PAPER

Detecting Distributed Denial-of-Service Attacks by analyzing TCP SYN packets statistically

Yuichi OHSITA[†a)], Shingo ATA[††b)], *Members*, *and* Masayuki MURATA[†††c)], *Fellow*

SUMMARY Distributed denial-of-service attacks on public servers have recently become more serious. More are SYN Flood attacks, since the malicious attackers can easily exploit the TCP specification to generate traffic making public servers unavailable. To assure that network services will not be interrupted, we need faster and more accurate defense mechanisms against malicious traffic, especially SYN Floods. One of the problems in detecting SYN Flood traffic is that server nodes or firewalls cannot distinguish the SYN packets of normal TCP connections from those of SYN Flood attack. Moreover, since the rate of normal network traffic may vary, we cannot use an explicit threshold of SYN arrival rates to detect SYN Flood traffic. In this paper we introduce a mechanism for detecting SYN Flood traffic more accurately by taking into consideration the time variation of arrival traffic. We first investigate the statistics of the arrival rates of both normal TCP SYN packets and SYN Flood attack packets. We then describe our new detection mechanism based on the statistics of SYN arrival rates. Our analytical results show that the arrival rate of normal TCP SYN packets can be modeled by a normal distribution and that our proposed mechanism can detect SYN Flood traffic quickly and accurately regardless of time variance of the traffic.

key words: *Distributed Denial of Service (DDoS), SYN Flood, Statistical Analysis, Gamma Distribution, Traffic Monitoring*

1. Introduction

The recent rapid growth and increasing utility of the Internet are making Internet security issues increasingly important. Denial-of-service (DoS) attacks are one of the most serious problems and must be resolved as soon as possible. These attacks prevent users from communicating with service providers and have damaged many major web sites all over the world.

The number of attacks is increasing, and the techniques used to attack servers are more complex. In the distributed denial-of-service (DDoS) attack often seen recently, multiple distributed nodes attack a single server concurrently. A malicious user tries to hack remote nodes by exploiting the vulnerabilities of software running on them, installs an attacking program on hijacked nodes, and keeps them waiting for an order to attack a victim server. When the malicious user sends a signal to them, they begin to attack to the same server. Even if the rate of attack for each node is small, the attack traffic can cause serious damage at the victim server when the number of hijacked nodes is large.

[†]Graduate School of Economics, Osaka University
[†††]Graduate School of Information Science and Technology, Osaka University
[††]Graduate School of Engeneering, Osaka City University
a) E-mail: y-ohsita@econ.osaka-u.ac.jp
b) E-mail: ata@info.eng.osaka-cu.ac.jp
c) E-mail: murata@ist.osaka-u.ac.jp

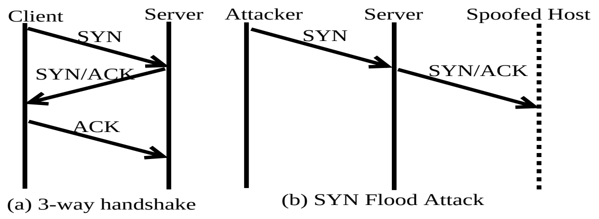


Fig. 1 Overview of a 3-way handshake and a SYN Flood attack

There are many kinds of DDoS attacks such as Smurf attacks [1], UDP Floods [2], and SYN Flood attacks [3]. In Smurf and UDP attacks, attackers generate many ICMP or UDP packets to exhaust the capacity of the victim's network link. In SYN Flood attacks, attackers send so many connection requests to one victim server that users cannot connect to that server. Because attackers can easily put servers into a denial-of-service state this way, about 90% of all DoS attacks are SYN Flood attacks [4].

SYN Flood attacks exploit the TCP (Transmission Control Protocol) specification. In the TCP, a client node communicates with a remote node (i.e., server) by way of a virtual connection established by a process called a 3-way handshake. As shown in Figure 1(a), a client first sends a server a SYN requesting to establish a connection. Then the server sends the client a SYN/ACK packet acknowledging receipt of the SYN packet. When the client receives the SYN/ACK packet, the client sends the server an ACK packet acknowledging receipt of the SYN/ACK packet and begins to transfer data.

In the 3-way handshake, the state in the server waits for the ACK packet from the client is called the *half-open* state. The server in the half-open state prepares for communication with the client, for example, by allocating a buffer. Since a server in the half-open state is using some of its resources for the client, the number of half-open states should be limited. The number of connections it can maintain while it is in the half-open state is controlled in a backlog queue. SYN packets in excess of the number that can be held in the backlog queue are discarded, and the server sends

RST packets to notify clients whose SYN packets are discarded.

Figure 1(b) shows an overview of a SYN Flood attack. Attackers send SYN packets whose source address fields are spoofed. The server receiving these SYN packets sends the SYN/ACK packets to spoofed addresses. If the node having the spoofed address actually exists, it sends a RST packet for the SYN/ACK packet because it didn't send the SYN packet. If there is no host having the spoofed address, however, the SYN/ACK packet is discarded by the network and the server waits in vain for an ACK packet acknowledging it. For losses of SYN/ACK packets, the server has a timer in the backlog queue, and half-open states exceeding the timer are removed. When the backlog queue is filled with spoofed SYN packets, however, the server cannot accept SYN packets from users trying to connect to the server.

Because the packets used in SYN Flood attacks do not differ from normal TCP SYN packets except in the spoofing of the source addresses, it is difficult to distinguish them from normal TCP SYN packets at the victim server. This is why SYN Flood attacks are hard to detect. Many methods to defend servers from these attacks, however, have been proposed.

In the ingress filtering [5], the internal router is configured to block packets that have source addresses from outside the internal network. This method cannot, however, remove all attack packets because attack packets with addresses of internal network cannot be blocked.

SYN cache [6] and SYN cookie [7] are mechanisms in server nodes. In the SYN cache mechanism, the server node has a global hash table to keep *half-open* states of all applications, while in the original TCP these are stored in the backlog queue provided for each application. As a result, the node can have more number of *half-open* states and the impact of SYN Flood attack can be reduced. However, this mechanism does not resolve the problem of SYN Flood attacks fundamentally. On the other hand, the SYN cookie mechanism can remove the backlog queue by using a *cookie* approach. In the original TCP the server node first allocates the server's resources and sends SYN/ACK packet. It is because there is no method to validate whether the received ACK packet after sending the SYN/ACK packet is really the acknowledgment of the SYN/ACK packet (i.e., the final packet of 3-way handshake). The SYN cookie embeds a *magic number* encrypted by the header of the SYN packet (e.g., IP addresses, port numbers) into the sequence number of the SYN/ACK packet. The server node then verify the ACK packet of the SYN/ACK packet by decrypting the sequence number of the ACK packet. The server node then allocate the server's resource only when the ACK packet is valid. This mechanism can remove the backlog queue, however, the process of encryption may become another weakness against the high-rated SYN packets. Moreover because the SYN cookie mechanism does not have any states of the connection until the last ACK packet is validated, the SYN cookie cannot retransmit the SYN/ACK packet when it is lost.

If routers can handle the state of TCP handshakes at the servers, for example, they can detect SYN Flood attacks more easily. Some firewalls therefore mitigate the damage of attacks by sending a SYN/ACK packet on behalf of the server and letting SYN packets to the server through only when the router receives the ACK packet of the delegated SYN/ACK packet from the client [8]. The server thus does not need to hold many half-open states for spoofed SYN requests. The cost of this mechanism, however, is high because firewalls need to handle the states of TCP connections. Also, while this method can avoid the damage of momentary attacks, it is vulnerable to long-term attacks that which overwhelm the firewalls. In such case, routers must detect attacks as quickly as possible and setup a filter to remove attack packets or to limit the rates of attack traffic.

Several methods for detecting attacks have been proposed, and one is to detect the mismatch between bidirectional packets [9]. When a server is not under attack, packet arrival rates for both directions are almost the same or at least of the same order, because the TCP needs an ACK packet for each packet that is sent. If the packet arrival rate for one direction is much higher than that for the other direction, the traffic in the high-rate direction might include some attack packets. In this mechanism, however, the router cannot detect the attack until the server can reply SYN/ACK packets for spoofed SYN packets. MULTOPS [10] is one of similar versions which checks asymmetries of traffics for both directions with the granularity of subnets. Another method in [11] is to use the difference between the rates of SYN packets (i.e., the head of the connection) and FIN/RST packets (i.e., tail of the connection). If the rate of SYN packets is much higher than that of FIN or RST packets, the router recognizes that the attacking traffic is mixed into the current traffic. [12] detects attacks by the number of source addresses. If the number of source addresses increase rapidly, the current traffic might include attack packets.

These methods have several problems, however, one of which is that they cannot detect attacks until servers are seriously damaged or until most of the connections are closed. Another is that they may mistake high-rate normal traffic for attack traffic because they do not take into consideration the normal time-of-day variation of network traffic or they do but using non-parametric approach without knowing how the normal traffic varies. Non-parametric approach can detect attacks in case of any variance of normal traffic, but require long time. Attack traffic should be identified more accurately and faster by considering the variance of normal traffic.

In this paper, we propose a method that detects attacks more quickly and more accurately by taking the time-of-day variance of traffic into consideration. For this purpose, we first collect all packets passing through the gateway of our university, and analyze the statistical characteristics of both normal and attack traffics. In Seccion 2 we then explain how we gathered data on router traffic and investigated the characteristics of normal traffic statistically. We then describe a new detection algorithm based on the results of our statistical analysis. In Section 3, we describe the definition of the attack traffic used in this paper and we show through trace-driven simulations that our method can detect all of attack traffics we defined. In Section 4 we conclude by briefly summarizing the paper and mentioning some of the future work we intend to do.

2. Statistical analysis of traffic and attack detection method

In this section, we first describe how we gathered the data we used to model normal traffic and how we analyzed that data. We then describe the algorithm we use to detect the attack traffic.

2.1 Monitoring and classification of real traffic

We deployed a traffic monitor at the gateway of Osaka University. We used optical-splitters to split the 1000 Base-SX fiber-optic cables and recorded the headers of all of packets transferred on this link. That is, we monitored all the packets in both the inbound and outbound directions at Osaka University.

We use `tcpdump` [13] to read the headers of packets. Although `tcpdump` cannot guarantee to read headers of all packets at wire-speed, we confirmed that the headers of less than 0.01% of the packets were not recorded and these losses did not affect the results of our statistical analysis.

We first classified monitored packets into *flows*. We defined a series of packets which have the same (src IP, src port, dest IP, dest port, protocol) fields as a single *flow* and we classify these *flows* into the following five groups.

Group N Flows that completed the 3-way handshake and were closed normally by an FIN or RST packet at the end of connections.

Group Rs Flows terminated by a RST packet before a SYN/ACK packet was received from the destination host. These flows were terminated this way because the destination host was not available for the service specified in the SYN request.

Group Ra Flows terminated by a RST packet before an ACK packet for the SYN/ACK packet was received. These flows were terminated this way because the SYN/ACK packets were sent to a host that was not in the Internet.

Group Ts Flows containing only SYN packets. These flows are not terminated explicitly (i.e., by RST/FIN packets) but by the timeout of flows. There would be three reasons that flows could be classified into this group. One was that, the destination node did not respond the SYN packet because, for example, the destination node is temporally shut down due to e.g., maintenance. A second was that the source address of the SYN packet was spoofed and the destination sent the SYN/ACK packet to the spoofed address. The third was that all of the SYN/ACK packets were discarded by the network (e.g., because of due to network congestion).

Group Ta Flows containing only SYN and its SYN/ACK packets. Like Group Ts flows, these flows were terminated by the timeout of flows. In this case, however, it was because all the ACK packets were dropped.

Table 1 Classification of flows

Group	number of flows	percentage
N	18,147,469	85.1
Rs	622,976	2.9
Ra	75,432	0.3
Ts	2,435,228	11.4
Ta	2,009	0.0

To identify the traffic of normal flows, we focused on the Group N flows. Hereafter, we refer these flows as *normal traffic* and to Groups Rs, Rs, Ts and Ta flows as *incomplete traffic*.

2.2 Time-dependent variation of normal traffic and its statistical modeling

In the work shown in this paper, we used the traffic data for 5 days: from 17:55 on March 20, 2003 to 19:45 on March 24, 2003. The average rate of incoming traffic (from the Internet to the campus network) was about 12.0 Mbps and the average rate of outgoing traffic was about 22.4 Mbps. During busy hours (09:00 to 17:00) the average incoming and outgoing rates were respectively 37.0 Mbps and 55.0 Mbps. A total of 1,983,116,637 TCP packets were monitored, 21,615,220 of which were SYN packets. The total number of flows that were monitored, however, was only 21,283,114. The difference between the number of SYN packets and the number of flows is due to the retransmission of SYN packets.

The numbers of flows classified into each of the five groups are listed in Table 1. These values were obtained using 180 seconds as the timeout. That is, if there are more than 180 seconds after the last packet in of the flow, we considered the flow to be terminated.

The time-dependent variations of SYN arrival rates of all flows, the flows in *normal traffic* and the flows in *incomplete traffic* are shown in Figures 2. Points where the arrival rate rises sharply (e.g., 28,000 sec and 57,000 sec) seem to be due to *incomplete traffic*. These results also show that we would mistakenly identify many points as attacks if we set a single threshold for the SYN arrival rates because the arrival rates of the normal traffic change over time. We can also see that the distribution of SYN arrival rates seems to be different in *incomplete traffic* from in the *normal traffic* especially at the tail.

To confirm this impression, we fitted the SYN arrival rates of normal traffic to several distributions. We selected four distributions as candidates.

The equation for the normal distribution $F(x)$ with the mean ζ and the variance σ^2 of measured SYN arrival rates is

$$F(x) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}\sigma} exp[\frac{-(y-\zeta)^2}{2\sigma^2}]dy. \tag{1}$$

The lognormal distribution of which variable is the logarithmic variable of the normal. The equation for the log normal distribution is

$$F(x) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}\sigma y} exp[\frac{-(\log y-\zeta)^2}{2\sigma^2}]dy. \tag{2}$$

In lognormal distribution, two parameters (ζ,σ) are calcurate from

$$\hat{\zeta} = \frac{1}{n}\sum_{i=0}^{n} \log x_i \tag{3}$$

$$\hat{\sigma}^2 = \frac{1}{n}\sum_{i=0}^{n} (\log x_i - \hat{\zeta}). \tag{4}$$

where n is the number of samples.

The equation for the Pareto distribution is

$$F(x) = \ 1 - (\tfrac{x}{k})^{\alpha}, \quad x \geq k \tag{5}$$

Parameters (α,k) in Pareto distribution are obtaioned from [14].

$$\hat{k} = min(x_1, x_2, \ldots, x_n), \tag{6}$$

$$\hat{\alpha} = n\left[\sum_{i=1}^{n} \log \frac{x_i}{\hat{k}}\right]. \tag{7}$$

The equation for the gamma distribution is

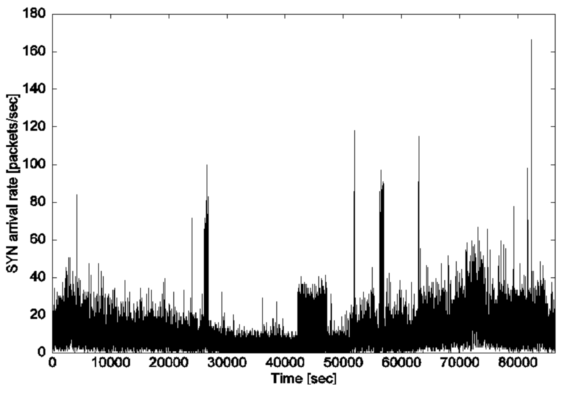
$$\Gamma(\lambda) = \int_{0}^{\infty} x^{\lambda-1}e^{-x}dx, \tag{8}$$

$$f(x) = \begin{cases} \frac{1}{\Gamma(\alpha)\beta^{\alpha}} x^{\alpha-1}e^{-\frac{x}{\beta}}, & 0 < x < \infty \\ 0, & -\infty < x < 0 \end{cases} \tag{9}$$

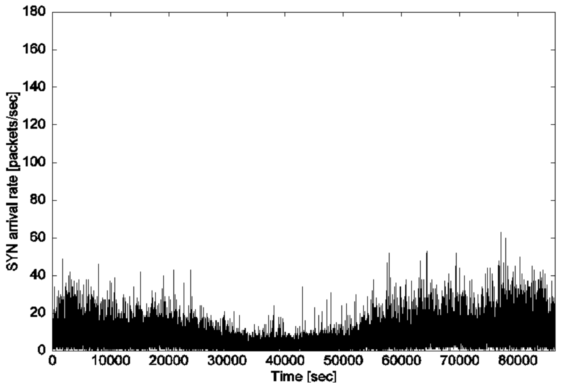
We calcuate parameters (α,β) in the gamma distribution so that it has the same average $E(X)$ and same variance $V(X)$ as the sample. The parameters are given by

$$\alpha = \frac{E(X)^2}{V(X)} \tag{10}$$

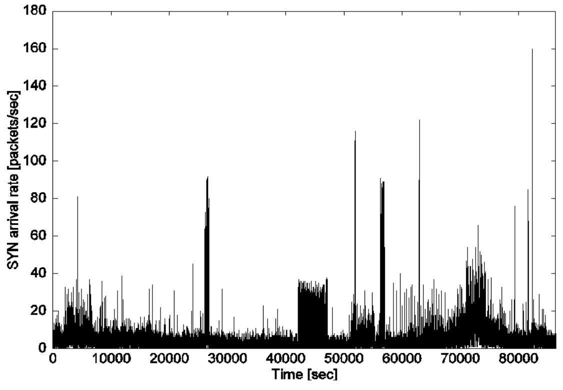
$$\beta = \frac{V(X)}{E(X)}. \tag{11}$$



(a) all flows



(b) normal traffic



(c) incomplete traffic

Fig. 2 Time-dependent variation of SYN arrival rates

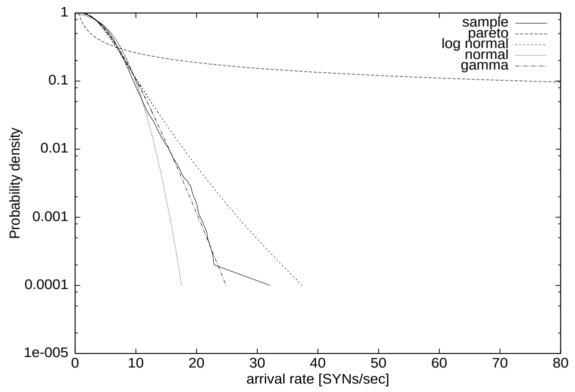


Fig. 3 Comparisons between the distributions of SYN rates and the four distributions(*normal traffic)*

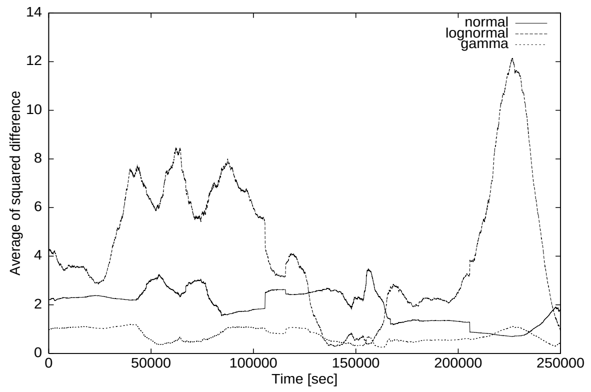


Fig. 4 Variation of average of squared differences between the sampled SYN rates and the three distributions

Figure 3 shows the result of fitting the normal traffic to four distributions. This figure compares the cumulative distribution of SYN packet arrival rates with the cumulative distributions descrived above. This curve is for the data obtained in 10-second intervals. We used 10,000 samples to obtain the SYN rate distributions. From this figure we can see that tail of the SYN rate distribution of the *normal traffics* is quite diffrent from Pareto distribution. Among rest three distributions, the gamma distribution is most suitable for the normal traffic in the region of 99-percentile and higher. On the other hand, the normal distribution is most appropriate in the area of less than 95-percentile. The lognormal distribution can also be fit to the normal traffic at 90-percentile and below.

To verify the appropriateness of the statistical modeling, we calcuate average of squared difference. In this experiment we especially focus on the tail part of the distribution of the normal traffic. We define $X_t (0 \leq X_t \leq 1)$ as the ratio of the tail part of the distribution. In other words, by setting $X_t = 0.9$ we obtain the region of the distribution at 90% and higher. Let us denote the number of samples of SYN rates as n. We sort sampled SYN rates in ascending order and label them $r_i (1 \leq i \leq n)$. $F^{-1}(x)$ is the inverse function of $F(x)$. Denote as D the average of squared differences from distributions $F(x)$.

$$D = \frac{\sum_{i=n-\lceil nX_t \rceil}^{n} (F^{-1}(\frac{i}{n}) - r_i)^2}{\lceil nX_t \rceil - 1}. \tag{12}$$

We calculated the value of D for each of our measurements of the SYN arrival rate (i.e., for every 10 seconds in our experiment). We used 10,000 samples to obtain the SYN rate distributions and the samples are obtained in 10-second intervals. That is, we need total 100,000 seconds to obtain the entire distribution. We then calculate the average of squared differences for each sample by using 10,000 histories of samples. Figure 4 shows the time-dependent variation of average of squared difference of normal traffic from normal, lognormal and gamma distributions. From this figure we can see that lognormal distribution is sometimes quite different from sample distribution. D on the gamma distribution is the smallest at any time, and its variation is also small. The variation of D on the normal distribution also does not vary regardless of time. From this observation, we can conclude that the gamma distribution is most appropriate to model the normal traffic statistically. The normal distribution is also useful for modeling, and the lognormal distribution gives a fair appropriateness.

We next evaluate for fitting statistical distributions with all traffic (i.e. the traffic including both normal and attack traffics). The results is shown in Figure 5. Figure 5 compares the distribution of SYN arrival rates of all flows three distributions used above. From this figure, we can observe a clear difference from the normal traffic case (Figure 5). Even in gamma and normal distributions the actual traffic is far from the modeling functions. It is because the attack traffic included in the all traffic gives a strong impact to the statistics, and clearly different from human-generated characteristics (e.g., constantly high rate for a long period). Especially, the influence of the attack traffic is significantly appeared at the tail part of the distribution. This is the reason why we focus on the tail part of the distribution for distinguish the attack traffic.

2.3 Attack detection method based on statistics of SYN arrival rates

As described in the previous subsection, the SYN arrival rates of the *normal traffic* can be modeled by gamma or normal distributions. Therefore it would be possible to detect the attack traffic by checking the difference between the sampled SYN rates and the modeled distribution functions at the tail part of the distribution. Since there is a clear difference between the attack traffic and the gamma/normal distribution function, we can identify the attack traffic by setting a kind of threshold about the difference. In this subsection we

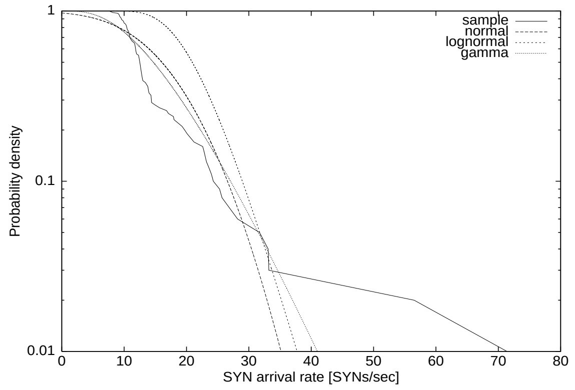


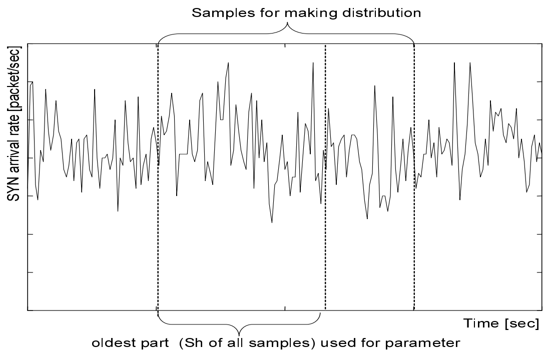
Fig. 5 Distribution of SYN packet arrival rate when attacks started.

propose a new detection method based on this motivation.

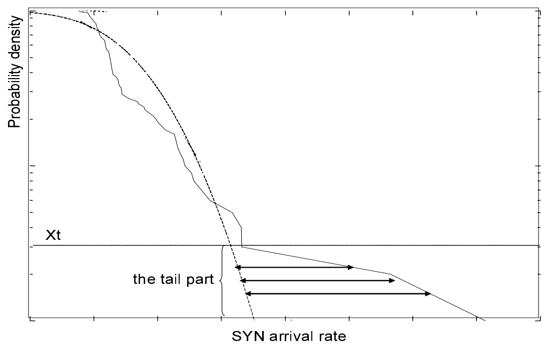
SYN arrival rates are calculated every time N SYN packets arrives. We measure the interval T between two sets of N SYN packet arrivals. We estimate the arrival rate from $\frac{N}{T}$. Note that this method is different from the SYN rate calculation described in the previous subsection. There are two reasons as following. First, in the heavy-loaded condition, we need to detect the attack more quickly. Hence the sampling interval should be variable according to the load of the network. Second, on the implementation issue this counter-based rate calculation is simpler than timer-based one because the interrupting timer is not necessary. We then collect M SYN rates, calculate the parameters of the modeled distribution function, and obtain the average squared difference between the sampled distribution and the modeled distribution function. Namely, total NM SYN packets are needed in order to obtain the distribution.

In calculating the average squared difference, we introduce two ratio values S_h and X_t, which are the ratio of the oldest part of samples and the tail part of the distribution, respectively. Figure 6 shows the outline of the average squared difference calculation. First, we calculate the parameter of the model function by using the S_h oldest part of sampled SYN rates. The reason why we use S_h is as follows. We calculate the value of D for each event of SYN rate calculation. The oldest one in M SYN rates are identified as the normal traffic in $M-1$ times. That is, if no attack traffic is detected previously, the older SYN rate has a tendency to be identified as normal traffic. We then calculate the squared difference D at the range of the X_t tail part of the distribution. In this paper, we set $X_t = 1 - S_h$ for simplicity.

Figure 7(a), Figure 8(a) and Figure 9(a) show the variation of the averages of squared differences for all flows and Figure 7(b), Figure 8(b) and Figure 9(b) show the ones for *normal traffic*. According to these results, the averages of the squared differences for the *normal traffic* are quite small and stable regardless of time. The


Samples for making distribution
SYN arrival rate [packet/sec]
Time [sec]
oldest part (Sh of all samples) used for parameter

(a) oldest part of samples



(b) tail of distribution

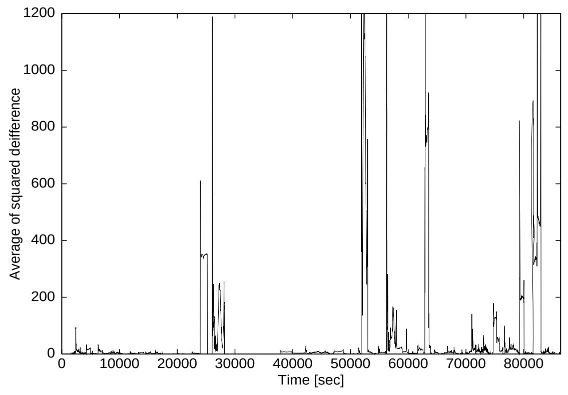
Fig. 6 Outline of the average squared difference calculation

averages of the squared differences for all flows, on the other hand, rise rapidly at several points (we call them *spikes* throughout this paper). Comparing Figures 7(a) with Figures 7(b) and Figures 8(a) with Figures 8(b) suggest that these *spikes* are caused by the *incomplete traffic* including attack traffic. Therefore, we can detect attacks by setting a threshold for the average of squared difference as the boundary between normal traffic and attack traffic.

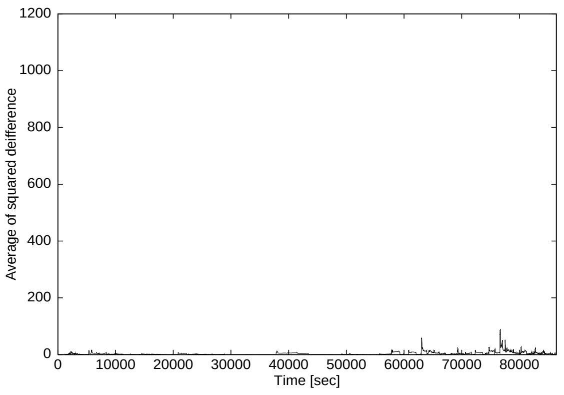
3. Performance evaluation

3.1 Definition of attack traffic

Prior to the performance evaluation, we define the attack traffic that must be detected as traffic that can put a server into a denial-of-service state. This state occurs when the backlog queue is full and new SYN packets arrive at the server. The length of the backlog queue is configured by a kernel parameter in the operating system, and the default parameters of the backlog queue on some major operating systems are listed in Table 2. The timeout values in this table are the durations until the half-open connections in the backlog queue are

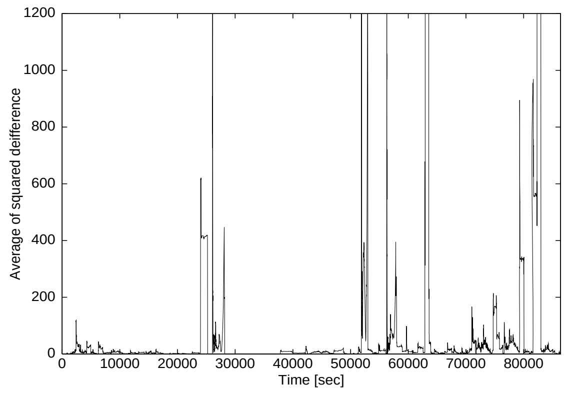


(a) all flows

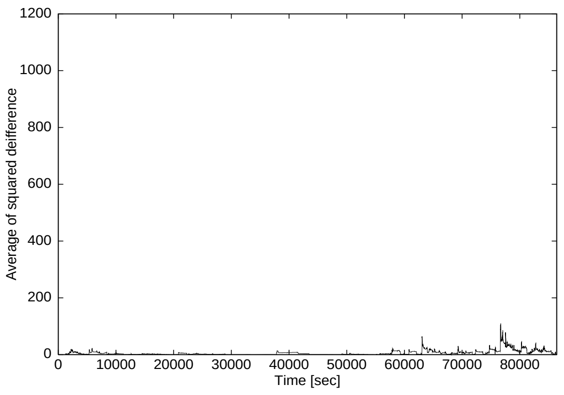


(b) normal traffic

Fig. 7 Variation of average of squared differences between the sampled SYN rates and the gamma distribution



(a) all flows



(b) normal traffic

Fig. 8 Variation of average of squared differences between the sampled SYN rates and the normal distribution

Table 2 default configuration of backlog queue

OS	max length	timeout (sec)
Linux	1,024	180
Solaris	1,024	240
Windows 2000 server	200	40

removed. That is, the half-open connections exceeding the timeout are closed by the server. To put a server into a denial-of-service state, attackers have to supply a number of SYN packets exceeding the maximum length of backlog queue within the timeout period. Supposing that target servers are running on Linux and we define attacks as cases when more than 1024 SYN packets having incompleted the 3-way handshake are sent within 180 seconds. Scanning our 5-day data, we found total 10 points satisfying this definition of attack traffic.

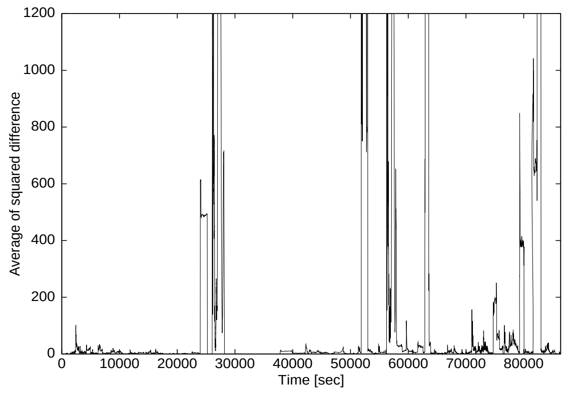
3.2 Accuracy of proposed detection method

We evaluated our detection algorithm by using a trace-driven simulation based on the traffic data we measured. We define the probability (P_-) of not detecting the attack traffic (i.e., the probability of the false-negative errors) and the probability (P_+) of erroneously detecting an attack (i.e., the probability of false-positive errors), which are calculated from following:

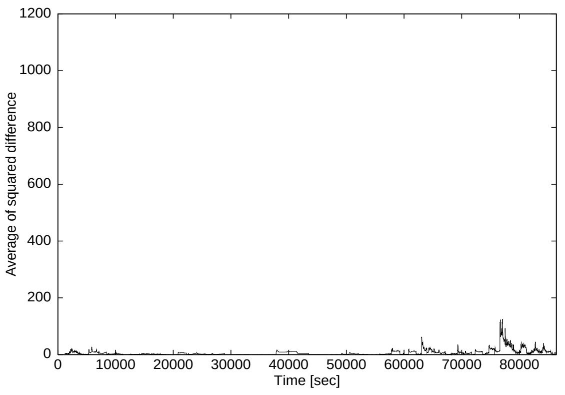
$$P_- = \frac{\sharp \text{ of attacks not detected}}{\sharp \text{ of attacks satisfying the definition}} \qquad (13)$$

$$P_+ = \frac{\sharp \text{ of points erroneously detected as attacks}}{\sharp \text{ of points detected as attacks}}$$

Probabilities of P_- and P_+ are shown in Figure 10 respectively as a function of the threshold for the average of squared difference. In this regard, we set N to 100, S_h to 90 and M to 100. These figures show that both distribution could detect all attacks when we set the threshold to less than 250. Though probability of detecting erroneously was 5 % when the threshold was 250, these erroneous detections were caused by a single client sending about 20 SYNs/sec. From the viewpoints of fairness and resource managements, this relatively



(a) all flows



(b) normal traffic

Fig. 9 Variation of average of squared differences between the sampled SYN rates and the lognormal distribution

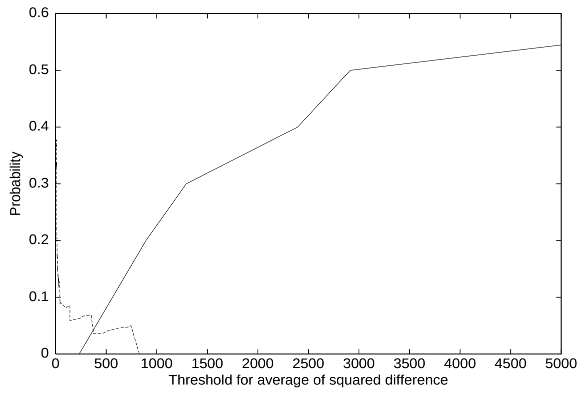
high-rate traffic should be limited. It can, after all, be regarded such traffic as "attack traffic" directed at the Internet itself rather than a specific server.

3.3 Detectable SYN rate of attack traffic

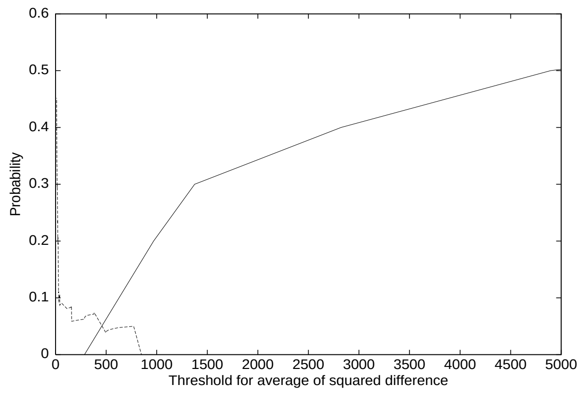
We also examine the SYN rates of attacks that can be detected without erroneous detections. Because low-rate attack traffic was not found in our data, we simulated such traffic by injecting low-rate attack traffic into the traced traffic.

3.4 Effect of parameters in our detection method

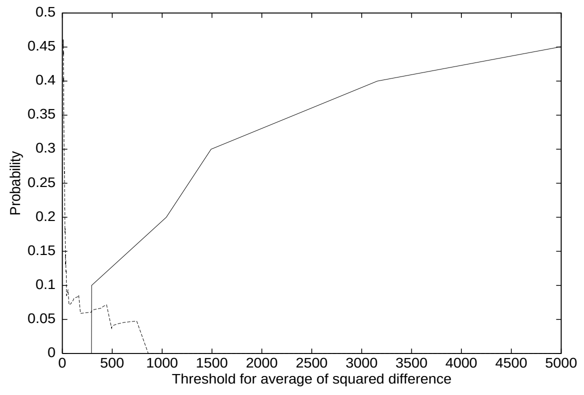
Figure 11 shows the SYN rates of attacks can be detected as a function of parameter S_h. We can detect lower-rate attacks by setting S_h to 75 than to 70. That is because when S_h is smaller, the number of samples used to estimate the parameters is smaller and we cannot model accurately. On the other hand, we can detect lower-rate attacks by setting S_h to 85 than by to 90.



(a) gamma distribution



(b) normal distribution



(c) lognormal distribution

Fig. 10 Relation between threshold for average of the squared difference and the probabilities of not detecting an attack (——) and of erroneously detecting an attack(- - -).

Too small X_t makes detection too sensitive because the number of samples compared with the models is small.

Figure 12 shows the SYN rates of attacks can be detected as a function of parameter N. In this regard,

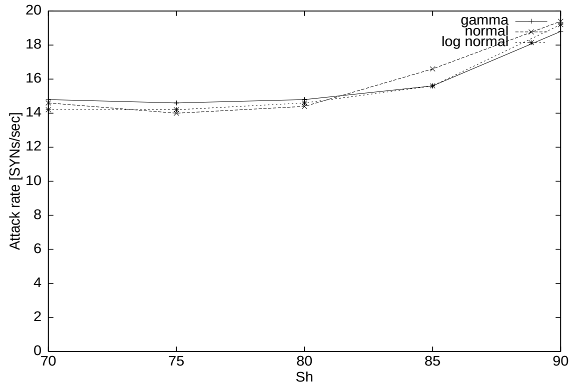


Fig. 11 Relation between the detectable SYN rate of attack traffic and parameter S_h.

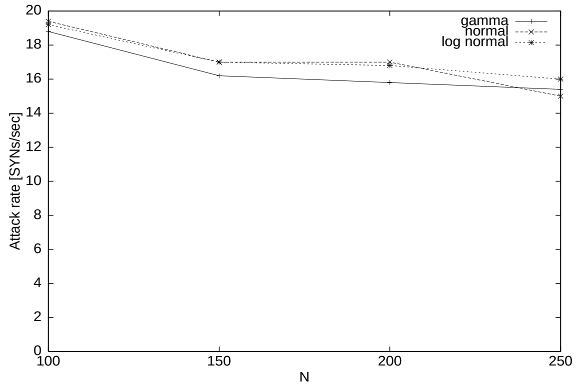


Fig. 12 Relation between the detectable SYN rate of attack traffic and parameter N.

we set X to 90 and M to 100. When we set too small N, momentary high rates are detected erroneously. On the other hand, larger N makes attack detection more dull and it takes more time to detect attacks.

Figure 13 shows the SYN rates of attacks can be detected as a function of parameter M. In this regard, we set S_h to 90 and N to 100. When we set M to larger value, we can model more accurately. However, we can detect lower-rate attacks by setting M to 200 than by to 250. That is because large M makes effect of attack traffic on the distribution of SYN arrival rates small and makes low-rate attacks difficult to be detected.

These results show also our method can detect smaller attacks than a single threshold cannot detect. Figure 2(b) shows the SYN arrival rates vary between 10 and 50 SYNs/sec. Therefore, to avoid erroneous detection, we should set a single threshold of SYN arrival rate more than 50 SYNs/sec though the threshold cannot detect low-rate attacks which occur in hours when the traffic is relatively low. Time-of-day variation of SYN rates influences methods using a single threshold. On the other hand, our method can detect attacks regardless of time-of-day variation of SYN rates. Therefore Figures 11 through 13, we can see that our

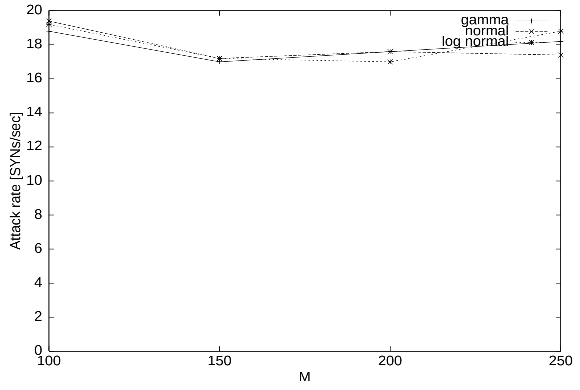


Fig. 13 Relation between the detectable SYN rate of attack traffic and parameter M.

method can detect attacks whose rates are lower than 20 SYNs/sec.

3.5 Comparison among three distribution functions

Figures 11 through 13 also show that there is no significant observation among three distribution functions (normal, lognormal, and gamma). So we can use any of these functions to detect the attack traffic in case of our simulation. But if we focus on the deployment of our detection mechanism, the calculation complexity is also important. It is clear that the calculation of the lognormal distribution is more complex than the one of the normal distribution. Both the normal and the gamma distributions requires much computational overhead, however, the calculation of parameters in the normal distribution is very easy. Also, the calculation of the normal distribution function can be simplified by using a table of standard normal distribution. In summary, the normal distribution is most appropriate to detect the attack traffic on considering both accuracy and implementation issues.

3.6 Time needed to detect the attack traffic

Figure 14 shows the dynamics of average of squared difference from the beginning of the attacks. In this figure, the SYN rates of the attacks are 20 SYNs/sec, 24 SYNs/sec and 28 SYNs/sec. In this figure N is 200, M is 100 and S_h is 90. We use the normal distribution as the model distribution. This figure shows that the averages of squared differences increase gradually after the beginning of attacks. When the threshold is set to 20, which detect attacks without detecting any attacks erroneously, attacks with SYN rates higher than 28 SYNs/sec can be detected within 20 seconds. In this case, the number of half-open states caused by attack is 560, which is smaller than the length of backlog queue in Linux.

To show that our mechanism can detect attacks faster, we compare the time needed to detect attacks

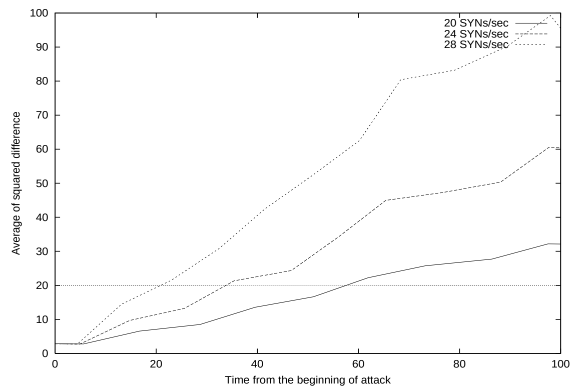


Fig. 14 Average of squared differences versus time after the beginning of attacks with various SYN rates.

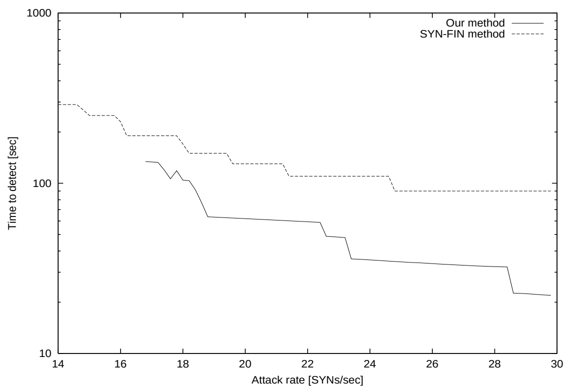


Fig. 15 Time to detect attacks with our method (—-) and with SYN-FIN method (- - -)

on our method with the time on the method proposed in [11]. Throughout this paper, we refer it as SYN-FIN method.

We first note here a brief description of the SYN-FIN method. First, we calculate Δ_i which is the difference between the number of SYN or SYN/ACK packets and the number of RST or FIN packets. We then obtain the normalized value of Δ_i by dividing the average number of RST or FIN packets F, which is given by $x_i = \Delta_i/F$. We then calculate y_i from

$$y_i = \begin{cases} 0 & (y_{i-1} + x_{i-1} - \alpha \leq 0) \\ y_{i-1} + x_{i-1} - \alpha & (otherwise) \end{cases} \quad (14)$$

Finally, we determine the traffic has some attacks by detecting the value of y_i exceeds the threshold T.

In the simulation, we set the values of α and T to be 0.15 and 0.37 respectively, which are the optimized parameters to detect attacks as fast as possible. In this simulation we used normal distribution as the model and set N to 200, M to 100 and S_h to 90. We set the threshold of D in our method to be 20, which can detect attacks without detecting any attacks erroneously.

Figure 15 compares the time to detect attacks between our method and SYN-FIN method. We varied the rate of attacking traffic and measure the time needed to detect the attacking traffic. From this figure, we can observe that our method is much faster to detect attacks than SYN-FIN method. One of the reasons is because SYN-FIN method uses a non-parametric approach to estimate the difference the characteristic of normal from the one of attacking traffics, while our method adopts a parametric approach (i.e., we model that the SYN rate of the normal traffic follows the normal distribution) to estimate it. The parametric approach can detect faster and more accurate than the non-parametric approach in the cases if the model is appropriate. However, SYN-FIN method has an advantage that it can also detect attacks with lower rate (e.g., less than 14 SYNs/sec). Our method cannot detect them because the traffic having the low rate attacks still follows the normal distribution.

3.7 Resource needed by detection method

From above results, our method can work with only 100 samples of SYN rates. If we monitor D for each destination address, we need 100 samples for each address. The captured traffic has 1,000 destination addresses in 1,000 seconds of inbound traffic, and 10,000 destination addresses in 1,000 seconds of outbound traffic. According to Figure 2(b), arrival rates are not so large and we can then assume a small range of integer value (i.e., 16 bits) is enough for counting SYN rates. Then we need 200 KByte for incoming traffic and 2 Mbyte for outgoing traffic.

4. Conclusion and future work

We analyzed the traffic at an Internet gateway and the results showed that we can model the arrival rates of normal TCP SYN packets as a normal distribution. Using this result, we described a new attack detection method taking the time variance of arrival traffic into consideration. Simulation results show that our method can detect attacks quickly and accurately regardless of the time variance of the traffic. Our future works are to set the threshold automatically and to optimizesome configurable parameters (e.g., sampling intervals, number of samples to obtain the distribution).

References

[1] "CERT advisory CA-1998-01 smurf IP Denial-of-Service attacks." available at `http://www.cert.org/advisories/CA-1998-01.html`, January 1998.

[2] "CERT advisory CA-1996-01 UDP port Denial-of-Service attack." available at `http://www.cert.org/advisories/CA-1996-01.html`.

[3] "CERT advisory CA-1996-21 TCP SYN flooding and IP spoofing attacks." available at `http://www.cert.org/advisories/CA-1996-21.html`, September 1996.

[4] D. Moore, G.M. Voelker, and S. Savage, "Inferring internet Denial-of-Service activity," Proceedings of the 2001 USENIX Security Symposium, pp.9–22, August 2001.

[5] P. Ferguson and D. Senie, "Network ingress filtering: Defeating denial of service attacks which employ IP source address spoofing." RFC 2267, January 1998.

[6] J. Lemon, "Resisting SYN flooding DoS attacks with a SYN cache," Proceedings of USENIX BSDCon'2002, pp.89–98, February 2002.

[7] A. Zuquete, "Improving the functionality of SYN cookies," Proceedings of 6th IFIP Communications and Multimedia Security Conference, pp.57–77, September 2002.

[8] T. Darmohray and R. Oliver, "Hot spares for DoS attacks," The Magazine of USENIX and SAGE, vol.25, no.4, p.3, July 2000.

[9] J. Mirkovic, D-WARD: DDoS network attack recognition and defence, Ph.D. thesis, Computer Science Department, University of California, Los Angels, June 2003.

[10] T.M. Gil and M. Poletto, "MULTOPS: A data-structure for bandwidth attack detecrion," Proceedings of USENIX Security Symposium' 2001,, pp.23–38, August 2001.

[11] H. Wang, D. Zhang, and K.G. Shin, "Detecting SYN flooding attacks," Proceedings of IEEE INFOCOM 2002, pp.1530–1539, June 2002.

[12] T. Peng, C. Leckie, and K. Ramamohanarao, "Detecting distributed denial of service attacks using source IP address monitoring." available at `http://www.ee.mu.oz.au/pgrad/taop/research/detection.pdf`, November 2002.

[13] "Tcpdump public repository." available at `http://www.tcpdump.org/`.

[14] V. Brazauskas and R. Serfling, "Robust and efficient estimation of the tail index of a one-parameter pareto distribution," North American Actuarial Journal, pp.12–27, April 2000.

Yuichi Ohsita received the M.E. degree in Information and Computer Science from Osaka University, Japan, in 2005. He is now a Research Associate at the Graduate School of Economics, Osaka University. His research interests include countermeasure against DDoS attacks. He is a member of IEICE and IEEE.

Shingo Ata received M.E. and Ph.D. degrees in Informatics and Mathematical Science from Osaka University in 1998 and 2000, respectively. He is an Associate Professor in Information and Communication Engineering at Osaka City University, Japan. His research works include networking architecture, design of communication protocols, and performance modeling on communication networks. He is a member of IEICEJ and ACM.

Masayuki Murata received the M.E. and D.E. degrees in Information and Computer Sciences from Osaka University, Japan, in 1984 and 1988, respectively. In April 1984, he joined Tokyo Research Laboratory, IBM Japan, as a Researcher. From September 1987 to January 1989, he was an Assistant Professor with Computation Center, Osaka University. In February 1989, he moved to the Department of Information and Computer Sciences, Faculty of Engineering Science, Osaka University. From 1992 to 1999, he was an Associate Professor in the Graduate School of Engineering Science, Osaka University, and from April 1999, he has been a Professor of Osaka University. He moved to Graduate School of Information Science and Technology, Osaka University in April 2004. He has more than three hundred papers of international and domestic journals and conferences. His research interests include computer communication networks, performance modeling and evaluation. He is a member of IEEE, ACM, The Internet Society, IEICE and IPSJ.