Master's Thesis

Title

Detection and Defense Method against Distributed SYN Flood Attacks

Supervisor

Professor Masayuki Murata

Author

Yuichi Ohsita

February 15th, 2005

Department of Information Networking

Graduate School of Information Science and Technology

Osaka University

Master's Thesis

Detection and Defense Method against Distributed SYN Flood Attacks

Yuichi Ohsita

Abstract

Distributed denial-of-service attacks on public servers have recently become a serious problem. To assure that network services will not be interrupted, we need faster and more effective defense mechanisms to protect against malicious traffic, especially SYN floods. One problem in detecting SYN flood traffic is that server nodes or firewalls cannot distinguish the SYN packets of normal TCP connections from those of a SYN flood attack. Another problem is single-point deffenses (e.g. firewalls) lack the scalability needed to handle an increase in the attack traffic. In this research, we have developed a new mechanism to detect and block SYN flood attacks.

First, we introduce a mechanism for detecting SYN flood traffic more accurately by taking into consideration the time variation of arrival traffic. We investigate the statistics regarding the arrival rates of both normal TCP SYN packets and SYN flood attack packets. We then describe a new detection mechanism based on these statistics. 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.

As a post-detection method, we develop a distributed defense mechanism which uses overlay networks. In this mechanism, alert messages are sent via the overlay networks. Defense nodes which receive the alert messages can identify legitimate traffic and block malicious traffic by delegating SYN/ACK packets. Legitimate traffic is protected via the overlay networks.

Our simulation results show that attacker-side defense can protect legitimate packets from higher-rate attacks than victim-side defense and that our proposed method can effectively block malicious traffic and protect legitimate traffic.

Keywords

Distributed Denial of Service (DDoS)

Statistical Analysis

Traffic Monitoring

SYN flood

Overlay Network

TCP Proxy

Contents

List of Figures

List of Tables

1 Introduction

The rapid growth and increasing utility of the Internet have made Internet security issues increasingly important. Denial-of-service (DoS) attacks are one of the most serious problems and a means of preventing such attacks must be devised 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 becoming more complex. In the distributed denial-of-service (DDoS) attacks often seen recently, multiple distributed nodes concurrently attack a single server. 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 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.

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 DDoS attacks are SYN flood attacks [4].

SYN flood attacks exploit the transmission control protocol (TCP) specification. In the TCP, a client node communicates with a remote node (i.e., a server) by way of a virtual connection established through a process called a 3-way handshake. As shown in Figure 1(a), a client first sends a server a SYN packet requesting establishment of a connection. The server then 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 waiting for the ACK packet from the client is called the *half-open* state. A server in the half-open state prepares for communication with the client by, for example, 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

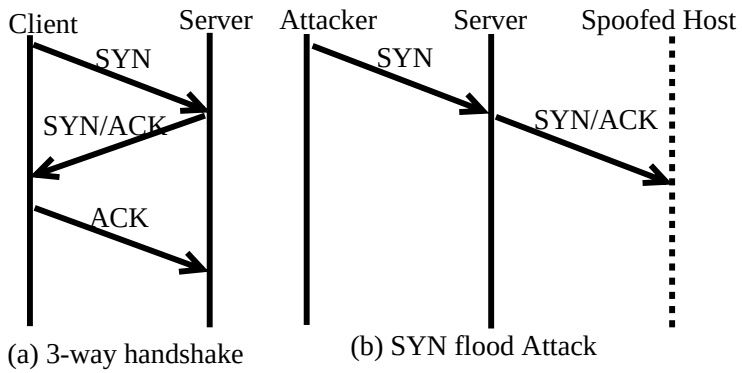


Figure 1: Overview of a three-way handshake and a SYN Flood attack

connections a server 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 the spoofed addresses. If a node having one of the spoofed addresses actually exists, it sends a RST packet for the SYN/ACK packet because it did not 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 block. Many methods to defend servers from these attacks, however, have been proposed.

SYN cache [5] and SYN cookie [6] are mechanisms applied 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 a larger number of *half-open* states and the impact of a SYN flood attack can be reduced. 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 a SYN/ACK packet. This is done because there is no way to validate whether the ACK packet received after sending the SYN/ACK packet is really the acknowledgment of the SYN/ACK packet (i.e., the final packet of the 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 verifies the ACK packet of the SYN/ACK packet by decrypting the sequence number of the ACK packet. The server node then allocates the server resources only when the ACK packet is valid. This mechanism can remove the backlog queue.

However, these single-point defense mechanisms have a fundamental problem with respect to scalability. In DDoS attacks, attack nodes are widely distributed all over the world. Attack traffic from attack nodes is aggregated into a very high rate attack at the server. At this point, a DDoS attack is highly scalabile because the amount of attack traffic can increase in proportion to the number of attacker nodes. On the other hand, single-point defense mechanisms lack scalability commensurate with the attack traffic increase. That is, high-rate attacks from widely distributed nodes can overwhelm the firewalls or the servers regardless of the implemented server-side defense mechanism (e.g. SYN Cache or SYN Cookie) is implemented. For this reason, a distributed defense mechanism is needed to defend servers from distributed attacks.

Implementing a distributed defense mechanism, such as a cooperation of distributed nodes, is more difficult than a single-point approach. D-WARD [7] has been proposed as a way to stop DDoS attacks near their source. In this method, an edge node detects attacks and limits the rate of traffic addressed to the victim server. However, detecting distributed attack traffic near attacker nodes is quite difficult when attack nodes are highly distributed and each attacker node generates a small amount of attack traffic. We believe that it is more practical to detect attacks near a victim node and alert other nodes deployed near attacker nodes. In pushback [8], a router detecting an attack requests upstream routers to limit the amount of traffic bounded to the victim node. This method can set a rate limit near attackers by recursively requesting the limitation from upstream routers. This is an effective countermeasure to attacks exhausting network links, but a rate limit is not an effective way to prevent attacks, such as SYN flood attacks, which exhausting the resources of servers. DefCOM [9] has been proposed as a framework that allows DDoS defense nodes to communicate with each other; however, the framework was reported without any description of

a specific method to detect or block attack traffic. In PacketScore [10], edge nodes compute a per-packet score which estimates the legitimacy of a packet. Core nodes perform score-based selective packet discarding. This method can effectively mitigate attacks when the characteristics of attack traffic differ from those of legitimate traffic. However, legitimate traffic may be mistakenly identified as attacks and blocked by this method. This can seriously impair the communication between the victim and legitimate clients.

Constant identification of all packets is costly. To defend servers from attacks effectively, we also need a mechanism for detecting attacks as quickly as possible. Several methods for detecting attacks have been proposed, and one is to detect the mismatch between bidirectional packets [7]. 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 attack packets. In this mechanism, however, the router cannot detect an attack until the server replies with SYN/ACK packets for spoofed SYN packets. MULTOPS [11] is a similar version which checks for traffic asymmetry in both directions using the granularity of subnets. Another method [12] is to use the difference between the rates of SYN packets (i.e., the head of the connection) and FIN/RST packets (i.e., the tail of the connection). If the rate of SYN packets is much higher than that of FIN or RST packets, the router recognizes that attacking traffic is mixed in with the current traffic. Attacks can also be detected through the number of source addresses [13]. If the number of source addresses increases 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 so using a non-parametric approach without knowing how normal traffic varies. A non-parametric approach can detect attacks if there is any variance from normal traffic, but require a long time. Attack traffic should be identified more accurately and quickly by considering the variance of normal traffic.

Given the above state of affairs, we clearly need a defense mechanism that (1) has enough scalability accommodate any the increase in the distributed attack traffic, (2) can detect attacks accurately on the victim sides, and (3) can correctly protect legitimate traffic. For the first issue,

blocking attacks at distributed points has higher scalability than defense at a single node. We can reliably identify legitimate packets by receiving the ACK packets corresponding to SYN/ACK packets. We use a proxy approach which responds to the acknowledgements of SYN packets on behalf of the victim node, and passes SYN packets only when the proxy receives the ACKs of SYN/ACK packets. For the second problem, we propose a method that detects attacks more quickly and more accurately by taking the time-of-day variance of traffic into consideration [21]. 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. Then, we propose a new detection algorithm based on the results of our statistical analysis. For the last problem We can prevent attakers from spoofing attack packets as legitimate packets by forwarding legitimate packets via overlay networks.

The practicality of deployment is also an important issue, because an effective defense against DDoS attacks is urgently needed. The solution must be easy to deploy and not negatively affect current IP frameworks.

In this thesis, we propose a new distributed defense system using overlay networks against distributed SYN flood attacks [15]. In this system, attacks are easily detected by victim-side nodes. After an attack is detected, alert messages are forwarded to all nodes via the overlay networks. The edge defense nodes which receive an alert begin to identify and block attack packets. At the same time, the defense nodes protect legitimate packets by forwarding them via the overlay networks. This method can be deployed in a phased manner. When only one autonomous system (AS) deploys this method, attacks are blocked at all edges of the AS. As more ASes deploy this method it becomes more efficient because attack traffic is blocked at more points. Finally, when all ASes deploy this method, there will be no attack traffic in core networks because it will be blocked at the edge nodes.

In Section 2 we give an overview of our defense mechanism. In Section 3 we explain our new detection method which takes the time-of-day variance of traffic into consideration. In Section 4 we explain in detail the operation after an attack is detected. In Section 5, we explain the deployment scenario for our mechanism. In Section 6, we show simulation results indicating that our method can effectively detect and block attack traffic while protecting legitimate traffic. In Section 7 we conclude this thesis.

2 Defense Mechanism Overview

Figure 2 shows an overview of our proposed architecture. We place *defense nodes* at the edge of a network (we call this network a *protected network*). Each defense node logically connects to one or more other defense nodes, and constructs an overlay network among the defense nodes. To identify legitimate SYN packets, defense nodes act as a SYN proxy which returns a SYN/ACK packet instead of the victim node doing so. The SYN packet is relayed only when the defense node receives the ACK packet of the SYN/ACK packet from the client (Figure 3). Once a flow (i.e., packets having the same (src IP, dest IP, src port, dest port, protocol) fields) is identified as legitimate traffic, packets of the flow are transferred via the overlay network and protected from attack traffic.

In the ideal situation, the defense node should handle all arriving packets and pick up legitimate packets from among them. However, this process causes processing overhead, and the defense node will become a performance bottleneck. To minimize the defense node overhead, it is desirable to identify only those packets going to the victim node. For this purpose, we use a mechanism for detecting a SYN flood attack. In the *attack detection mode*, the defense node monitors packets outbound from the protected network. If the defense node detects attack traffic, the defense node alerts other defense nodes of the address of the victim node. Upon receiving the alert, each defense node moves into the *defense mode* for the victim's address. In the defense mode, the defense node delegates SYN/ACK packets to identify legitimate traffic. The defense mode is continued until the attack ends.

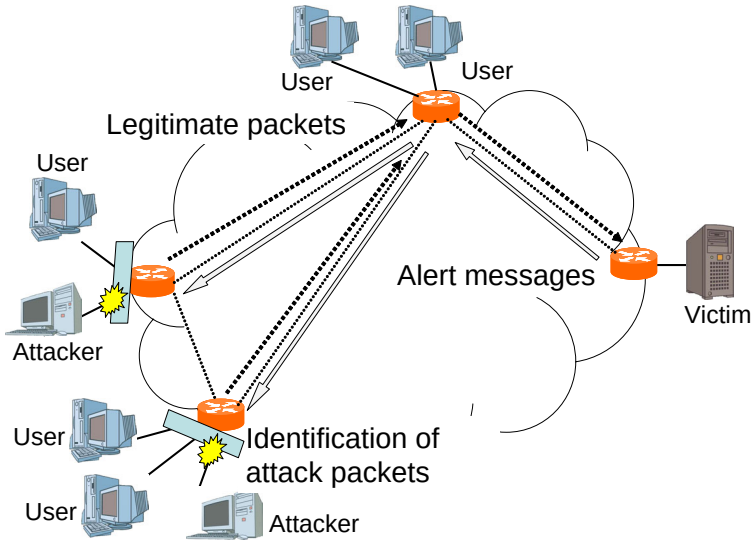


Figure 2: Distributed defense using overlay networks

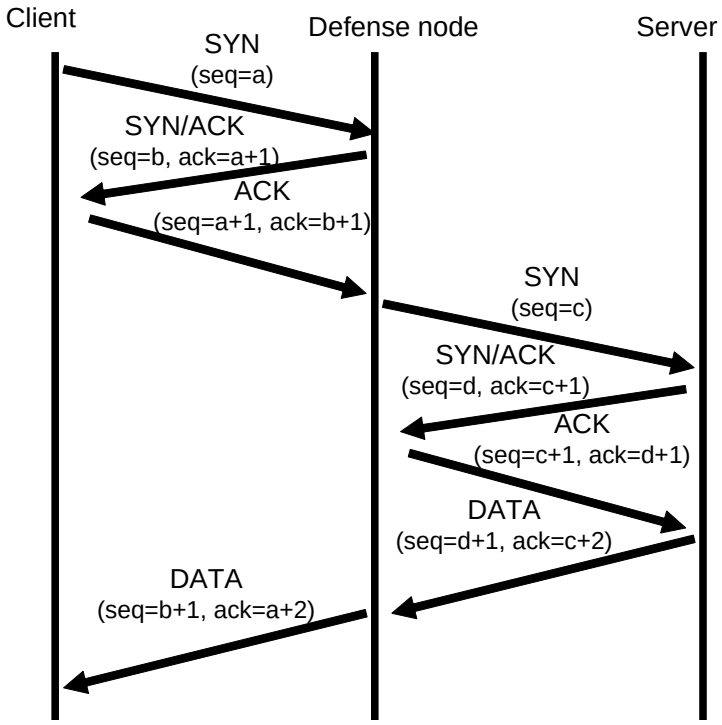


Figure 3: Delegation of SYN/ACK packets

3 Attack Detection Mode

In this section, we describe the detection method used in the attack detection mode in our mechanism. This mechanism can detect attacks accurately and quickly by taking the time-of-day variance of traffic into consideration.

First, we describe how we gathered the data used to model normal traffic and how we analyzed that data. We then describe the algorithm used to detect attack traffic.

3.1 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` [16] 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. 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.

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*.

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

In the work shown in this thesis, 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 4. 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 differ for *incomplete traffic* compared to that for *normal traffic*, especially at the tail.

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

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

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

A variable has a lognormal distribution if the natural logarithm of the variable has a normal distribution. The equation for a lognormal 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 a lognormal distribution, two parameters (ζ,σ) are calculated 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 a 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 [17].

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

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

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 calculate 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

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

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

Figure 5 shows the result of fitting the normal traffic to the four distributions. This figure compares the cumulative distribution of SYN packet arrival rates with the cumulative distributions described above. This curve is for data obtained at 10-second intervals. We used 10,000 samples to obtain the SYN rate distributions. From this figure, we can see that the tail of the SYN rate distribution of the *normal traffic* is quite different from the Pareto distribution. Among the other three distributions, the gamma distribution is the best match for the normal traffic in the region of the 99th percentile and higher. On the other hand, the normal distribution is most appropriate in the area of less than the 95th percentile. The lognormal distribution can also be fit to the normal traffic at the 90th percentile and below.

To verify the appropriateness of the statistical modeling, we calculated the average of the squared difference. In this experiment, we focused especially 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 sorted the sampled SYN rates in ascending order and labeled them $r_i (1 \leq i \leq n)$. $F^{-1}(x)$ is the inverse function of $F(x)$. Denoted as D, the average of the squared differences from a distribution $F(x)$ is

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

We calculated the value of D for each of our measurements of the SYN arrival rate (i.e., every 10 seconds in our experiment). We used 10,000 samples to obtain the SYN rate distributions and

the samples were obtained at 10-second intervals. That is, we needed a total of 100,000 seconds to obtain the entire distribution. We then calculated the average of the squared differences for each sample by using 10,000 sample histories. Figure 6 shows the time-dependent variation of the average of the squared difference for normal traffic from the normal, lognormal, and gamma distributions. From this figure we can see that the lognormal distribution was sometimes quite different from the sample distribution. D on the gamma distribution is the smallest at almost any time, and its variation is also small. The variation of D on the normal distribution also does not vary greatly regardless of time. From this observation, we can conclude that the gamma distribution is the most appropriate to statistically model normal traffic. The normal distribution is also useful for modeling, and the lognormal distribution is fairly appropriate.

We next evaluated the fit of statistical distributions with all traffic (i.e., traffic including both normal and attack traffic). Figure 7 compares the distribution of the SYN arrival rates for all flows with the three distributions used above. We can see a clear difference from the normal traffic case (Figure 6). Even for the gamma and normal distributions the actual traffic was far from the modeling functions. This was because inclusion of the attack traffic strongly affected the statistics, and was clearly different from human-generated traffic (e.g., it had a constantly high rate for a long period). The influence of the attack traffic is especially noticeable at the tail part of the distribution. This is why we focus on the distribution tail for identifying attack traffic.

3.3 Attack detection method based on SYN arrival rate statistics

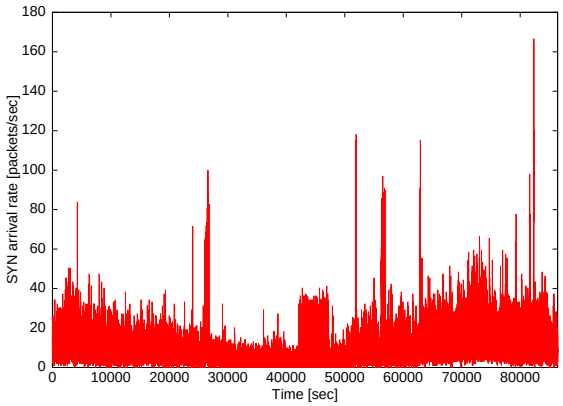
As described in the previous subsection, the SYN arrival rates of the *normal traffic* can be modeled by gamma or normal distributions. Therefore it would be possible to detect the attack traffic by checking the difference between the sampled SYN rates and the modeled distribution functions at the tail 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 propose a new detection method based on this motivation.

SYN arrival rates are calculated every time N SYN packets arrive, and the interval T between two sets of N SYN packet arrivals is measured. We estimate the arrival rate from $\frac{N}{T}$. This method differs from the SYN rate calculation described in the previous subsection for two reasons. First, under heavily loaded conditions, we need to detect an attack more quickly. Hence the

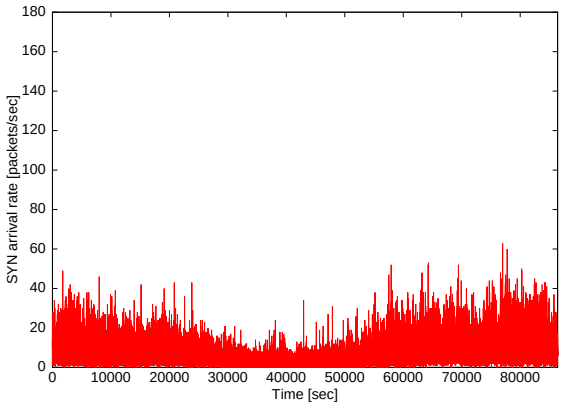
sampling interval should be variable according to the network load. Second, for implementation this counter-based rate calculation is simpler than a timer-based one because an interrupting timer is not needed. We then collect M SYN rates, calculate the parameters of the modeled distribution function, and obtain the average squared difference between the sampled distribution and the modeled distribution function. A total of NM SYN packets are needed to obtain the distribution.

To calculate the average squared difference, we introduce two ratio values X_h and X_t, which are the ratio of the head and the tail part of the distribution, respectively. Figure 8 shows the outline of the average squared difference calculation. First, we calculate the parameter of the model function by using the X_h oldest part of the sampled SYN rates. The reason for using X_h is as follows. We calculate the value of D for each event of the SYN rate calculation. The oldest of M SYN rates is identified as the normal traffic in $M - 1$ times. That is, if no attack traffic was detected previously, the oldest SYN rate tends to be identified as normal traffic. We then calculate the squared difference D in the range of the X_t tail part of the distribution. In this thesis, we set $X_t = 1 - X_h$ for simplicity.

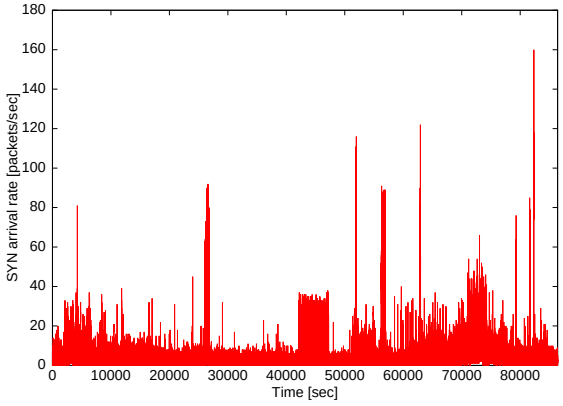
Figures 9(a), 10(a), and 11(a) show the variation of the average squared difference for all flows and Figures 9(b), 10(b), and 11(b) show the variation for *normal traffic*. These results show that the averages of the squared difference for *normal traffic* were quite small and stable regardless of time. The averages of the squared difference for all flows, on the other hand, rose rapidly at several points (we call these *spikes* throughout this thesis). A comparison of Figure 9(a) with Figure 9(b) and Figure 10(a) with Figure 10(b) suggests that these *spikes* were caused by the *incomplete traffic* including attack traffic. Therefore, we can detect attacks by setting a threshold for the average squared difference as a boundary between normal traffic and attack traffic.



(a) all flows



(b) normal traffic



(c) incomplete traffic

Figure 4: Time-dependent variation of SYN arrival rates

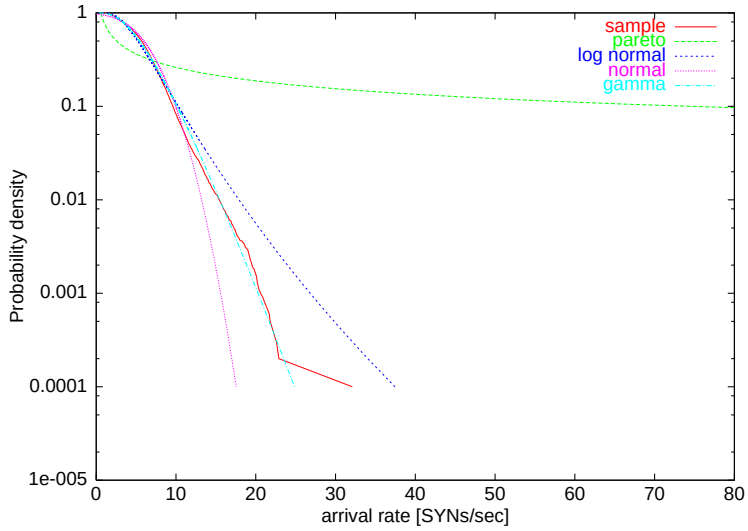


Figure 5: Comparisons between the distributions of SYN rates and the four distributions (*normal traffic)*

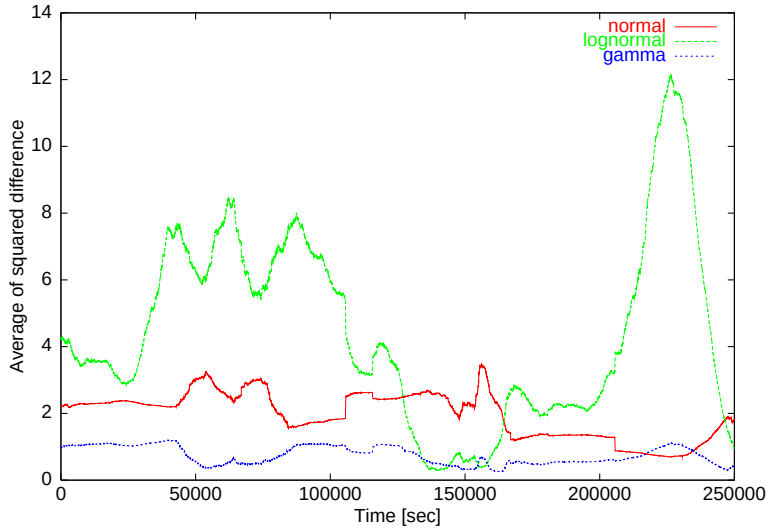


Figure 6: Variation of average of squared differences between the sampled SYN rates and the three distributions

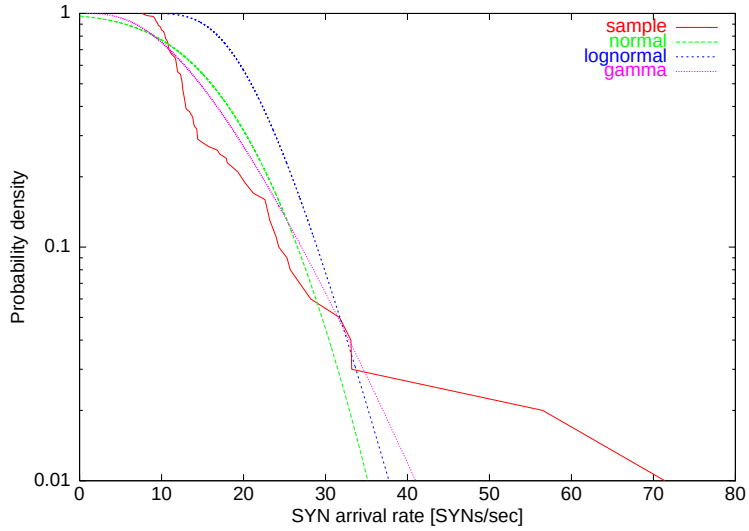
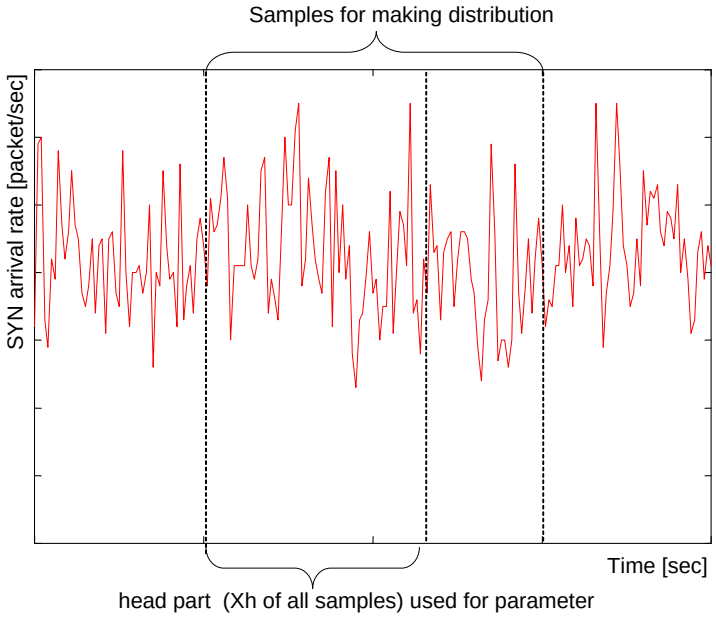
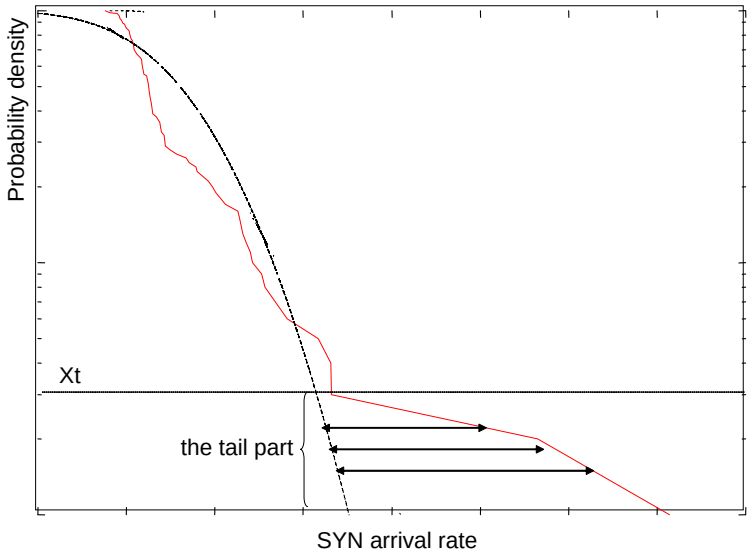


Figure 7: Distribution of SYN packet arrival rate when attacks started

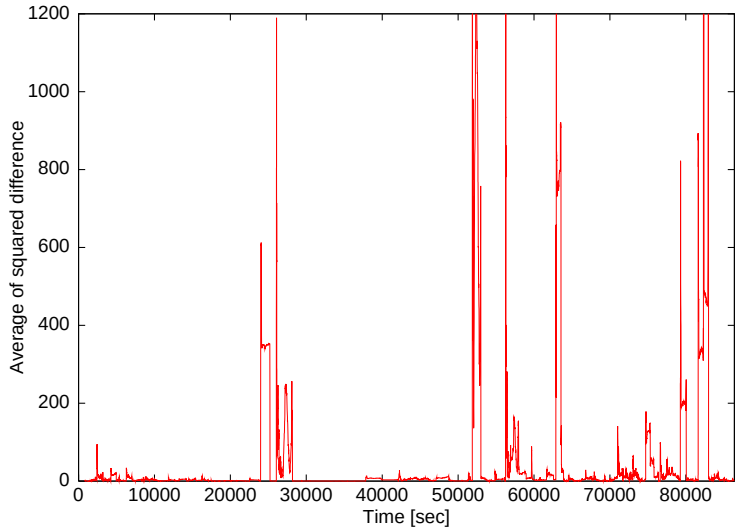


(a) head of distribution

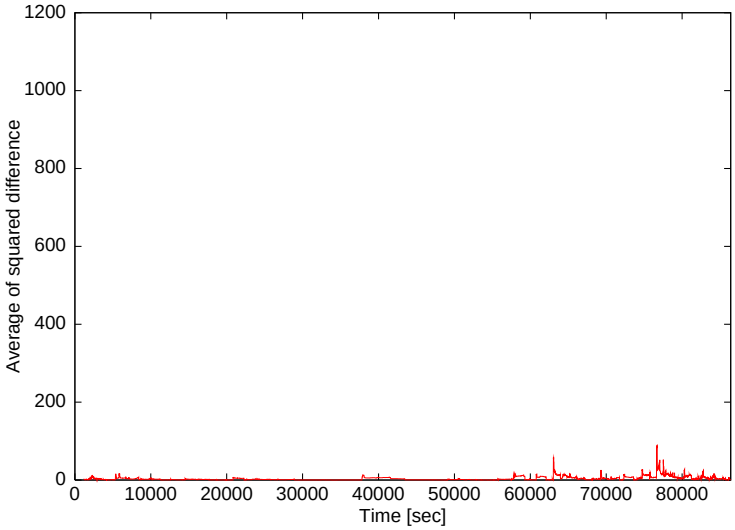


(b) tail of distribution

Figure 8: Outline of the average squared difference calculation

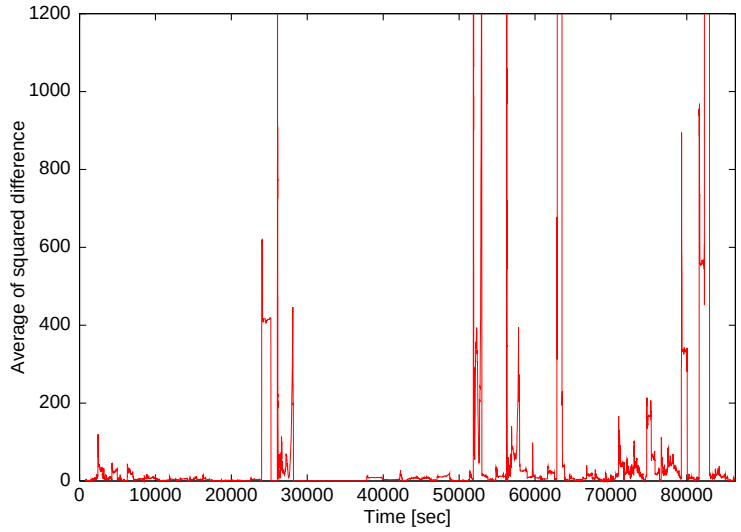


(a) all flows

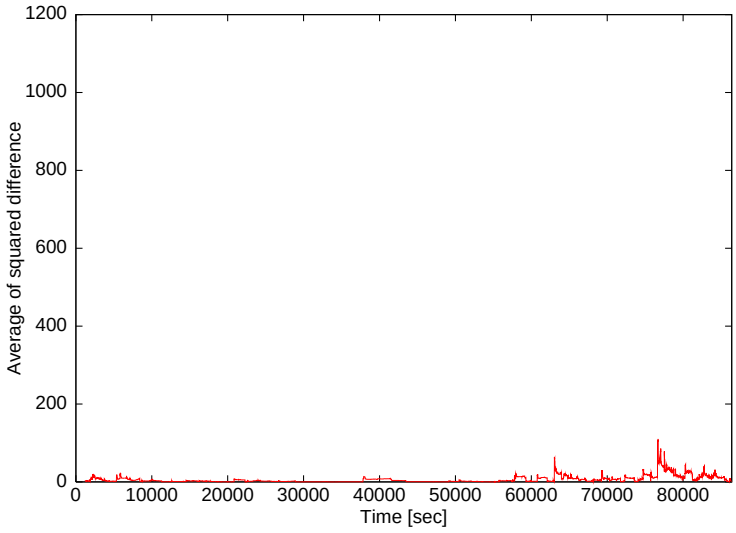


(b) normal traffic

Figure 9: Variation of average of squared differences between the sampled SYN rates and the gamma distribution

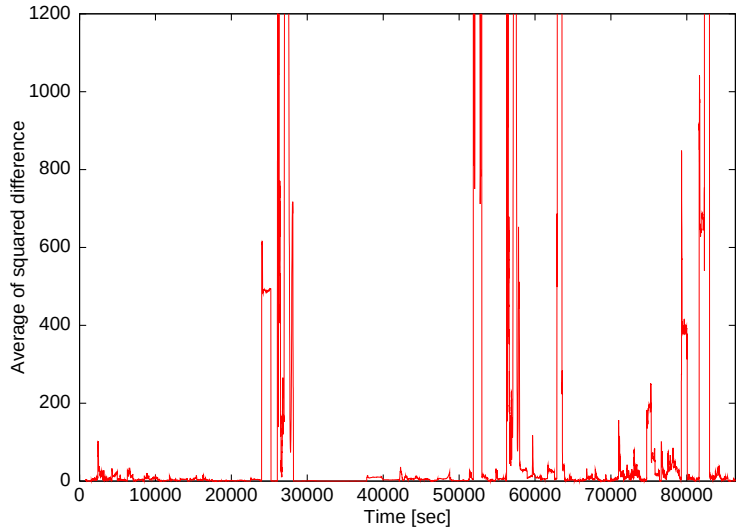


(a) all flows

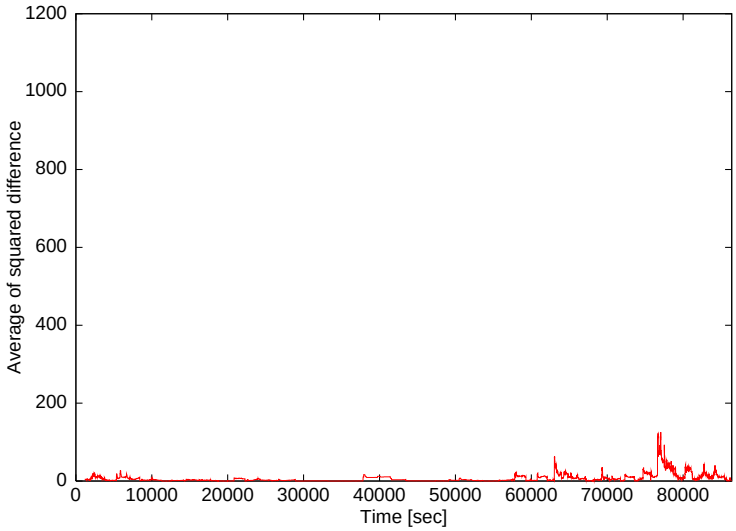


(b) normal traffic

Figure 10: Variation of average of squared differences between the sampled SYN rates and the normal distribution



(a) all flows



(b) normal traffic

Figure 11: Variation of average of squared differences between the sampled SYN rates and the lognormal distribution

4 Defense Mode

In defense mode, the defense node performs the following operations:

1. Alerting all defense nodes
2. Delegation of SYN/ACK packets
3. Protecting legitimate packets
4. Ending the defense mode

In the following sections, we describe these operations in detail.

4.1 Alerting all defense nodes

Figure 12 shows the steps to alert all defense nodes after an attack is detected. Once the attack is detected, the IP address of the victim node is sent to all defense nodes as alert messages via the overlay network. The defense nodes that receive the alert then move into the *defense mode*, and begin to return SYN/ACK packets for SYN packets whose destination addresses are that of the victim server.

Note that the propagation of the alert message depends on the topology of the overlay network. However, the problem of how to construct an effective overlay network is a separate research topic beyond the scope of this thesis.

4.2 Delegating SYN/ACK packets

In the defense mode, the defense node delegates SYN/ACK packets. When the defense node receives a SYN packet, it checks whether the destination address of the received packet is the IP address of the victim node. If the packet is intended for delivery to the victim node, the defense node returns a SYN/ACK packet to the address specified in the source address of the received packet. Then, after the defense node receives the acknowledgement for the SYN/ACK packet, it tries to establish a connection to the victim server. To identify whether the received ACK packets are acknowledgements of SYN/ACK packets, the defense node uses the data shown in Table 2 for each flow.

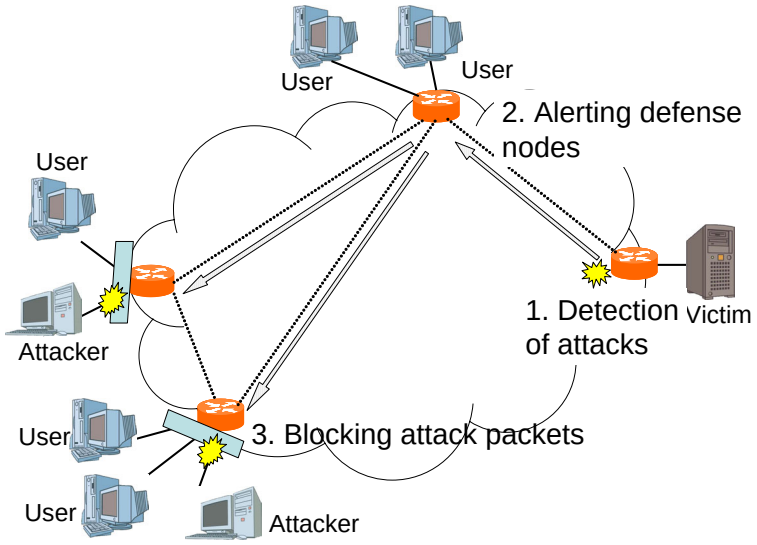


Figure 12: Steps of alerting

However, the defense node must hold a number of structures equal to the number of delegating SYN/ACK packets in the attack mode. The defense nodes should save their resources such as memory or CPU load while they hold legitimate connection requests even if they receive a number of SYN packets.

To save resources we use the same approach as the SYN cache mechanism. The SYN cache uses a hash table to search the data structures. The hash value is computed from the source and destination IP addresses and the source and destination port numbers. Entries having the same hash value are kept on a forward linked list. The length of the list is limited. When the list is full (i.e., the length of the link is equal to the maximum value) and a new connection request is received, the oldest (i.e., the head) entry in the list is dropped and a new request is appended at the tail of the list.

4.3 Protecting legitimate packets

We identify flows which complete the 3-way handshake as legitimate traffic. Once a flow is identified as legitimate traffic, the defense node relays packets of the flow to the server via the overlay network. In this thesis, we use the TCP Proxy [18] to relay legitimate traffic.

TCP Proxy is a method which controls transmission quality at the transport layer. TCP Proxies

Table 2: Data structure used to identify flows

Source address 32 bit	
Destination address 32 bit	
Initial sequence number (receiver) 32 bit	
Initial sequence number (sender) 32 bit	
Source port 16 bit	Destination port 16 bit
Timer	reserved for future use

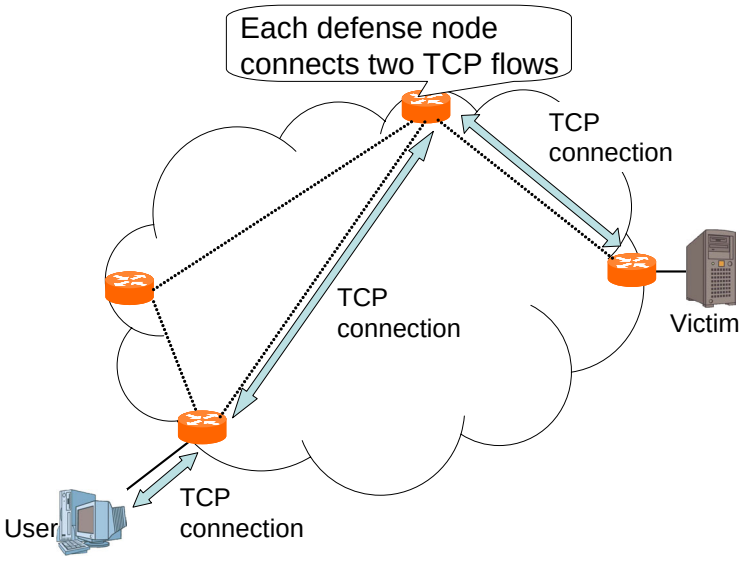


Figure 13: Relaying the legitimate packets

construct overlay networks and establish the connections to the next-hop TCP Proxies, which are determined according to the destination addresses. TCP Proxies relay packets by using hop-by-hop connections established via the overlay networks.

Figure 13 shows an overview of how legitimate packets are relayed using TCP Proxies. The defense nodes establish the hop-by-hop connections. Each node relays packets by connecting the flow from the previous hop and the flow to the next hop.

To connect the flow from the previous hop and the flow to the next hop, each defense node has to hold flows and search them quickly. The method in [19] can search flows quickly by using a hash table. To apply this method to our defense nodes, though, we need to adjust it. While [19]

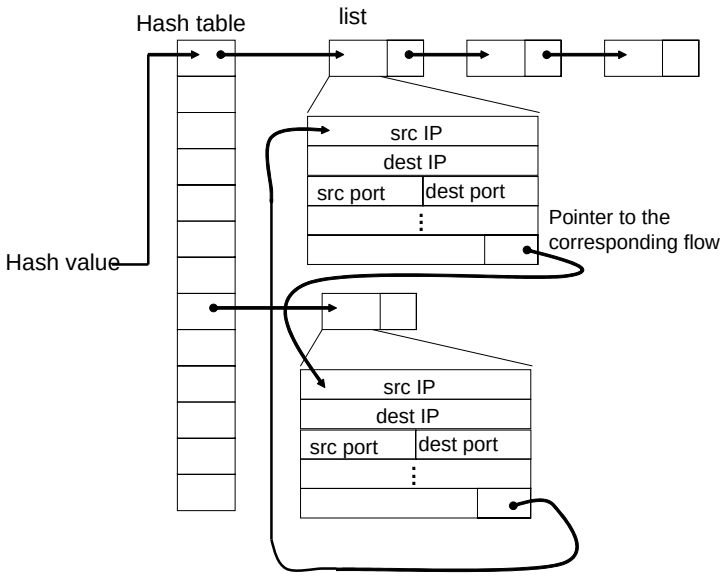


Figure 14: Data structure to hold normal flows

uses only (src IP, dest IP, server port) fields to identify flows, the defense nodes need to recognize flows having different client port numbers as different flows. Additionally, defense nodes need to know the corresponding flows to the next hop at the same time as they search the flows.

Figure 14 shows the data structure used in a defense node to hold normal flows. Entries having the same hash value are maintained in linked lists. In an entry, a defense node holds information needed by the TCP connection and the pointer to the entry of the corresponding flow. Upon receiving a packet to or from the victim, a defense node searches in the hash table for the flow of the packet. The defense node then forwards the packet to the corresponding flow.

4.4 Ending the defense mode

Since the resources of the defense node are limited, the defense mode should be terminated as soon as the attack ends. To enable this, it is necessary to detect the end of an attack at the defense node.

To detect the end of an attack, the defense node counts the number of connection requests (i.e., SYN packets) which time out or are dropped. When the number becomes 0, the attack is considered to have ended. Unlike attack detection, detection of the end of an attack does not have to be particularly fast since a long defense mode does not disturb legitimate connections.

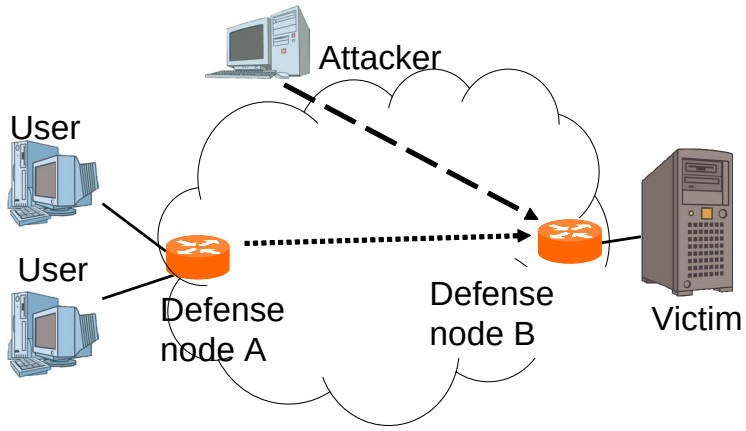


Figure 15: Problem in ending the defense mode

A problem arises, though, when all of the defense nodes independently detect the end of an attack. This situation is shown in Figure 15. In this figure, all of the attack traffic is passed via defense node D_B. If the detection is performed independently, defense node D_A first detects the end of the attack and stops delegating SYN/ACK packets. After finishing the delegation at D_A, all of the SYN packets passing D_A are subject to identification at D_B. The load on D_B thus increases because the total number of SYN packets increases, and D_B may drop some SYN packets because of the SYN cache limit on D_B. This degradation of performance will not occur if D_A continues to delegate SYN/ACK packets until D_B detects the end of the attack (i.e., the attack has completely ended in this case).

Therefore, the defense node should stop delegating SYN/ACK packets only when there are no attack packets at either the defense node or intermediate defense nodes on the way to the victim node.

Figure 16 shows the steps to stop delegating SYN/ACK packets. First, the defense node nearest to the victim node detects the end of the attack. This defense node sends a message indicating the end of the attack to all adjacent nodes (i.e., those logically connected from the defense node). A defense node receiving the message still delegates SYN/ACK packets until it detects the end of the attack. Upon detecting the end of the attack, each defense node ends the defense mode and forwards the message to the downstream adjacent defense nodes. The defense is completely ended after all defense nodes have received the message and ended the defense mode.

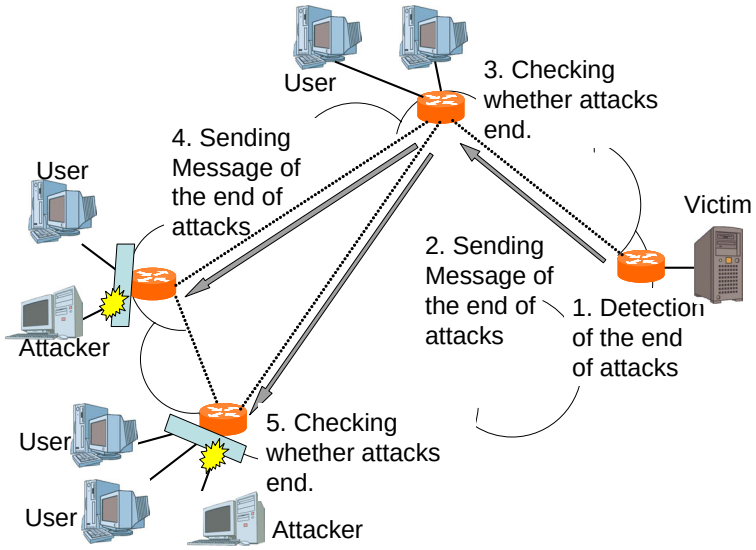


Figure 16: Steps to ending the defense mode

5 Deployment Scenario

In this section, we explain how our mechanism can be deployed in the Internet. In our mechanism, the unit of deployment is the AS. In this thesis, we refer to an AS in which our mechanism is deployed as a *protected AS*. All edge routers are defense nodes in a *protected AS*. Otherwise, an AS is referred to as *unprotected*. Figures 17 through 19 show the strategic scenario for the deployment of our defense mechanism. There are three stages as follows.

1st stage (Fig.17): Only one AS is *protected*. Others are *unprotected*.

2nd stage (Fig.18): Several ASes are *protected*.

final stage (Fig.19): All ASes are *protected*.

At the first stage, we consider our method to be deployed in only one AS, as shown in Figure 17. In this figure, AS 1 (the yellow cloud) is *protected*. Outside AS 1 all attack traffic to the victim node is first delivered to the victim node. The defense node nearest to the victim node then detects the attack traffic, and alerts the other defense nodes of the attack. Attack traffic is therefore blocked at the defense nodes placed at the edge of AS 1. In the case shown in Figure 17, our method enables AS 1 to block attack packets at three points. This means that our defense mechanism can defend against attack traffic up to three times as effectively as a single-point defense mechanism.

At the second stage of deployment (Figure 18), our method is deployed in several ASes which cooperate with each other. In the case shown in Figure 18, AS 1, AS 6, and AS 7 are *protected*. After an attack alert, the delegation of SYN/ACK packets is performed at the edge of the *protected* ASes. As a result, attack traffic generated in AS 6 and AS 7 is blocked at the egress edges of these ASes. Attacks from AS 2, AS 3, and AS 4 are blocked at the edge of AS 1 (the defense node for the link to AS 2). Attacks from AS 5 are also blocked at the edge of AS 1. Increasing the number of *protected* ASes means that attack traffic is blocked at more defense nodes. Moreover, the amount of legitimate traffic that our mechanism can protect may increase.

At the final stage of deployment (Figure 19), all ASes are *protected*. In the case shown in Figure 19, no attack packets reach AS 1 because all attack packets are blocked inside each AS. The attack traffic is no longer delivered to the core network when detected.

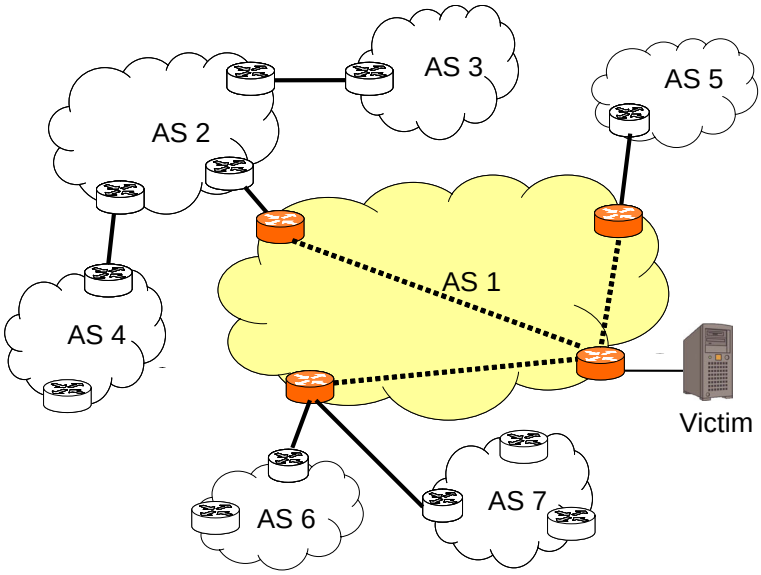


Figure 17: First stage of deployment

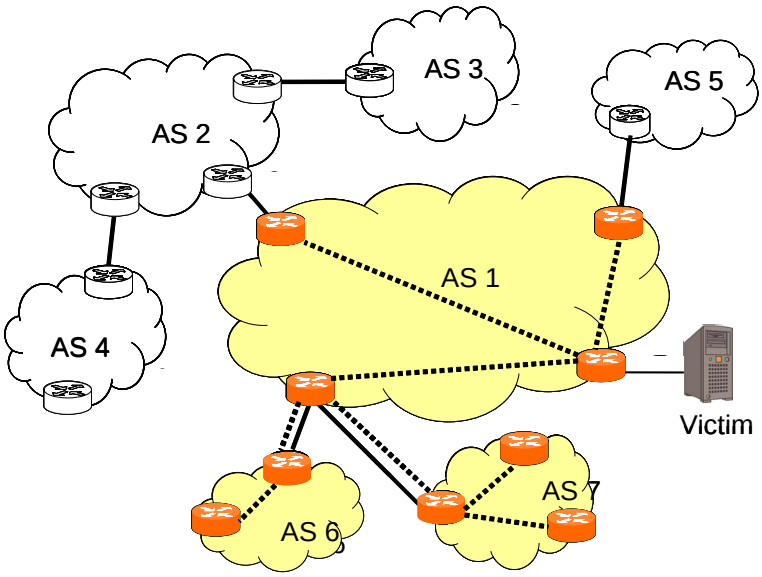


Figure 18: Second stage of deployment

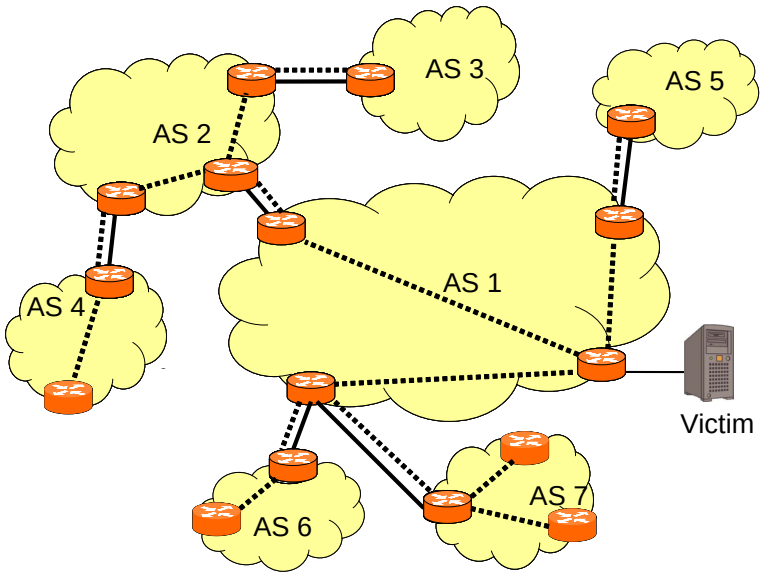


Figure 19: Final stage of deployment

6 Evaluation

6.1 Detection of attacks

6.1.1 Definition of attack traffic

Prior to the evaluation of our detection method, 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 3. The timeout values in this table are the durations until the half-open connections in the backlog queue are 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.

6.1.2 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:

$$\begin{aligned} P_- &= \frac{\sharp \text{ of attacks not detected}}{\sharp \text{ of attacks satisfying the definition}} \\ P_+ &= \frac{\sharp \text{ of points erroneously detected as attacks}}{\sharp \text{ of points detected as attacks}} \end{aligned} \tag{13}$$

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, X_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

Table 3: Default configuration of backlog queue

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

the viewpoints of fairness and resource managements, this relatively 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.

6.1.3 Detectable SYN rate of attack traffic

We also examined the SYN rates of attacks that can be detected without erroneous detection. 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.

6.1.4 Effect of parameters in our detection method

Figure 21 shows the SYN rates of attacks that can be detected as a function of parameter X_h. We could better detect lower-rate attacks by setting X_h to 75 rather than to 70. This is because a smaller X_h means a smaller number of samples are used to estimate the parameters and we cannot model as accurately. On the other hand, we could better detect lower-rate attacks by setting X_h to 85 rather than to 90. Too small a value of X_t makes the detection too sensitive, though, because the number of samples compared with the models is small.

Figure 22 shows the SYN rates of attacks that could be detected as a function of parameter N. In this case, we set X to 90 and M to 100. When N was too small, momentary high rates were erroneously detected. On the other hand, a larger N made attack detection less sensitive and more time was needed to detect attacks.

Figure 23 shows the SYN rates of attacks that could be detected as a function of parameter M. In this case, we set X_h to 90 and N to 100. When we set M to a larger value, we can model more accurately. However, we can better detect lower-rate attacks by setting M to 200 than by to 250. This is because a large M lessens the impact of attack traffic on the distribution of SYN arrival

rates and makes low-rate attacks difficult to detect. Moreover, a larger M requires more samples to detect attacks; i.e., more time is needed to detect attacks.

There is a trade-off between accuracy and quickness of detection. To defend servers against attacks, attacks should be detected quickly enough to prevent the backlog queue of the victim being filled by the attacks. For this reason, we set the parameters to values based on the maximum length of the server's backlog queue.

D becomes largest when the number of samples including attack packets is $M(1-X)$. The number of packets arriving before we obtain $M(1-X)$ samples is $MN(1-X)$. That is, the number of attack packets arriving before the attack is detected is less than $MN(1-X)$. Thus, by setting the parameters to make $MN(1-X)$ the maximum length of the server 's backlog queue we can detect attacks quickly enough.

6.1.5 Comparison of the three distribution functions

Figures 21 through 23 also show that there was no significant observation among the three distribution functions (normal, lognormal, and gamma). Therefore, we could use any of these functions to detect attack traffic under the conditions of our simulation. Regarding the deployment of our detection mechanism, though, the calculation complexity is also important. The calculation of the lognormal distribution is clearly more complex than that of the normal distribution. While the normal and gamma distributions both require much computational overhead, 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 standard normal distribution table. In summary, the normal distribution is the most appropriate for detecting attack traffic in terms of both accuracy and ease of implementation.

6.1.6 Setting the threshold

D is close to the squared difference between the arrival rate of all traffic and that of normal traffic. That is, D is related to the attack rate. Therefore, we compared the average of the squared difference with the attack rate. As Figure 24 shows, $\sqrt{D}$ was close to but slightly less than the attack rate. It was less than the attack rate because the rate of normal traffic when an attack occurs is not always at the tail of the distribution.

We generated normal traffic which followed normal distributions with different variances. We injected attacks whose rates were 100 SYNs/sec, 200 SYNs/sec, and 300 SYNs/sec. We then calculated D. Figure 25 shows that the attack rate minus $\sqrt{D}$ was proportional to the variance of the distribution of normal traffic. Thus, we could set the threshold value based on the squared attack rate to be detected and the variance of normal traffic.

6.1.7 Time needed to detect the attack traffic

Figure 26 shows the dynamics of the average squared difference from the beginning of an attack. The SYN rates of the attacks were 20 SYNs/sec, 24 SYNs/sec, and 28 SYNs/sec, and N was 200, M was 100, and X_h was 90. We used the normal distribution as the model distribution. This figure shows that the average squared difference increased gradually after the beginning of an attack. When the threshold was set to 20, which enabled attack detection without detecting any attacks erroneously, attacks with SYN rates higher than 28 SYNs/sec were detected within 20 seconds. In this case, the number of half-open states caused by an attack would be 560, which is smaller than the length of a backlog queue in Linux.

To show that our mechanism can detect attacks faster, we compared the time needed to detect attacks using our method with the time needed with the method proposed in [12]. We will refer to the latter method as the SYN-FIN method.

Here, we will briefly describe 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 it by the average number of RST or FIN packets F; $x_i = \Delta_i / F$. We then calculate y_i from

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

Finally, we determine whether traffic includes an attack by checking if the value of y_i exceeds the threshold T.

In the simulation, we set the values of α and T to 0.15 and 0.37, respectively. These were the optimized parameters to detect attacks as quickly as possible. In this simulation we used a normal distribution as the model and set N to 200, M to 100, and X_h to 90. We set the threshold D in our method to 20, which enabled attack detection without any attacks being detected erroneously.

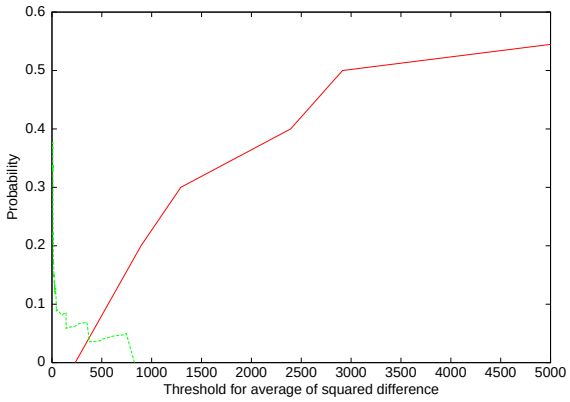
Figure 27 compares the time needed to detect attacks between our method and the SYN-FIN method. We varied the rate of attacking traffic and measured the time needed to detect the attacking traffic. From this figure, we can see that our method detected attacks much faster than the SYN-FIN method. One reason for this is that the SYN-FIN method uses a non-parametric approach to estimate differences between the characteristics of normal traffic and those of attacking traffic, while our method uses a parametric approach (i.e., we model the SYN rate of normal traffic as following a normal distribution). Detection with the parametric approach is faster and more accurate than with the non-parametric approach if the model is appropriate. However, the SYN-FIN method has an advantage in that it can also detect lower rate attacks (e.g., less than 14 SYNs/sec). Our method cannot detect these because traffic including low rate attacks still follows a normal distribution.

6.1.8 Resources needed by the detection method

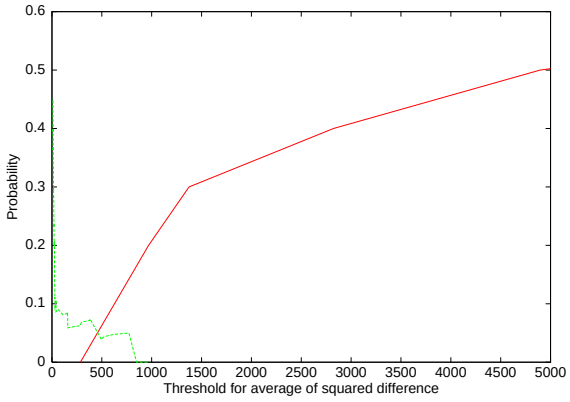
The above results show that our method can work with only 100 samples of the SYN rates. If we monitor D for each destination address, we need 100 samples for each address. The captured traffic has 1,000 destination addresses in 1,000 seconds of inbound traffic, and 10,000 destination addresses in 1,000 seconds of outbound traffic. According to Figure 4(b), the arrival rates were not so large and we can assume a small range of integer values (i.e., 16 bits) is enough for counting SYN rates. We then need 200 KBytes for incoming traffic and 2 Mbytes for outgoing traffic.

6.1.9 Availability on other networks

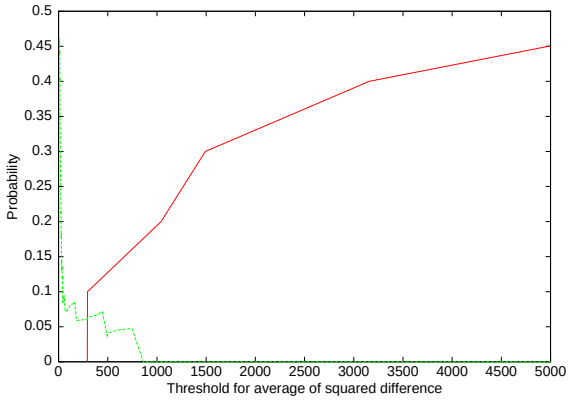
Though our detection method is based on analysis results for packets monitored at Osaka University, our method can be used in other networks. The distribution followed by normal traffic is due to the access timing of users whose tendencies do not vary with network scale. In any network, the arrival rates of SYN packets rise rapidly during SYN flood attacks. Our method monitors this rapid increase. In addition, the parameter setting described above is based on the number of packets and not affected by the scale of networks.



(a) gamma distribution



(b) normal distribution



(c) lognormal distribution

Figure 20: Relation between threshold for average of the squared difference and the probabilities of not detecting an attack (——) and of erroneously detecting an attack (- - -).

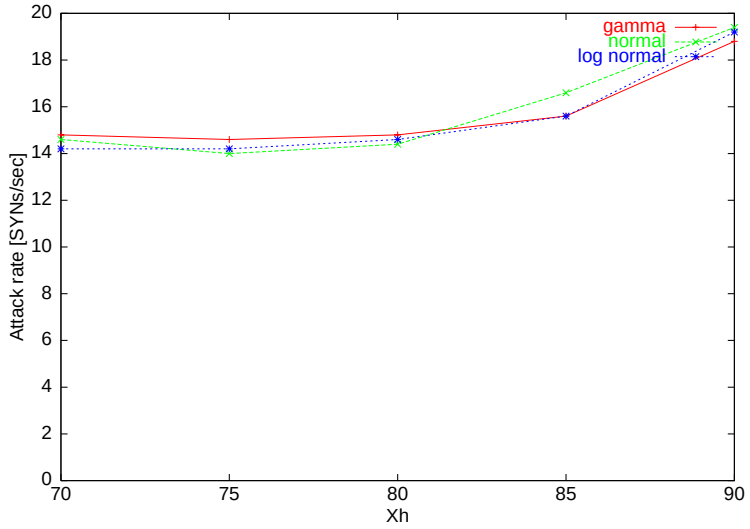


Figure 21: Relation between the detectable SYN rate of attack traffic and parameter X_h

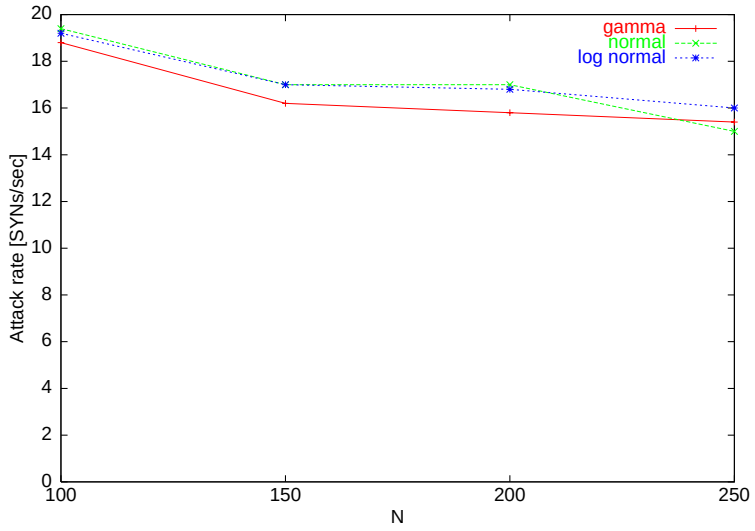


Figure 22: Relation between the detectable SYN rate of attack traffic and parameter N

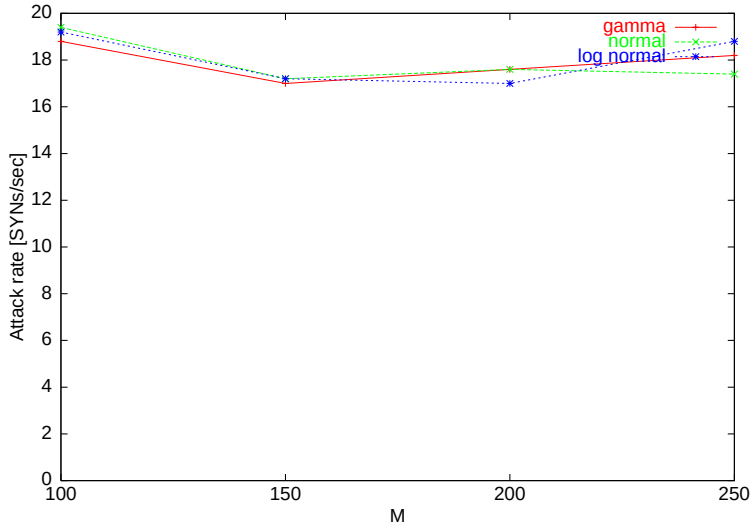


Figure 23: Relation between the detectable SYN rate of attack traffic and parameter M

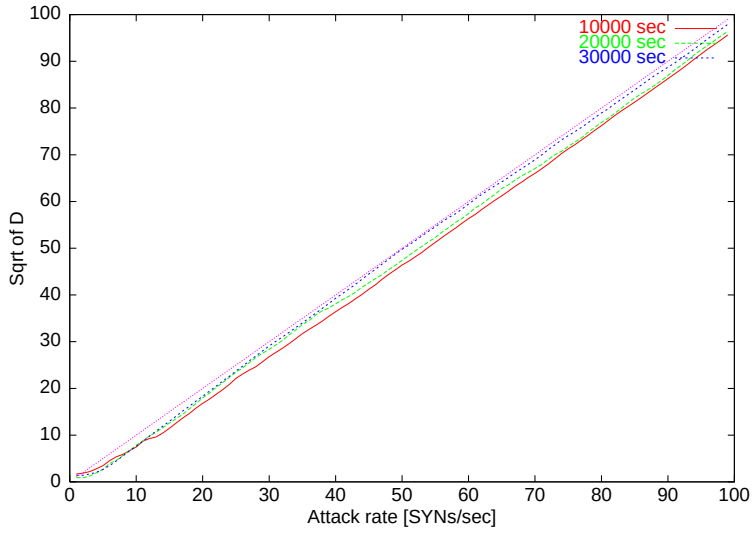


Figure 24: $\sqrt{D}$ vs. attack rate

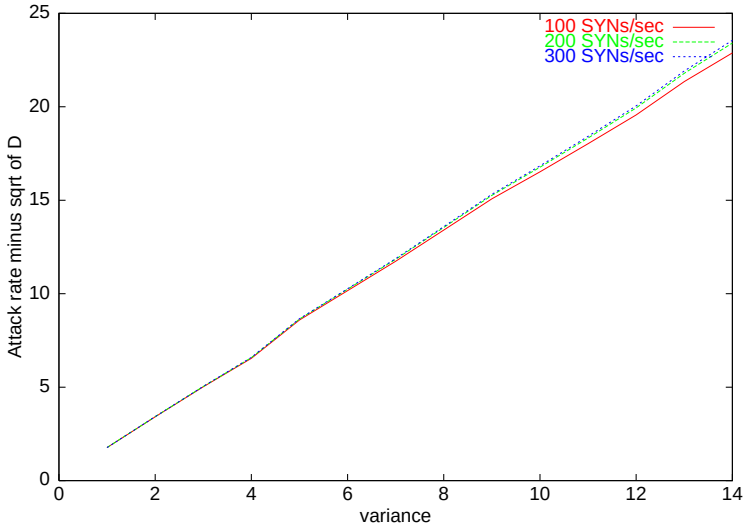


Figure 25: Attack rates minus $\sqrt{D}$ vs. variance of normal traffic

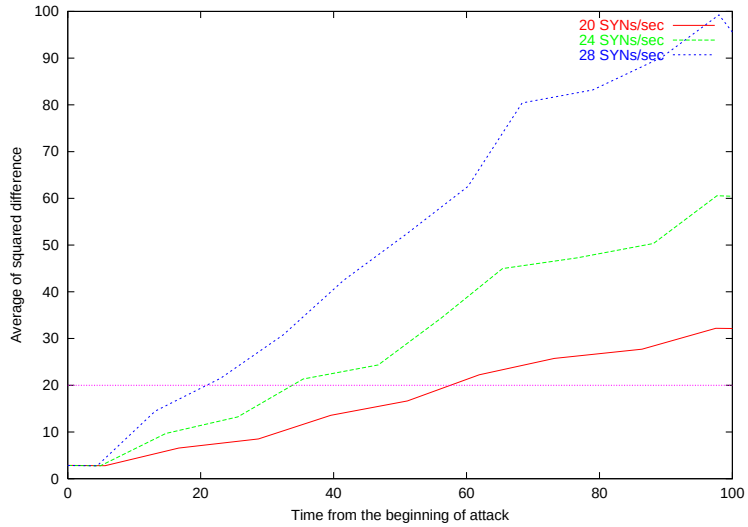


Figure 26: Average of squared differences versus time after the beginning of attacks with various SYN rates

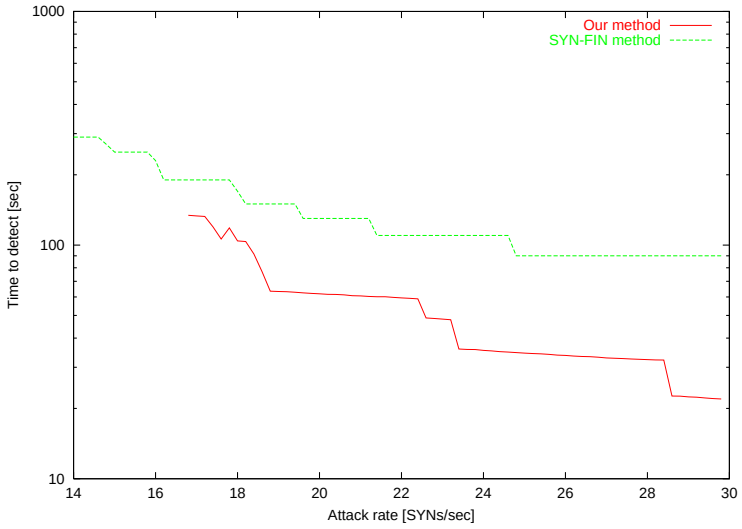


Figure 27: Time to detect attacks

6.2 Protection of legitimate packets

6.2.1 Probability of dropping legitimate SYN packets vs. attack rate

To demonstrate the effectiveness of a distributed defense mechanism, we compared the probability of dropping legitimate SYN packets when deploying an attacker-side defense mechanism (Figure 28(b)) with that when deploying a victim-side defense mechanism (Figure 28(a)). We assumed that the average round-trip time (RTT) between the clients and the victim server was 200 ms, and the average RTT between the clients and the attacker-side defense node was 20 ms. We also assumed that all attack packets were received by the same defense node. From the result described in 3, the SYN arrival rates of normal traffic would follow a normal distribution with a mean of 100 SYNs/sec. We set the SYN Cache parameters to the values used in FreeBSD.

Figure 29 shows the probabilities of legitimate SYN packets being dropped based on the rate of attack traffic. We have plotted three results: (1) without a defense mechanism, (2) with victim-side defense, and (3) with attacker-side defense. As shown, the attacker-side defense protected legitimate packets much better than the victim-side defense. This was because the RTTs between clients and the attacker-side defense node were much shorter than the RTTs between clients and the victim-side defense node. The average holding time for each connection request on the SYN cache was also short, which increased the availability of the SYN cache.

[4] reports observing attacks whose rates exceeded 600,000 SYNs/sec. In the event of such high-rate attacks, victim-side defense cannot protect legitimate packets and the probability of dropping legitimate SYN packets rises to almost 1. On the other hand, if we deploy attacker-side defense, the probability of dropping legitimate SYN packets would be less than 0.1. Moreover, attacker-side defense is much more effective because attackers are distributed and the attack rates at each defense node are low.

6.2.2 Time dependent variation of probability of dropping legitimate SYN packets

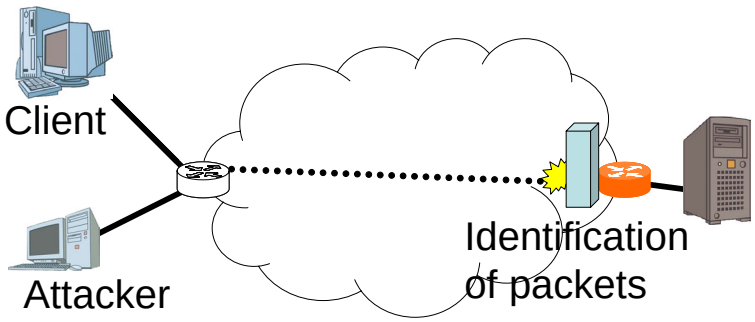
We next consider the effectiveness of the distributed defense mechanism. We consider the three scenarios shown in Figure 30. Case 1 is a single-point defense mechanism. In Case 2, there are several defense nodes on the edge of the network, but not all edge nodes are defense nodes. In Case 3, all edge nodes except one (the ingress node for Client A) are defense nodes.

We injected attack traffic after a certain period of time from the beginning of the simulation.

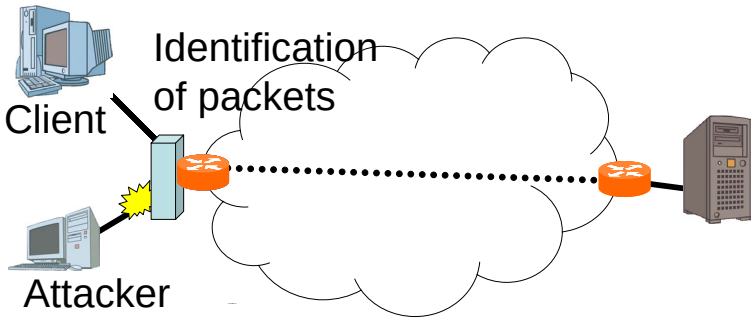
Each attacker sent 200,000 attack packets per second, and the attack began 100 sec from the simulation start and ended at 200 sec from the simulation start. Each client sent 30 SYNs/sec.

We plotted the time dependent variation of the probability of dropping legitimate SYN packets at three points for each case. Client A was connected to a non-defense node. Client B had a attacker node on the same segment. Client C was connected to a defense node in Cases 2 and 3, and there was no attacker node on the same network.

In Case 1, none of the clients could send legitimate SYN packets to the victim node. On the other hand, in Case 2, the probability of packet loss for Client B and Client C became very low soon after the attack started while the probability for Client A remained high. This is because our method quickly detects attacks and protects legitimate packets. In Case 3, the probabilities were very low for all clients. This result shows that our method can effectively protect legitimate packets and block attack packets.



(a) Victim-side defense



(b) Attacker-side defense

Figure 28: Environment supposed in our simulations

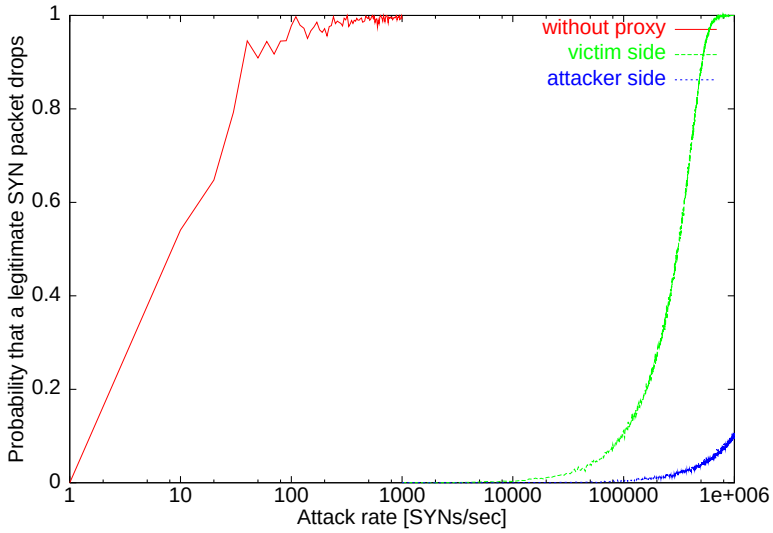
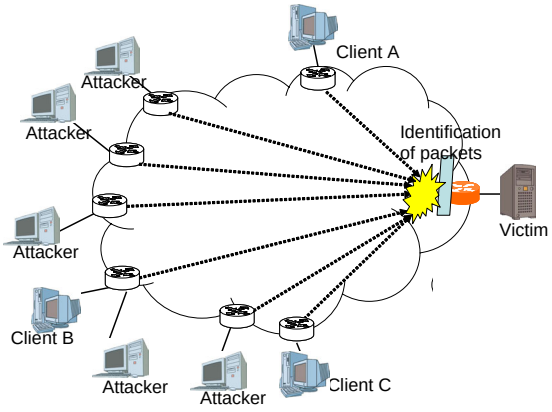
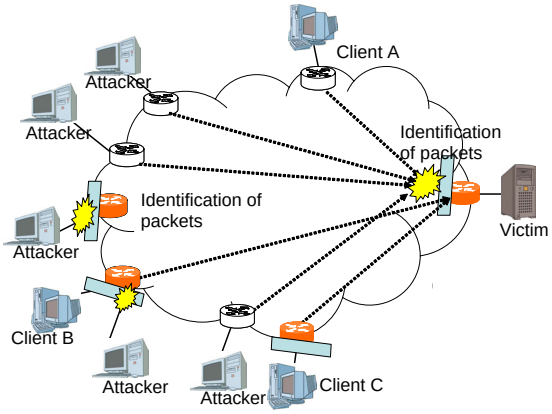


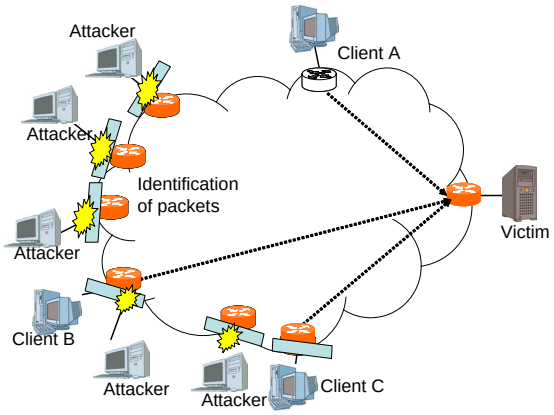
Figure 29: Probability of dropping legitimate SYN packets vs. attack rate



(a) Case1: Single-point defense

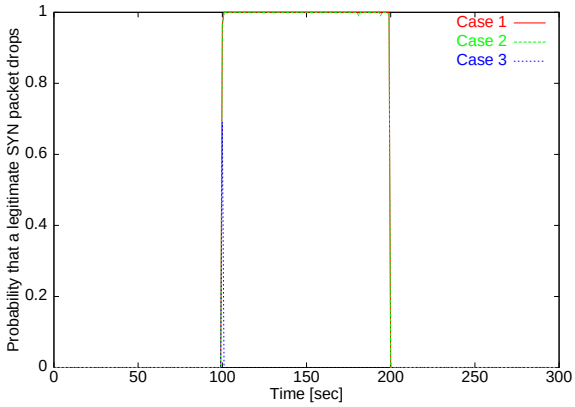


(b) Case2: Some of ASes including attackers deploy our method

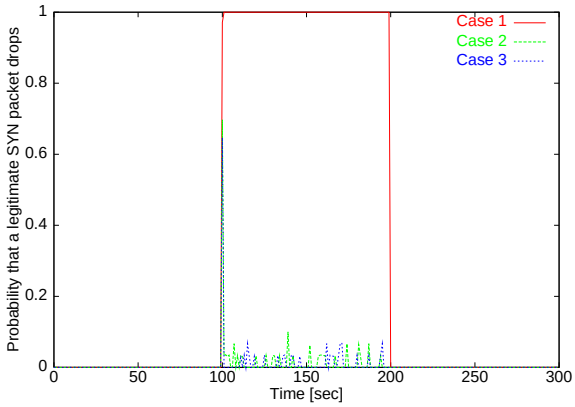


(c) Case3: All ASes including attackers deploy our method

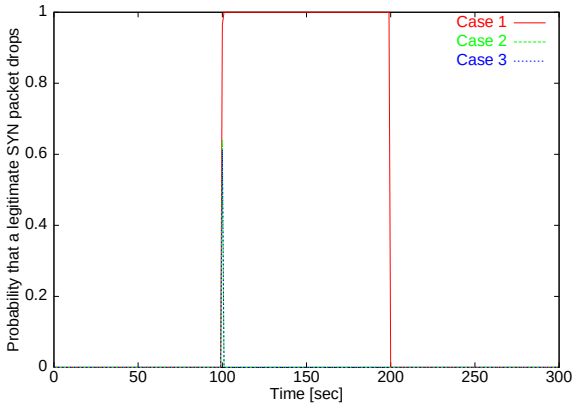
Figure 30: Environment supposed in our simulations



(a) Client A



(b) Client B



(c) Client C

Figure 31: Probability of dropping legitimate SYN packets

7 Conclusion and Future Work

We have developed a distributed detection and defense mechanism to protect against distributed SYN Flood attacks. Our mechanism is based on collaboration among distributed defense nodes through a constructed overlay network. Such an overlay network is an effective way to provide attack alerts and protect legitimate traffic.

To detect attacks accurately, we propose a detection method that takes the time-of-day variance of traffic into consideration. First, 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. Based on this result, we described a new attack detection method.

Through simulation, we have shown that our method can detect and block attack packets quickly and accurately regardless of the time variance of the traffic. We have also shown the effect of attacker-side defense and the effectiveness of our method.

One example of our future work will be to develop ways of identifying attack packets at the points where the routes of packets may vary.

Acknowledgement

I would like to express my sincere appreciation to Professor Masayuki Murata, who was my supervisor and has given me tremendous advice throughout my studies and in the preparation of this thesis.

The work described in this thesis would not have been possible without the support of Lecturer Shingo Ata of Osaka City University. He has always provided me with appropriate guidance and invaluable advice.

I am also indebted to Associate Prof. Naoki Wakamiya, Hiroyuki Ohsaki, Go Hasegawa, and Research Associates Shin'ichi Arakawa, Ichinoshin Maki, and Masahiro Sasabe who have given me helpful advice and support.

Finally, I want to heartily thank my many friends and colleagues in the Department of Information Networking of Osaka University for their support.

References

[1] "CERT advisory CA-1998-01 smurf IP Denial-of-Service attacks." available at `http://www.cert.org/advisories/CA-1998-01.html`, Jan. 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`, Sept. 1996.

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

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

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

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

[8] S. Floyd, S. M. Bellovin, J. Ioannidis, K. Kompella, R. Manajan, and V. Paxson, "Pushback messages for controlling aggregates in the network." draft-floyd-pushback-messages-00.txt, internet-draft, work in progress, July 2001.

[9] J. Mirkovic, M. Robinson, P. Reiher, and G. Kuenning, "Alliance formation for DDoS defense," in *Proceedings of the New Security Paradigms Workshop, ACM SIGSAC*, pp. 11–18, Aug. 2003.

[10] Y. Kim, W. C. Lau, M. C. Chuah, and H. Chao, "PacketScore: Statistics-based overload control against distributed denial-of-service attacks," in *Proceedings of IEEE INFOCOM 2004*, Mar. 2004.

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

[12] H. Wang, D. Zhang, and K. G. Shin, "Detecting SYN flooding attacks," in *Proceedings of IEEE INFOCOM 2002*, vol. 3, pp. 1530–1539, June 2002.

[13] 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`, Nov. 2002.

[14] Y. Ohsita, S. Ata, and M. Murata, "Detecting distributed Denial-of-Service attacks by analyzing TCP SYN packets statistically," in *Proceedings of IEEE Globecom 2004*, Nov. 2004.

[15] Y. Ohsita, S. Ata, and M. Murata, "Deployable overlay network for defense against distributed SYN flood attacks," *Technical Reports of IEICE*(IN2003-125), pp. 13–18, Dec. 2004.

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

[17] 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, Apr. 2000.

[18] I. Maki, G. hasegawa, M. Murata, and T. Murase, "Throughput analysis of TCP proxy mechanism," in *Proceedings of ATNAC 2004*, Dec. 2004.

[19] S. Ata, M. Murata, and H. Miyahara, "Efficient cache structures of IP routers to provide policy-based services," in *Proceedings of IEEE ICC 2001*, June 2001.

[20] Y. Ohsita, S. Ata, and M. Murata, "Detecting distributed denial of service attacks by utilizing statistical analysis of TCP SYN packets," *Technical Reports of IEICE*(IN2003-201), pp. 23–28, Feb. 2004.

[21] Y. Ohsita, S. Ata, and M. Murata, "Detecting distributed Denial-of-Service attacks by analyzing TCP SYN packets statistically," *submitted to IEICE Transactions on Communications*, Nov. 2004.