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, an abrupt change is said to occur. In the proposed mechanism, y_k corresponds to the k-th arrival interval and $\bar{y}$ corresponds to 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. Using this mechanism, we can obtain roughly one estimation result of the window size of a TCP connection per RTT.

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, 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 is denoted as a cycle. 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 the difference between two successive estimated window sizes as follows:

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

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, $\beta_e(c)$, which is the estimated value of β in the c-th cycle, from the rate of decrease of the window size in a packet loss event is calculated using:

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

Thus, the EWMA of $\beta_e(c)$ values is derived 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 the packet loss rate using the information administered by the Management Information Base (MIB) [16] 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 is a bottleneck, the packet loss rate derived from the MIB information is roughly the same as the packet loss rate that the TCP connections passing through the router actually experience. Note that when a different router in the network is the bottleneck, this method underestimates the packet loss rate of TCP connections. This lowers the accuracy of the control mechanism proposed in this subsection. However, we believe that the performance degradation is not so large since we activate the proposed mechanism only when the router is congested.

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. In [5], we showed that 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. Thus, the target packet discarding probability for tampered-TCP connections can be determined.

The number of dropped packets at the router is denoted as n_d, the number of all the packets which passed through the router is denoted as n_a, and the average value of α_e for all the TCP connections passing through the router is denoted as $\bar{A}_e$. Thus, the packet loss rate, p, can be estimated as follows:

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

p can be averaged, using the following EWMA calculations:

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

where γ_d is a smoothing parameter. Note new values for p and $\bar{p}$ are calculated whenever a new value for the target packet discarding probability is determined.

4) Assessing the tampering property

In [3], the authors extended the equation in [17] for an average throughput of a TCP connection for arbitrary values of α and β. They also showed that when the following equation is satisfied, the TCP connection obtains the same throughput as a normal TCP Reno connections:

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

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

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

where γ_w $(0 < \gamma_w < 1)$ is a parameter that takes into consideration the estimation error of $\overline{\alpha_e}$ and $\overline{\beta_e}$. Note that the above assessment of the tampering property of the TCP connection is repeated whenever 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 a tampered-TCP to regulate its throughput to roughly the same as TCP Reno connections. In setting p', the focus is 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 a situation is called a pseudo TCP Reno connection. The p' is determined so as to equalize the throughput of the pseudo TCP Reno connection with that of the regulated tampered-TCP connection.

Figure 4 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 packets that a pseudo TCP Reno sender sends in a cycle is $\frac{1}{\overline{p}}$. Because this value is equal to the shaded area in Figure 4(a), the following equation is satisfied:

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

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, a similar equation is satisfied:

$$\frac{1}{2} \cdot (W_T + \overline{\beta_e} W_T) \cdot \frac{(1 - \overline{\beta_e})}{\overline{\alpha_e}} W_T = \frac{1}{p'} \quad (3)$$

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:

$$\frac{\frac{1}{\overline{p}}}{\frac{1}{2}W_R} = \frac{\frac{1}{p'}}{\frac{(1 - \overline{\beta_e})}{\overline{\alpha_e}} W_T} \quad (4)$$

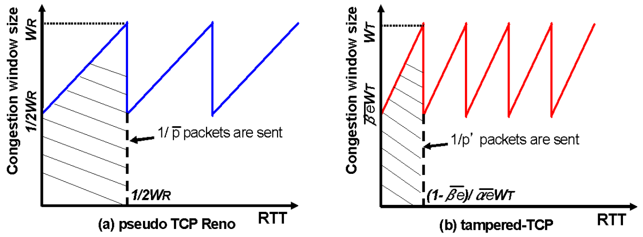


Fig. 4. Setting the target packet discarding probability in the cwnd-based method

From Equations (2)-(4), the target packet discarding probability can be obtained as follows:

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

Note that the target packet discarding probability is calculated whenever u_w packets of the TCP connection arrive at the router. u_w is given by $u_w = \frac{1}{p'}$.

B. Throughput-based Method

The throughput-based method monitors the throughput of each TCP connection and regulates the tampered-TCP connections at regular intervals. This interval is called as the control interval. In each control interval, an *observed throughput* is derived based on the information from traditional traffic monitoring tools like sFlow [18] and NetFlow [19].

In addition, network parameters, such as RTT, packet loss ratio, and so on, are estimated in order to determine the throughput, assuming that the TCP connection is a TCP Reno. This estimated throughput is called an *estimated throughput*. If the observed throughput is larger than the estimated throughput, then the TCP connection is said to be not TCP Reno, but a tampered-TCP, and its throughput is regulated 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

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. The total number of bytes in the i-th control interval is denoted as $b(i)$, the length of the i-th control interval is denoted as $t(i)$ and the observed throughput in the i-th control interval is denoted 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 [17] which estimates the throughput of a TCP connection uses the following parameters: packet size, delayed ACK option value, RTT, retransmission timeout, and packet loss rate. To calculate the estimated throughput if it is assumed that the TCP connection is a TCP Reno connection, all the parameters are estimated as follows:

- Packet size
The traffic monitoring tools store the amount of traffic that arrives at the router in both units of packets and bytes. The total number of packets in the i-th control interval is denoted as $n(i)$, and the estimated packet size is denoted 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
The ACK sequence number of the j-th ACK packet is denoted 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, all duplicate ACK packets and ACK packets just after the duplicate ACK packets are ignored in the calculation, because the ACK sequence numbers of such ACK packets are not appropriate for determining the delayed ACK option value.

- RTT
Though many different kinds of mechanisms have been proposed to estimate the RTT in past papers [20–22], the mechanism proposed in [23], which utilizes TCP's timestamp option [24], is used in this paper. This mechanism estimates the RTT as follows. 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 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 $dp2$. With 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:

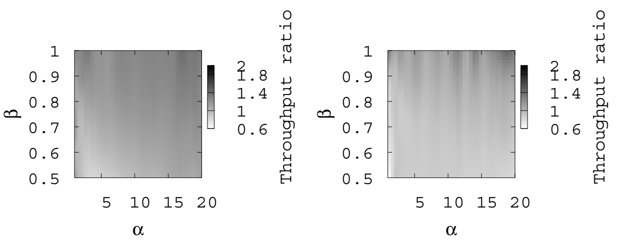
$$\overline{rtt_e}(i) = \frac{\sum_{j=1}^{n_r} rtt_e(i,j)}{n_r}$$

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

- Retransmission timeout
[25] recommends that 4 times of the RTT be used as an estimated value of the retransmission timeout. In this paper, this method is used to estimate the retransmission timeout $rto_e(i)$ as:

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

- Packet loss rate
The estimated packet loss rate is derived in a manner similar to 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. Thus, packet loss



(a) Cwnd-based method (b) Throughput-based method

Fig. 5. Changes in the throughput ratio when using the proposed mechanism

rate $p(i)$ is simply calculated from $n_d(i)$ and $n_a(i)$ as follows:

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

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

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

Finally, the estimated throughput $T_e(i)$ in the i-th control interval is given by:

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

4) Assessing the 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:

$$\frac{T_o(i)}{T_e(i)} > (1 + \gamma_t) \tag{5}$$

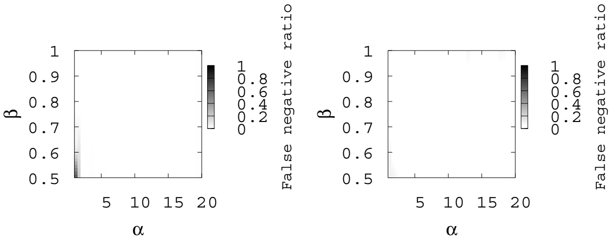
where γ_t $(0 < \gamma_t)$ is a parameter that accounts for error in estimating $T_o(i)$ and $T_e(i)$. Note that the above assessment of the tampering property of the TCP connection is repeated for every control interval, which reduces the effect of assessment misses.

5) Setting the target packet discarding probability
Since the throughput of a TCP connection is proportional to the inverse of the square root of the packet loss rate [11], this can be used to determine the target packet discarding probability $p'(i)$ in the i-th control interval. Based on this property, $p'(i)$ is given by:

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

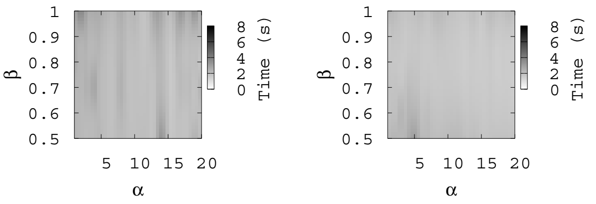
IV. Simulation Experiments of Proposed Mechanism

In this section, the simulation results to evaluate the performance of the proposed mechanism described in Section III are presented. The control parameters for the cwnd-based method are set as $\delta = 0.5$, $h = 0.0001$, $\gamma_\alpha = 0.6$, $\gamma_\beta = 0.6$, $\gamma_d = 0.6$, $\gamma_w = 0.1$, and $k_w = 4$. The control parameters for the throughput-based method are set as $\gamma_l = 0.6$, $\gamma_t = 2$, and $k_t = 6$. The simulation model is shown in Figure 1 where 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 70 seconds. In first 10 seconds, only TCP Reno connections transmit data, and after 10 seconds, the tampered-TCP connection starts transmission.



(a) Cwnd-based method (b) Throughput-based method

Fig. 6. False negative ratio for tampered-TCP connections



(a) Cwnd-based method (b) Throughput-based method

Fig. 7. Detection time for tampered-TCP connections

We use the average values of the evaluation metrics of the last 60 seconds. The performance of the proposed mechanism is evaluated when α, the increase ratio of the congestion window size of the tampered-TCP connections, takes values in the interval [1,20] while β, the decrease ratio of the congestion window size of the tampered-TCP connection, takes values in the interval [0.5,1.0]. The throughput ratio is used as an evaluation metric. It is defined as:

$$Throughput\ ratio = \frac{(Throughput\ of\ \text{tampered-TCP})}{(Throughput\ of\ \text{TCP Reno})} \tag{6}$$

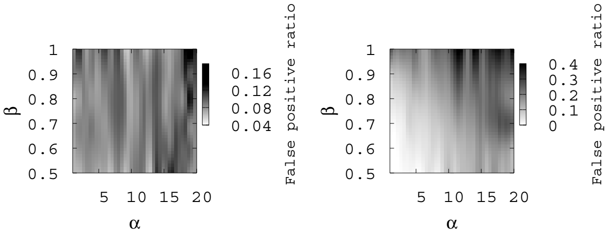
In addition, following three metrics are used to examine the performance of the proposed mechanism: false negative ratio, detection time of tampered-TCP connections, and false positive ratio. The detection time is defined as the time that the proposed mechanism takes to detect tampered-TCP connections.

A. Throughput Ratio

Figure 5 plots the change in the throughput ratio of the cwnd-based method and the throughput-based method. Figure 5(a) shows that the cwnd-based method keeps the throughput ratio about 1 for almost all the parameters. Figure 5(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 account for the estimation error in Equation (5), which causes tampered-TCP connections in this region to occasionally be assessed as normal TCP Reno connections. However, the throughput ratio is kept about 1 in other region.

B. False Negative Ratio and Detection Time

Figures 6 and 7 show changes in the false negative ratio and the detection time for the cwnd-based method and throughput-based method. 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 in the cwnd-based method is nearly 1. This means that the cwnd-based method does not assess a normal TCP Reno connection as a tampered-TCP connection. In addition, in the region where the tampering property of the tampered-TCP connections is weak, the false negative ratio becomes high. This is because the parameter γ_w is used to account for the estimation error in Equations (1), which causes the tampered-TCP connections in this region to occasionally be assessed as normal TCP Reno connections. In case of the throughput-based method, the false



(a) Cwnd-based method (b) Throughput-based method

Fig. 8. False positive ratio for TCP Reno connections

TABLE I
THROUGHPUT AND THROUGHPUT RATIO OF MISASSESSED TCP RENO AND SUCCESSFULLY-ASSESSED TCP RENO CONNECTIONS

Cwnd-based method				
α	β	Misassessed Reno (Mbps)	Reno (Mbps)	Throughput ratio
10	0.7	2.117	2.399	0.882
20	0.9	2.233	2.377	0.939
Throughput-based method				
α	β	Misassessed Reno (Mbps)	Reno (Mbps)	Throughput ratio
10	0.7	2.548	2.401	1.061
20	0.9	2.175	2.431	0.895

negative ratio around $(\alpha, \beta) = (1, 0.5)$ is nearly 0. This is because a normal TCP connection during the slow start phase is sometimes misassessed as a tampered-TCP. However, the effects of the misassessment are small as we will mention later. One of the solutions to this problem is setting a longer control interval than the other ones for the first assessment of the tampering property of each TCP connection. In the other region, tampered-TCP connections are detected at almost 100 % for both methods.

Figure 7 shows that both methods take about 2 seconds to detect the tampered-TCP connections. Currently, when ISP monitors and detects connections that use large bandwidth, the MIB information is mainly used, and the typical update interval of the MIB information is 5 minutes. Thus, it can be said that the proposed mechanism detects tampered-TCP connections much faster.

C. False Positive Ratio

Figure 8 depicts the false positive ratio for 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 modeled in this paper increases its congestion window size rapidly as its tampering property becomes stronger, which leads to unstable changes in the congestion window size and in the throughput of the competing TCP Reno connections. This causes an estimation error for α and β in the cwnd-based method, and for the observed throughput and the estimated throughput in the throughput-based method.

However, in the case of false positive errors, the throughput of misassessed TCP Reno connection does not decrease so largely. This is shown by Table I, which presents the throughput and throughput ratio of misassessed TCP Reno connections and successfully-assessed TCP Reno connections. This result means that the proposed mechanism sets the target packet discarding probability so as to adapt the too aggressive/conservative control of the previous interval. Therefore, a temporary misassessment of normal TCP connections would be fixed in the following intervals, even when some control parameters and monitored parameters are changed.

By these results, it can be said that the proposed mechanism sometimes misassesses TCP Reno connections as tampered-TCP. However, the effects of this misassessment are small.

V. Conclusion

In this paper, we 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 the throughput of tampered-TCP connections by dropping incoming packets at an appropriate probability. By results of the simulation experiments, we presented that the proposed mechanism regulates the throughput ratio about 1 and achieve the fairness amongst TCP connections. For future work, we will evaluate the router overhead of the proposed mechanism in various network condition. We also plan to investigate the performance of the proposed mechanism in the actual Internet environment. In addition, we areinterested in using another TCP variant, which is compound TCP included in Windows Vista, as a well-behaved TCP.

References

[1] 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.

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

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

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

[5] J. Maruyama, G. Hasegawa, and M. Murata, “Is tampered-TCP really effective for getting higher throughput in the Internet?,” in *Proceedings of ATNAC 2006, pp. 167-171*, Dec. 2006.

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

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

[8] 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.

[9] 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.

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

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

[12] C. Dovrolis and P. Ramanathann, “Proportional differentiated services, part II: Loss rate differentiation and packet dropping,” in *Proceedings of the International Workshop on Quality of Service*, 2000.

[13] X. Zhou, D. Ippoliti, and T. Boult, “HPPD: A hop-count probabilistic packet dropper,” in *Proceedings of IEEE ICC*, 2006.

[14] “The Network Simulator - ns-2.” available at `http://www.isi.edu/nsnam/ns/`.

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

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

[17] 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.

[18] 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.

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

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

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

[22] 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.

[23] 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.

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

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