PAPER

Deployable Overlay Network for Defense against Distributed SYN flood Attacks

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. Most of them are SYN flood attacks, since the malicious attackers can easily exploit the TCP specification to generate traffic making public servers unavailable. We need a defense method which can protect legitimate traffic so that end users can connect the target servers during such attacks. In this paper, we propose a new framework, in which all of the TCP connections to the victim servers from a domain are maintained at the gateways of the domain (i.e., near the clients). We call the nodes maintaining the TCP connection *defense nodes*. The defense nodes check whether arriving packets are legitimate or not by maintaining the TCP connection. That is, the defense nodes delegate reply packets to the received connection request packets and identify the legitimate packets by checking whether the clients reply to the reply packets. Then, only identified traffic are relayed via overlay networks. As a result, by deploying the defense nodes at the gateways of a domain, the legitimate packets from the domain are relayed apart from other packets including attack packets and protected. Our simulation results show that our method can protect legitimate traffic from the domain deploying our method. We also describe the deployment scenario of our defense mechanism.

key words: *Distributed Denial of Service (DDoS), SYN flood, Overlay Network, TCP Proxy*

1. Introduction

The recent rapid growth of and increased use of the wide use of the Internet are making Internet security issues increasingly important. Distributed Denial-of-Service (DDoS) attacks are one of the most serious problems. The DDoS attack causes serious damage at the victim server by increasing the number of hijacked nodes even if the rate of attack traffic generated by each node is quite small. Recently, there are many kinds of DDoS attacks such as Smurf attacks [1], UDP floods [2], and SYN flood attacks [3]. In SYN flood attacks, attackers send so many connection requests to one (i.e., victim) server that end users cannot connect to it. In recent reports, most DoS attacks are SYN flood attacks, which are the second most frequent type of attacks among all attacks [4]. In addition, according to [4], recently many bot networks use SYN flood attacks to shut down target servers. As a result, the number of SYN flood attacks is still increasing.

Therefore, we need a defense method which can protect legitimate traffic so that end users can connect the target servers during such attacks. Prior to mention about the defense methods against SYN flood attacks, we describe the requirements on the defense methods below.

- **R1** Distinguish and protect legitimate packets accurately even during very heavy attacks. In DDoS attacks, attack nodes are widely distributed all over the world. Attack traffic from attack nodes is aggregated into a very heavy attack at the server.
- **R2** Protect legitimate packets from a domain to the victim server even if the intermediate domains do not deploy the methods. It is often the case that the intermediate domains do not deploy the methods because it is difficult to deploy new mechanism in the whole Internet at once.
- **R3** Work transparently with existing nodes. That is, defense methods should not require any modifications or software updates on servers and clients, because it is difficult to modify a large number of nodes at once.
- **R4** Have no impacts on the traffic which can connect the server even without the protection. If the traffic is treated by a defense method, the process of defense methods may become the performance bottleneck and cause the increase of the end-to-end delay and so on.

Several methods against SYN flood attacks are proposed [5]–[17]. However, none of them fulfill all of four requirements described above. Detailed comparisons to related works are described in Section 2.

Therefore, in this paper, we propose a new system to fulfill all of four requirements described above. In our method, the identification and protection of the outgoing legitimate traffic from a domain are performed by the node deployed at the gateways of the domain. We call the node *defense node*. Unlike existing methods, the purpose of our method is NOT eliminate the attack traffic, but the main focus of our method is to protect of legitimate traffic which are generated during actual communications between valid users. For this motivation, we need major two mechanisms: (1) identify the legitimate traffic accurately, and (2) transfer/protect legitimate traffic safely. Conventionally, these two mech-

[†]Graduate School of Economics, Osaka University
[††]Graduate School of Engeneering, Osaka City University
[†††]Graduate School of Information Science and Technology, Osaka 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

anisms are so difficult and unrealistic for the deployment because handling all the traffic requires too many resources, however, we consider the possibility of deployment technically in depth, and found that they are deployable in actual by combining TCP proxy and overlay network and handling only the packets which are mixed with attack packets on the way to the destination server.

The key ideas of our method are as follows.

- Identify legitimate traffic from a domain by maintaining the TCP connections at the gateways of the domain. That is, the defense nodes delegate reply packets to the received connection request packets and identify the legitimate packets by checking whether the clients reply to the reply packets. Unlike the traditional firewalls delegating the reply packets near the victim servers, our method can immediately and accurately identify the legitimate traffic without dropping the legitimate connection requests even during heavy attacks, because defense node does not hold the legitimate connection request as long as victim servers since round trip times (RTTs) between the clients and the defense nodes are much smaller than the RTTs between clients and the victim servers. That is, our method fulfills R1. In addition, by maintaining the TCP connections, we can identify legitimate traffic without any modifications of clients or servers (R3).
- Protect legitimate packets by relaying them via overlay networks. By using overlay networks, our method can relay legitimate packets apart from other packets including attack traffic, even if intermediate domains do not deploy our method. That is, our method fulfills R2.
- Not maintain the TCP connections of the traffic but relays them as normal IP packets, if legitimate traffic from the domain of the defense node is not mixed with attack packets on the way to the destination server (e.g., the case that attack packets are blocked at other domains). That is, our method fulfills R4.

Our system performs as follows. First, attacks are detected by victim-side defense nodes which can detect attacks easily. After an attack is detected, alert messages are forwarded to all nodes via the overlay networks. The edge defense nodes which received the 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. After that, if a defense node detects that there are no attack packets at the intermediate domains to the victim node, the defense node stops the defense mechanism.

In Section 2, we describe the existing methods and their limitations. In Section 3 we explain the overview of our defense mechanism and describe the detailed operation. In Section 4, we explain the deployment scenario of our mechanism. In Section 5, we show some simulation results that our method can effectively block attack traffic and protect legitimate traffic. In Section 6 we conclude by briefly summarizing the paper and mentioning some of the future works we intend to do.

2. Related Works

Several methods against SYN flood attacks have been proposed so far. SYN cookie [5] and SYN cache [6] are mechanisms deployed on server (victim) nodes. The SYN cookie mechanism can remove the queue for connection requests by embedding the information in the SYN packet 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. On the other hand, 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, heavy attacks from widely distributed nodes can overwhelm the firewalls or the servers regardless of implementation of these methods, because one server deals with many attack packets from many attacker nodes. That is, these methods do not fulfill R1. For this reason, a distributed defense mechanism is needed to defend servers from distributed attacks.

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 because 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. However, these methods to limit the rate of traffic also limit the rate of the legitimate traffic to the victim. That is, these method cannot protect legitimate traffic and do not fulfill R1. On the other hand, our method does not limit the rate of traffic but block only flows identified as attack by maintaining the TCP connections. In addition, flows identified as legitimate traffic are protected by relaying via overlay network. Therefore, our method does not drop the legitimate traffic.

DefCOM [11] proposes another framework to mark suspicious packets at edge nodes and limit the rates for the suspicious packets at core nodes. PacketScore [12] and ALPi [13] are similar methods. In these meth-

ods, edge nodes compute a per-packet score which is used to estimate the legitimacy of a packet. Core nodes perform score-based selective packet discarding. These filtering methods can effectively mitigate attacks when the characteristics of attack packets differ from those of legitimate packets. However, the packets used in SYN flood attacks do not differ from legitimate SYN packets except in the spoofing of the source addresses. When attack packets have almost the same characteristics as legitimate packets, legitimate traffic may be mistakenly identified as attacks and blocked by these methods. This can seriously impair the communication between the victim and legitimate clients because mistakenly blocking legitimate packets significantly increases the packet loss rates between the legitimate clients and the victim. That is, these methods do not fulfill R1. On the other hand, our method can identify legitimate packets accurately by maintaining the TCP connections to the victim.

Another framework is proposed in [9]. In this framework, traffic monitors called watchdogs are placed at distributed points. When attack occurs, watchdogs send the information about the attack to other watchdogs by using multicast. Then, a packet filter to prevent the detected attack is installed at distributed points. MovingFirewall [10] is also a framework to drop attack packets at distributed places. In this framework, after an attack is detected near the victim, firewalls trace attack flows upstream. Then, the firewall nearest to the attackers mitigates the attack traffic using the attack signature.

However, these frameworks only block attack packets and do not consider how to protect the legitimate packets. Because legitimate packets checked by a firewall are relayed as normal IP packets, it may be possible that the legitimate packets are mixed with attack traffic again and cannot be protected. One of the examples is the case where domain A deploys these frameworks and does not directly connect to other domains deploying these frameworks. In this case, though attack packets from domain A are blocked, the legitimate packets from domain A are mixed with attack packets at the intermediate domain which does not deploy these frameworks. As a result, the legitimate packets from domain A are also checked by the firewalls deployed at the other domains, and when the firewalls have heavy load, the legitimate packets from domain A may also be dropped. That is, though domain A deploys these frameworks, whether the legitimate packets from domain A are dropped or not depends on the loads of the firewalls at the other domains, as is the case without deploying these frameworks. That is, these frameworks do not fulfill R2.

On the other hand, in our method, flows identified as legitimate are relayed via overlay network. By relaying the identified traffic via overlay networks, we can relay and protect the identified traffic distinctly from other traffic, even in the case that the source domain of the identified traffic is not directly connected.

The frameworks using the overlay network to protect legitimate traffic have also been proposed as SOS [14]–[16], Mayday [17] or FON [18]. WebSOS [16] is one of applications of SOS, in which SOS is applied to defense mechanism for web servers. In this framework, packet filtering is set so that only packets via overlay network can reach the server. The route to the server on the overlay network is randomly selected from candidate routes for each session to distribute the traffic to the server. Additionally, to avoid attack traffic entering the overlay network, clients are authenticated at the edge of overlay network. However, authentication requires clients to implement a kind of authentication software. Traffic from unauthorized clients (i.e., clients not installed the authentication software) are classified into attack traffic, and not protected by these frameworks. That is, these methods require the modification of all clients and do not fulfill R3. On the other hand, our method identifies attack packets without any modification of clients by maintaining the TCP connections. In addition, in this method, all packets to the victim server are relayed via overlay network even if the packets are not mixed with attack packets on the way to the server. In this case, due to random routing or overheads of overlay nodes, the delays of such legitimate traffic which does not require the protection may increase unnecessarily. That is, these frameworks do not fulfill R4. On the other hand, in our method, defense nodes maintain the TCP connection only for the packets which cannot connect to the server without protection.

3. Defense method to protect legitimated traffic from a domain by using overlay networks

3.1 Overview of defense method to protect legitimate traffic

Figure 1 shows an overview of our proposed architecture. We place a node called *defense node* at the gateways of domains. Each defense node logically connects to one or more other defense nodes, and constructs an overlay network among the defense nodes. Each defense node identifies legitimate SYN packets by returning SYN/ACK packet instead of the victim node. Then, the SYN packet is relayed only when the defense node receives the ACK packet of the SYN/ACK packet from the client (Fig. 2). Because defense node does not hold the SYN packets as long as victim server due to small RTTs, even if heavy attacks occur, the defense nodes can identify legitimate packets from the domain without dropping the legitimate connection requests. 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

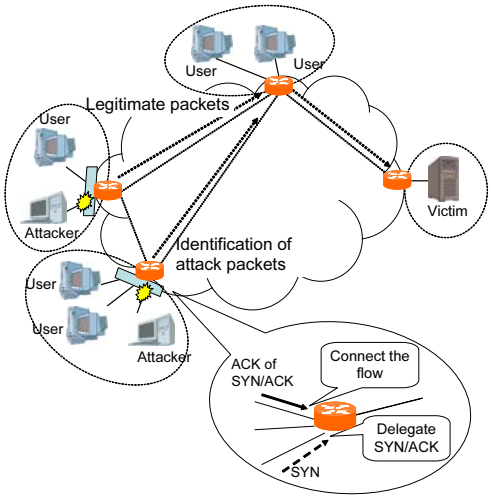


Fig. 1 Overview of our framework

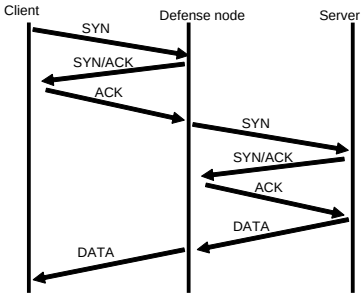


Fig. 2 Delegation of SYN/ACK packets

overlay network and distinguished from attack traffic.

In the ideal situation, the defense node should handle all arriving packets and pick up legitimate packets from 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 requiring to be protected (i.e., packets which will be mixed with attack packets on the way to the destination server). For this purpose, we use a mechanism for detecting the starts and ends of SYN flood attacks. A defense node has two modes, *attack detection mode* and *defense mode* and moves between two modes as shown in Fig. 3.

In the *attack detection mode*, the defense node monitors packets and checks whether the arriving traffic is attack or not. This check is performed at the server side. This is because it is difficult to detect highly distributed attacks at edge routers or core networks since the number of attack packets is very small there. Additionally, though there are several proposals to detect attacks, each method has pros and cons. For example, though a more complex method can detect attacks more accurately, it requires more resources such as CPU and memories. Therefore, we separate methods to detect

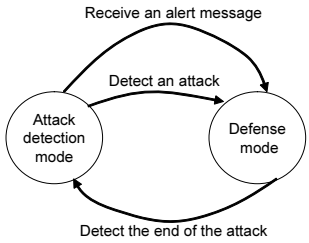


Fig. 3 State transition diagram between attack detection mode and defense mode

attacks from our framework so that we select any detection methods according to the situations or policies of the domain. In our evaluation described in Section 5, we use the method proposed in [19]. This method detects attacks by comparing the SYN arrival rates with normal distributions. This method can detect attacks fast regardless of time variation of traffic.

When the defense node detects attack traffic, the defense node moves into the defense mode for the victim's address and sends alert messages to other defense nodes. Then, each defense node receiving the alert message also moves into the defense mode for the victim's address.

In the *defense mode*, the defense node delegates SYN/ACK packets in order to identify legitimate traffic and relays the identified traffic apart from other traffic by using the overlay networks. The defense mode is continued until there becomes no attack traffic in the intermediate domains.

In the following subsections, we describe the details of the above operations.

3.2 Changing the modes of defense nodes

3.2.1 Starting defense mode

When the defense node detects attack traffic, the defense node moves into defense mode for the victim's address by adding the address to the *victim server list* held by each defense node. Then, the defense node alerts other defense nodes.

Figure 4 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* by adding the address to their victim server lists. Then, they begin to return SYN/ACK packets for SYN packets whose destination addresses are that of the victim server. These alert messages are generated at the event of attack detection, and forwarded once for each defense node. No other alerts are forwarded during the defense mode, except the events of ending the defense mode. These alerts therefore do not strictly affect on the network bandwidth availabil-

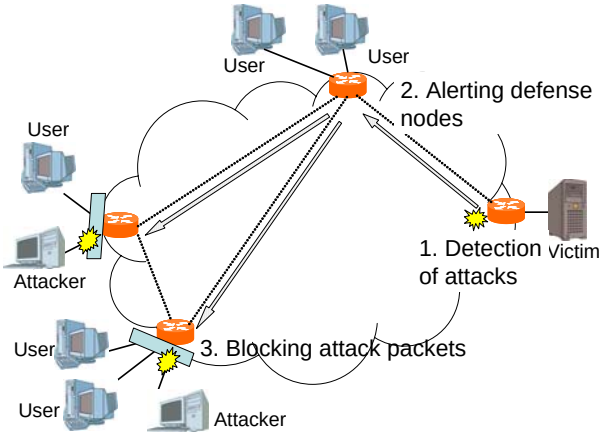


Fig. 4 Steps to start defense mode

ity.

3.2.2 Ending the defense mode

Since the resources of the defense node are limited, the defense mode should be terminated as soon as the protection of legitimate traffic becomes unnecessary (i.e., there becomes no attack traffic on the way to the victim server). To enable this, it is necessary to detect the nonexistence of attack traffic.

To detect the nonexistence of attack traffic, the defense node counts the number of connection requests (i.e., SYN packets) which time out or are dropped. When the number is 0 for a given length of time (T_{end}), the defense node is considered to receive no attack traffic. Unlike attack detection, detection of the nonexistence of attack traffic does not have to be particularly fast since a long defense mode does not disturb legitimate connections. By waiting for a pre-defined timeout, we can also protect legitimate traffic against pulsing attacks in which the attack traffic oscillates between the maximum rate and zero.

However, even when a defense node receives no attack packets, it is possible that the legitimate traffic will be mixed with attack traffic if they are relayed as normal IP packets. This situation is shown in Fig. 5. In this figure, all of the attack traffic is passed via defense node D_B and D_A receives no attack packets. However, if D_A finishes 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 delegated by D_B 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 there becomes no attack traffic on D_B (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

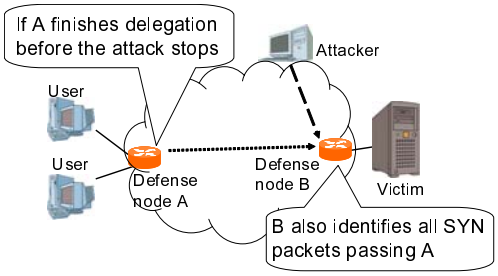


Fig. 5 Problem in finishing the defense mode

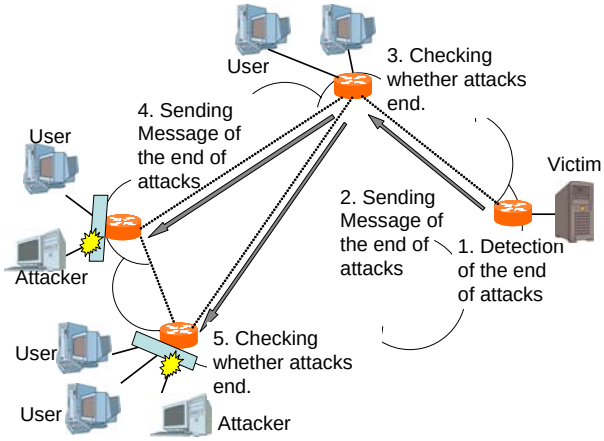


Fig. 6 Steps to finish the defense mode

to the victim node.

Figure 6 shows the steps to end the defense mode. First, the defense node nearest to the victim node detects the nonexistent of attack traffic. This defense node ends the defense mode by deleting the corresponding IP address from the victim server list. Then, the 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 remains in the defense mode until it detects the nonexistent of attack traffic. Upon detecting the nonexistent of attack traffic, the defense node ends the defense mode by deleting the corresponding IP address from the victim server list, 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.

3.3 Identification and protection of legitimate traffic during the defense mode

When a defense node is in the defense mode for a victim server, the defense node identifies and protects legitimate traffic to the server by maintaining the TCP connection. When receiving a packet, the defense node performs the following steps. First, the defense node checks whether the destination address of the packet is

Table 1 Data structure used to identify flows

Source address 32 bits	
Destination address 32 bits	
Initial sequence number (receiver) 32 bits	
Initial sequence number (sender) 32 bits	
Source port 16 bits	Destination Port 16 bits
Timer 8bits	reserved for future use

included in the victim server list. If the destination address is not included, the defense node relays the packet as a normal IP packet after verifying that it does not hold the corresponding TCP connection. If the destination address is included (i.e., the defense node is in the defense mode for the destination address), the defense node performs the following operations.

- If the packet is a SYN packet, the defense node delegates a SYN/ACK packet and holds the connection request.
- If the packet is an ACK packet and the defense node holds connection request corresponding to the received packet, the defense node establishes the connection to the destination server via overlay network.
- If the defense node has established the connection corresponding to the received packet, the defense node relays the packet by using the established connection to the destination server.
- In other case, the defense node drops the packet.

In the rest of this subsection, we describe these operations in detail.

3.3.1 Delegating SYN/ACK packets

When the defense node receives a SYN packet to the victim server, 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.

When delegating a SYN/ACK packet, the defense node must hold the data shown in Table 1 to identify the ACK packets which is the acknowledgement of the SYN/ACK packet. The number of structures held by the defense node is equal to the number of delegating SYN/ACK packets. 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 the resources, in this paper, we use the SYN cache [6] 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. This is because the oldest entry is the most likely to be an attack packet since the legitimate SYN packets remain in the backlog queue only as long as the RTT between sending the SYN/ACK packet and receiving its acknowledgement.

3.3.2 Establishing the connection to the victim server

When receiving an ACK packet to the victim server, the defense node checks whether the corresponding connection request is held in the SYN cache. If the corresponding connection request is held, the defense node establishes the connection to the server via the overlay network, because the flow the ACK packet belongs to is identified as legitimate traffic since the ACK packet completes the 3-way handshake.

The defense node establishes two connections per flow, the connections between the defense node and the victim server and between the defense node and the source node of the flow. In our method, the defense node relays packets by connecting the two TCP connections. To connect the two connections, we use the mechanism used in TCP proxy [20], which is a method controlling transmission quality at the transport layer. TCP proxies 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. [20] shows that the TCP throughput can be improved by TCP proxies in spite of buffering delay of the proxies due to shorten RTT by dividing small TCP connections. Though the hop-by-hop TCP connections can gain more throughput rather than a single TCP connection between the same end nodes, defense nodes do not always have to establish hop-by-hop connection since our purpose of using overlay network is to distinguish the traffic identified as legitimate. Therefore, the intermediate defense nodes have only to relay the legitimate packets to the next hop defense node. Since hop-by-hop connections have both pros and cons, we use both types of TCP connections based on the administrative policy.

TCP connections are required to be held in a data structure in which we can search them quickly, because defense nodes search the corresponding TCP connections every time receiving a packet. The method in [21] can search flows quickly by using a hash table. To apply this method to our defense nodes, though, we need to adjust it. While [21] uses only (src IP, dest IP, server port) fields to identify flows, the defense nodes need to recognize connections having different client port numbers as different connections. Additionally, defense nodes need to know the corresponding connec-

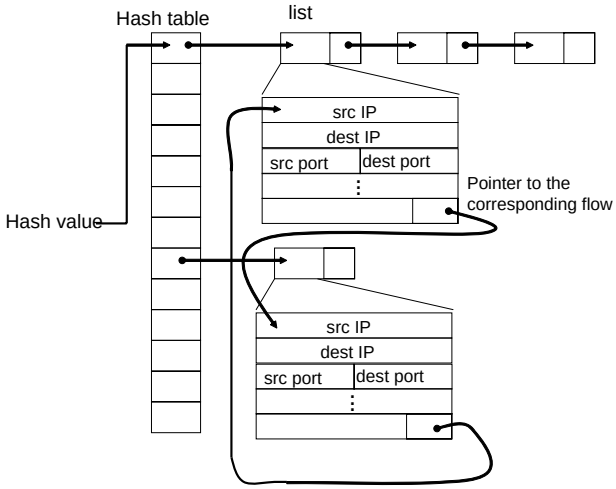


Fig. 7 Data structure to hold normal flows

tions to the next hop at the same time as searching the connections.

Figure 7 shows the data structure used in a defense node to hold legitimate connections. 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 connection. 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.

The size of data structure shown in Fig. 7 is only 240 Bytes per flow because a TCP connection can be held in a 120 Byte control block [22]. We also need sender/receiver buffers. According to [20], though larger buffers can gain more throughput, we can gain at least as much throughput as a single TCP connection between the same end nodes even if the sender/receiver buffer sizes are only 10 Kbyte. We need total 20 Kbytes per flow to hold both sender and receiver buffers.

Another important issue on establishing connections via overlay network is how to select the next hop, or how to construct the overlay topology. In SOS [14]–[16], the next hop is chosen randomly to distribute the load of traffic among overlay connections. Alternatively, we can route the legitimate traffic by using the information of latency or throughput between overlay nodes [23] or the QoS-Aware routing method [24]. However, the suitable topology and routing depend on policies of domains. Therefore, we separate methods to select the next hop or to construct overlay networks from our framework so that we can use any methods according to the situations or policies of the domains. In our evaluation described in Section 5, we use the minimum hop routing to simplify the evaluation model.

3.3.3 Relaying legitimate packets by using the established connections

If the received packet is not a SYN packet or the acknowledgement of the SYN/ACK packet held in the SYN Cache, the defense node searches the data structure shown in Fig. 7 to check whether the corresponding flow has been established. If the flow has been established, the packet is relayed by using the established TCP connection as following steps. First, the defense node receiving the packet stores the packet into the buffer of TCP proxy, and sends its ACK packet back to the source node. Then the packet is delivered by the other TCP connection to the destination node.

We also need to consider about the security of the overlay network. In this paper, we regard the flows which complete the 3-way handshake as legitimate traffic. However, a malicious user may send many packets after completing the 3-way handshake to make the overlay network overloaded and unavailable. Though random routing proposed in [14] can mitigate the intermediate links flooded by attack packets, it cannot avoid attack packets flooding the link nearest to the victim. In our method, on the other hand, the source addresses of packets on the overlay network are never spoofed because the source addresses are verified by checking the acknowledgements of SYN/ACK packets at the initial phase. For this reason, we can easily filter the flooding packets by limiting a rate from each source address. If there are too many attackers and the filtering is insufficient, we also need to limit the rate from a defense node to avoid the attack packets degrading the performance of the whole overlay network. However, because limiting the rate from a defense node affects the legitimate traffic from the node, we need to set the rate limit carefully. How to set the rate limit is one of our future works.

3.4 Extension to defend servers against other types of attacks

We have discussed how to defend servers from SYN flood attacks above. However, there are other types of attacks. For example, SYN/ACK flood attacks are attacks in which attackers generate too many SYN/ACK packets to the victim server by sending SYN packets whose source addresses are spoofed with the victim's address.

We can easily extend our method so as to defend servers from this kind of attacks by maintaining TCP connections not only to the victim servers but also from the victim servers. Maintaining TCP connection from the victim servers can be performed as following steps. First, the defense node nearest to the victim server receives a SYN packet from the victim server. The defense node relays the SYN packet to the defense node

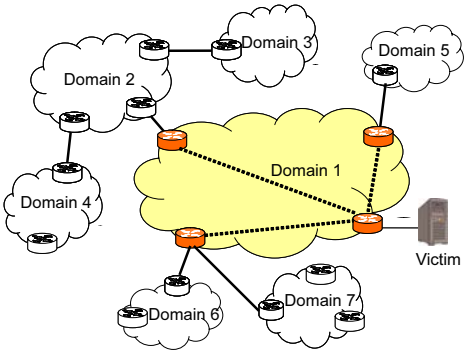


Fig. 8 First stage of deployment

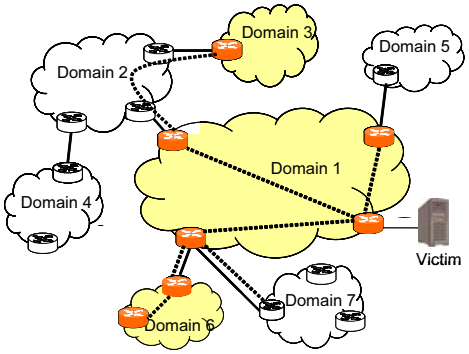


Fig. 9 Seccond stage of deployment

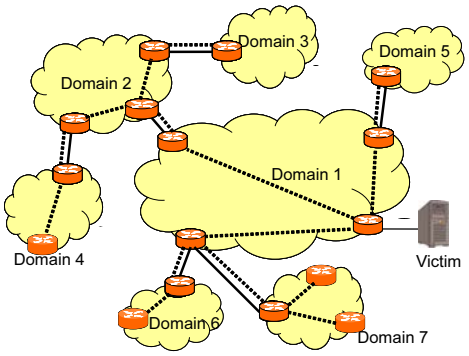


Fig. 10 Final stage of deployment

nearest to the destination node via overlay networks. Then, the defense node nearest to the destination node connects to the destination node and relays the packets between the destination node and the victim server by connecting two TCP flows in a similar way to the case described in Subsection 3.3.3.

In this case, all connections from the victim servers are relayed via overlay networks and maintained by defense nodes. That is, when a defense node receives SYN packets whose source addresses are the victim's address and the SYN packets are not relayed via overlay networks, the SYN packets are regarded as packets generated by attackers. Similarly, when the defense node receives a SYN/ACK packet to the victim server and the defense node has not received the corresponding SYN packet via the overlay network, the SYN/ACK packets are also regarded as attack packets. That is, defense nodes can easily identify attack packets and drop them. As a result, because attack packets are dropped near attackers, we can protect servers from this kind of attacks.

This way, maintaining the TCP connection at the defense nodes is also effective to other kinds of attacks using TCP.

4. Deployment Scenario

In this section, we explain how our mechanism can be deployed in the Internet. We deploy our method in a phased manner because it is impossible to deploy in the whole Internet at once. In this paper, we refer to a domain in which our mechanism is deployed as a *protected domain*. All edge routers are defense nodes in a *protected domain*. Otherwise, a domain is referred to as *unprotected*. Figures 8 through 10 show the strategic scenario for the deployment of our defense mechanism. There are three stages as follows.

1st stage (Fig.8): Only one domain is *protected*. Others are *unprotected*.
2nd stage (Fig.9): Several domains are *protected*.
Final stage (Fig.10): All domains are *protected*.

At the first stage, we consider our method to be deployed in only one domain, as shown in Fig. 8. In this figure, domain 1 is *protected*. Outside domain 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 domain 1. In the case shown in Fig. 8, our method enables domain 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 (Fig. 9), our method is deployed in several domains which cooperate with each other. In the case shown in Fig. 9, domain 1, domain 3, and domain 6 are *protected*. The protected domains do not have to be physically connected. Domains can be protected only by connecting with other protected domains logically (e.g., Domain 3 in Fig. 9). After an attack alert, the delegation of SYN/ACK packets is performed at the edge of the *protected* domains. As a result, attack traffic generated in domain 3 and domain 6 is blocked at the egress edges of these domains. Attacks from domain 2 and domain 4 are blocked at the edge of domain 1 (the defense node for the link to domain 2). Attacks from domain 5 and domain 7 are also blocked at the edge of domain 1. Increasing the number of *protected* domains means that

attack traffic is blocked at more defense nodes. Moreover, at the second stage, clients in protected domains can connect to the victim node even when attack rate is so high that clients in unprotected domains cannot connect to the victim node. For example, when attack rates from domain 2 and domain 4 are too high for the defense node at the edge of Domain 1 to deal with, legitimate clients in domain 2 and domain 4 may fail to connect to the victim. Even in this case, clients in domain 3 can connect to the victim because the packets from the domain 3 are identified by the defense node not in domain 1 but in domain 3 and are only relayed by the defense node in domain 1. As the number of *protected* domains increases, the amount of legitimate traffic that our mechanism can protect may increase.

At the final stage of deployment (Fig. 10), all domains are *protected*. In the case shown in Fig. 10, no attack packets reach domain 1 because all attack packets are blocked inside each domain. The attack traffic is no longer delivered to the core network when detected.

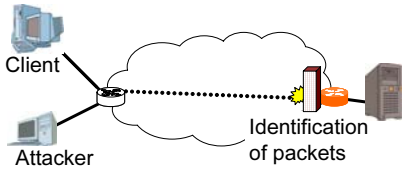
5. Evaluation

In this section, we evaluate the performance of proposed defense mechanism through simulation. First, we show the effectiveness of client-side defense by comparing the dropping rate of legitimate traffic where a single defense node is placed at the client-side with the one where the node is placed at the server-side. Then, we verify that our method can efficiently protect legitimate traffic from domains deploying our method by simulating the case where we place defense nodes at several domains (i.e., not all domains). In addition, we evaluate our method in the case of pulsing attacks. Finally, we investigate the number of TCP connections held by a defense node during the defense mode in order to evaluate the memories required to protect legitimate traffic.

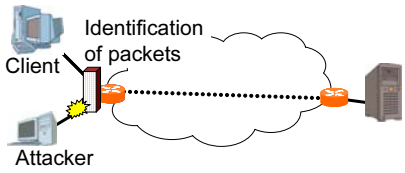
5.1 Effectiveness of client-side defense

To demonstrate the effect of identifying legitimate traffic near clients, we compared the probability of dropping legitimate SYN packets when deploying a client-side defense mechanism (Fig. 11(b)) with that when deploying a server-side defense mechanism (Fig. 11(a)). In this evaluation, the average RTT between the clients and the victim server was set to 200 ms, and the average RTT between the clients and the client-side defense node was set to 20 ms. By using the result described in [19], we generated the legitimate SYN packets whose rate follows a normal distribution with a mean of 100 SYNs/sec. We set the SYN Cache parameters to the values used in FreeBSD.

Figure 12 shows the probabilities of legitimate SYN packets being dropped based on the rate of attack traffic. As shown, the client-side defense protects



(a) Server-side defense



(b) Client-side defense

Fig. 11 Server-side defense and client side defense

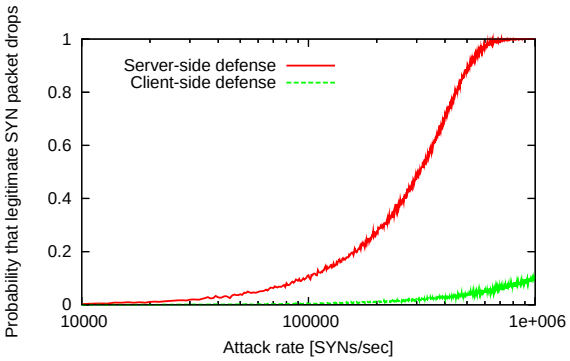


Fig. 12 Probability of dropping legitimate SYN packets vs. attack rate

legitimate packets much better than the server-side defense. This is because the RTTs between clients and the client-side defense node are much shorter than the RTTs between clients and the server-side defense node. The average holding time for each connection request on the SYN cache is also short, which increases the availability of the SYN cache.

[25] reports observing attacks whose rates exceeded 600,000 SYNs/sec. In the event of such heavy attacks, server-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 client-side defense, the probability of dropping legitimate SYN packets would be less than 0.1.

In summary, the client-side defense can catch up more than the server-side defense and can protect legitimate packets even when attack rate is large. That is, our method fulfills the R1 described in Section 1.

5.2 Effectiveness of our method to protect legitimate traffic

We next considered the effectiveness of our method

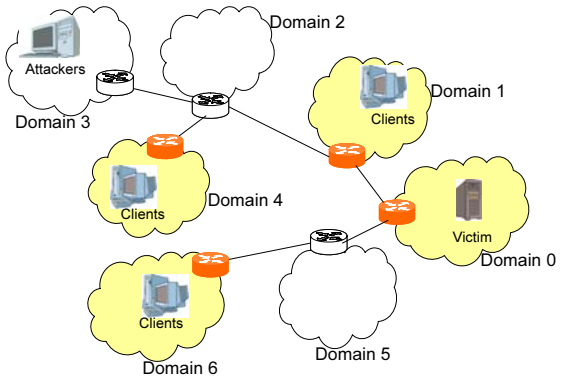


Fig. 13 Enviroment used in our simulation

when our method is deployed in the several domains (i.e., not all the domains). In this evaluation, we used the case shown in Fig. 13. In this case, Domains 0, 1, 4, and 6 deploy our methods. RTTs between the directly connected domains (e.g., Domains 0 and 1) are 80 ms. RTTs between clients in a domain and the gateway of the domain are 20 ms. That is, RTTs between clients in Domain 4 and the gateway of Domain 0 are 260 ms.

Domain 0 has a victim server and Domain 3 has attackers which inject attack packets after a certain period of time from the beginning of the simulation. The total attack rate generated by attackers in Domain 3 is 600,000 SYNs/sec, and the attack begins at 600 sec from the simulation start and ends at 1200 sec from the simulation start. 30 SYNs/sec of legitimate traffic are generated from Domains 1, 4 and 6 to the victim server. In addition, the victim server generates 3 SYNs/sec of legitimate traffic to Domain 6. We set the timeout of SYN cache to 180 sec and T_{end} to 180 sec.

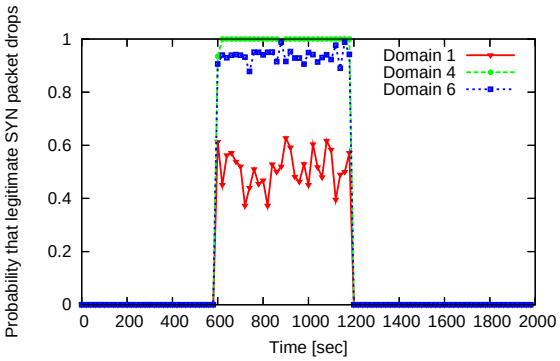
In this evaluation, we compared our method with the following two cases.

Server-side defense Only server-side defense node (i.e., the defense node deployed at the gateway of Domain 0 in Fig. 13) identifies the legitimate packets and blocks attack packets.

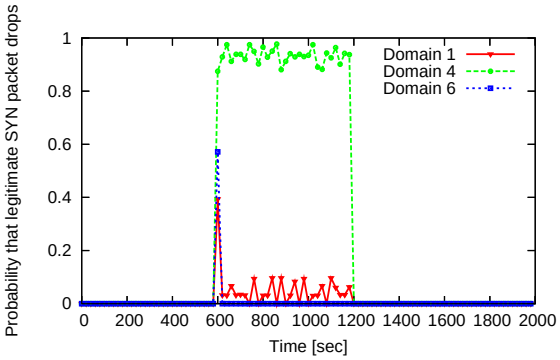
Attacker-side defense Similar to the MovingFirewall, the node nearest to the attackers blocks attack packets but other nodes perform nothing. In the case of Fig. 13, because Domains 2 and 3 do not deploy defense nodes, the defense node deployed at the gateway of Domain 1 identifies the legitimate packets and blocks attack packets.

In all cases, defense nodes identify the legitimate traffic by delegating the SYN/ACK packets.

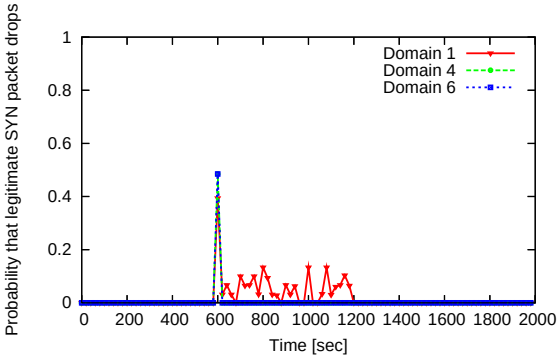
Figure 14 compares the dropping rate of legitimate traffic from Domains 1, 4 and 6. From this figure, most of the legitimate SYN packets are dropped in the case of the server-side defense. Especially, the legitimate clients in Domains 4 and 6 cannot establish the connections at all. This is because it takes long time to establish connections since the RTTs between the defense



(a) Server-side defense



(b) Attacker-side defense



(c) Our method

Fig. 14 Probability of dropping legitimate SYN packets

node at Domain 0 and the clients in Domains 4 and 6 are large. As a result, the SYN packets are dropped from the backlog queue of the defense node at Domain 0 due to a number of attack packets.

In the case of attacker-side defense, the probabilities of packet loss for Domain 1 and 6 become very low soon after the attack starts. This is because the attacks are immediately detected, and the defense node at Domain 1 begins to distinguish legitimate packets from attack packets. Then, none of the legitimate SYN packets from Domain 6 are dropped because there is no

attack traffic on the way from Domain 6 to the victim server. In addition, because the RTTs between the defense node of Domain 1 and clients in the domain are small, the probability of packet loss for Domain 1 also becomes very low.

However, the probability of packet loss for Domain 4 remains almost 1 during the attacks. This is because the legitimate packets from Domain 4 are still mixed with attack packets on the way to the defense node of Domain 1. In addition, because it takes long time to identify the packets from Domain 4 as legitimate traffic since the RTTs between the defense node at Domain 1 and clients at Domain 4 are large, the legitimate packets from Domain 4 are dropped by attack packets. That is, attacker-side defense cannot protect legitimate traffic if the legitimate traffic is mixed with attack traffic at intermediate domains not deploying defense nodes.

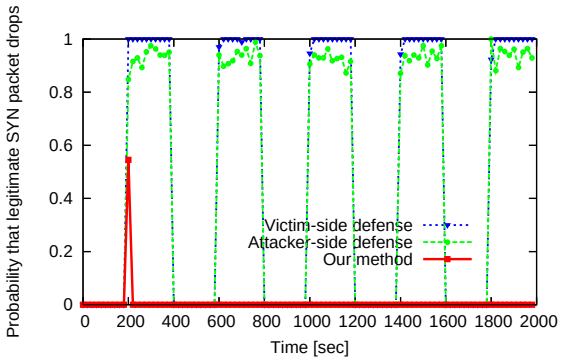
On the other hand, in the case of using our method, the probability of packet loss for Domain 4 also becomes very low soon after the attack starts. This is because the legitimate SYN packets from Domain 4 are not mixed with attack traffic since the defense node at Domain 4 begins to protect legitimate traffic by relaying them apart from other packets. That is, our method can protect the legitimate packets even if the intermediate domains (i.e., Domain 2 in this case) do not deploy our method (R2).

5.3 Effectiveness to the pulsing attack

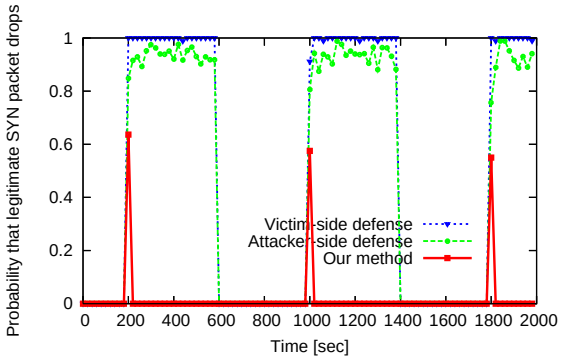
We evaluated our method in the case of pulsing attacks. In this simulation, we used the same environment as the previous subsection. We injected two types of pulsing attacks. First, attack rate changes between 0 and 600,000 SYNs/sec every 200 sec. Second, attack rate is changed between 0 and 600,000 SYNs/sec every 400 sec.

We investigated the probability of dropping legitimate SYN packets from Domain 4 for each type of attacks. Figure 15 shows the results. In this figure, we plotted the time dependent variation of the probability of dropping legitimate SYN packets in the case of our method, attacker-side defense and victim-side defense. This figure shows that, in the cases of attacker-side defense and server-side defense, clients in Domain 4 cannot connect to the victim server when the attack rate is 600,000 SYNs/sec as with the same observation in the previous subsection. On the other hand, the probability of packet loss in the case of our method becomes very low soon after the attack has occurred because defense nodes immediately move into defense mode.

When the attack rate is varied every 200 sec, the probability of packet loss in the case of our method never becomes high after the defense mode has begun. This is because defense nodes do not finish the defense mode since the interval between the end of an attack and the start of another attack is small.



(a) Attack rate changes every 200 sec



(b) Attack rate changes every 400 sec

Fig. 15 Probability of dropping legitimate SYN packets in the case of pulsing attacks

Defense nodes finish the defense mode if the number of packets which time out or are dropped has been 0 for 180 sec. In addition, attack packets remain in backlog queue until timeout if they are not dropped. The timeout is set to 180 sec. That is, the number of packets which time out does not become 0 until 180 sec after attack finished because some attack packets in backlog queue time out. Thus, defense nodes finish defense mode 360 sec after attack finished. Therefore, if an attack starts within 360 sec after another attack finished, packet loss rate remains low because defense node still remains defense mode.

On the other hand, when attack rate is changed every 400 sec, the probability of packet loss in the case of our method also becomes high every 400 sec. This is because the defense nodes end the defense mode while the attack rate is 0. However, the packets resent are not dropped because the probability of packet loss was high only for a second. For this reason, clients from Domain 4 can connect the victim server even when attack rate changes every 400 sec. This way, our method can protect legitimate packets even in the case of pulsing attacks.

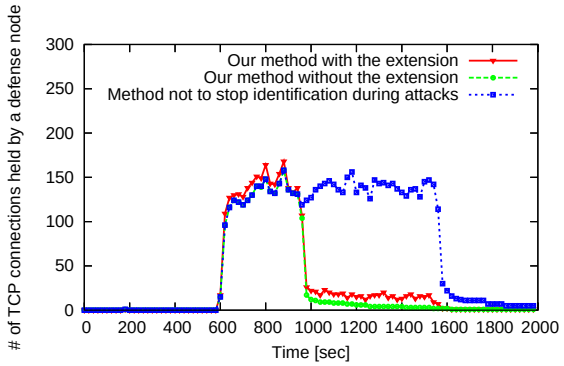


Fig. 16 Number of TCP connections held by a defense node

5.4 Loads on defense nodes

Finally, to evaluate the loads on defense nodes to hold legitimate TCP connections, we investigated the number of TCP connection held by a defense node. In this evaluation, we used the same environment used in Subsection 5.2 and the length of the legitimate TCP connections was set so as to follow the Pareto distribution whose mean is 5 sec.

Figure 16 compares the number of TCP connections held by the defense node at Domain 6 in following three cases, our method with the extension described in 3.4, our method without the extension, and a method similar to SOS [14]–[16] which does not stop the identification of legitimate traffic when at least one defense node receives attack traffic.

From this figure, the number of TCP connections held by a defense node in the cases of our method decreases at 980 sec from the simulation start, while it remains large in the case of method which does not stop the identification when at least one defense node receives attack traffic. This is because all of the attack traffic is blocked by the defense node at Domain 1 and there becomes no attack traffic on the way from Domain 6 to the victim server. Therefore, the defense node at Domain 6 ends the defense mode after verifying that the number of connection requests (i.e., SYN packets) which time out or are dropped is 0 for a given length of time. As a result, our method avoids unnecessary overhead of defense nodes, which may cause increase of the end-to-end delays, by avoiding holding the TCP connections which can connect to the server even without protection (R4).

From this figure, we can also see that the extension described in Subsection 3.4 does not require so many resources. Though the defense node in our method with the extension holds slightly more connections than in our method without the extension, the difference is small. This is because the number of connections from the victim server is much smaller than those from the clients. The servers like web servers rarely send connection requests but wait for connection requests. However, if a victim server sends many connection requests, the extension requires more resources. In such cases, we need to use smaller data structures to maintain the TCP connections from the victim server, which is one of our future works.

6. Conclusion and future work

In this paper, we have proposed a defense mechanism which can protect legitimate packets without any modification in clients from distributed SYN flood attacks. In our method, all of the TCP connections to the victim servers from a domain are maintained at the gateways of the domain (i.e., near the client). We call the nodes maintaining the TCP connection *defense nodes*. During the attacks, the defense nodes check whether arriving packets are legitimate or not by maintaining the TCP connection. That is, the defense nodes delegate reply packets to the received connection request packets and identify the legitimate packets by checking whether the clients reply to the reply packets. Then, only identified traffic are relayed via overlay networks. As a result, by deploying the defense nodes at the gateways of a domain, the legitimate packets from the domain are relayed apart from other packets including attack packets and protected.

However, there are other kinds of attacks. For example, some attackers send many packets to degrade the quality of service (e.g. delays or packet loss rate). These attacks are known as QoS attacks and cause serious impact on communication between clients and the server especially in the case of real-time application. To avoid these attacks, we need efficient filtering method or rate limiting method. Because our method can also verify that the packets on the overlay network are not spoofed while ingress filtering can only confirm that the source addresses of the packets passing the gateway are in the network, by using our method, we can easily filter attack packets of attacks other than SYN flood attacks. However, the filtering method is one of our future works.

Other future works are to identify attack packets at the points where the routes of packets may vary, to construct an effective overlay network to propagate the alert messages, and to construct smaller data structures to maintain the TCP connection between the clients and the victim server.

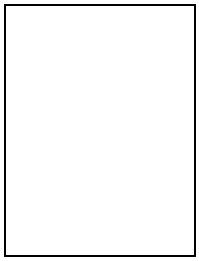
References

[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.

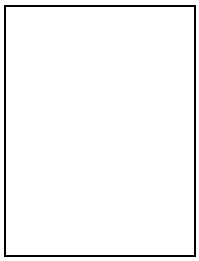
[2] "CERT advisory CA-1996-01 UDP port Denial-of-Service attack." available at `http://www.cert.org/advisories/CA-1996-01.html`.

[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.

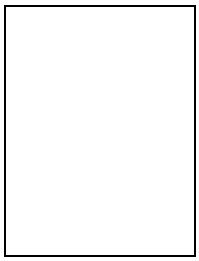
[4] "Symantec internet security threat report." available at `http://www.symantec.com/enterprise/threatreport/index.jsp`, Mar. 2005.
[5] A. Zuquete, "Improving the functionality of SYN cookies," in *Proceedings of 6th IFIP Communications and Multimedia Security Conference*, pp. 57–77, Sept. 2002.
[6] J. Lemon, "Resisting SYN flooding DoS attacks with a SYN cache," in *Proceedings of USENIX BSDCon'2002*, pp. 89–98, Feb. 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] C. Papadopoulos, R. Lindell, J. Mehringer, A. Hussain, and R. Govindan, "Cossack: Coordinated suppression of simultaneous attacks.," in *Proceedings of DISCEX III*, pp. 2–13, Apr. 2003.
[10] H. Fuji, E. Y. Chen, K. Okada, and D. Kashiwa, "Movingfirewall: A countermeasure against distributed denial of service attacks," *NTT Technical Review*, vol. 1, Aug. 2003.
[11] G. Oikonomou, P. Reiher, M. Robinson, and J. Mirkovic, "A framework for collaborative DDoS defense," in *Proceedings of the 2006 Annual Computer Security Applications Conference*, pp. 33–42, Dec. 2006.
[12] 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*, vol. 3, pp. 141–155, Mar. 2004.
[13] P. E. Ayres, H. Sun, and H. J. Chao, "ALPi: A DDoS defense system for high-speed networks," *IEEE Journal on Selected Areas in Communications*, Oct. 2006.
[14] A. D. Keromytis, V. Misra, and D. Rubenstein, "SOS: An architecture for mitigating DDoS attacks," *IEEE Journal on Selected Areas in Communications*, vol. 22, pp. 176–188, Apr. 2004.
[15] N. H. R and K. C. Sekaran, "Design and deployment of proactive models for mitigating denial-of-service and distributed dinial-of-service attacks," *International Journal of Computer Science and Network Security*, vol. 7, pp. 167–175, July 2007.
[16] A. D. Keromytis, V. Misra, and D. Rubenstein, "WebSOS: An overlay-based system for protecting web servers from denial of service attacks," *IEEE Journal on Selected Areas in Communications*, vol. 48, pp. 781–807, Aug. 2005.
[17] D. G. Andersen, "Mayday: Distributed Filtering for Internet Services," in *Proceedings of 4th USENIX Symposium on Internet Technologies and Systems*, Mar. 2003.
[18] J. Kurian and K. Sarac, "FON: A federated overlay networks for dos defense in the internet," in *Proceedings of Global Internet Symposium*, 2006.
[19] Y. Ohsita, S. Ata, and M. Murata, "Detecting distributed Denial-of-Service attacks by analyzing TCP SYN packets statistically," *IEICE Transactions on Communications*, vol. E89-B, pp. 2868–2877, Oct. 2006.
[20] I. Maki, G. hasegawa, M. Murata, and T. Murase, "Throughput analysis of TCP proxy mechanism," in *Proceedings of Australian Telecommunication Networks and Applications Conference 2004*, pp. 341–348, Dec. 2004.
[21] S. Ata, M. Murata, and H. Miyahara, "Efficient cache structures of IP routers to provide policy-based services," in *Proceedings of IEEE ICC 2001*, vol. 5, pp. 1561–1565, June 2001.
[22] W. R. Stevens and G. R. Wright, *TCP/IP Illustrated*, vol. 2. Addison-Wesley Longman Publishing Co., Inc., 1995.
[23] D. G. Andersen, H. Balakrishnan, and R. M. M. Frans Kaashoek, "Resilient overlay networks," in *Proceedings of 18th ACM Symposium on Operating Systems Principles*, pp. 131–145, Oct. 2001.
[24] Z. Li and P. Mohapatra, "Qron: Qos-aware routing in overlay networks," *IEEE Journal on Selected Areas in Communications*, vol. 22, pp. 29–40, Jan. 2004.
[25] 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.



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