Measurement, Analysis and Control to Changes of Network Traffic

Submitted to
Graduate School of Information Science and Technology
Osaka University

July 2008

Yuichi OHSITA

List of Publications

Journal Papers

1. 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, No.10, pp. 2868–2877, Oct. 2006.

2. Y. Ohsita, S. Ata, and M. Murata, "Identification of Attack Nodes from Traffic Matrix Estimation," *IEICE Transactions on Communications*, vol. E90-B, No.10, pp. 2854–2864, Oct. 2007.

3. Y. Ohsita, S. Ata, and M. Murata, "Deployable Overlay Network for Defense against Distributed SYN Flood Attacks," to appear in *IEICE Transactions on Communications* vol. E91-B, No.8, Aug. 2008.

4. Y. Ohsita, T. Miyamura, S. Arakawa, E. Oki, S. Shiomoto, and M. Murata, "Estimation of Current Traffic Matrices from Long-term Traffic Variations," submitted to *IEICE Transactions on Communications*.

5. Y. Ohsita, T. Miyamura, S. Arakawa, S. Ata, E. Oki, S. Shiomoto, and M. Murata, "Gradually Reconfiguring Virtual Network Topologies based on Estimated Traffic Matrices," submitted to *IEEE/ACM Transactions on Networking*.

Refereed Conference Papers

1. 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*, pp. 2511–2515 Nov. 2004.

2. Y. Ohsita, S. Ata, and M. Murata, "Deployable Overlay Network for Defense against Distributed SYN Flood Attacks," in *Proceedings of IEEE International Conference on Computer Communications and Networks (ICCCN 2005)*, pp. 407–412, Oct. 2005.

3. Y. Ohsita, S. Ata, and M. Murata, "Identification of Attack Nodes from Traffic Matrix Estimation," in *Proceedings of 4th International Trusted Internet Workshop*, Dec. 2005.

4. Y. Ohsita, T. Miyamura, S. Arakawa, E. Oki, S. Shiomoto, and M. Murata, "Estimating Current Traffic Matrices Accurately by Using Long-term Variations Information," to be presented at *Broadnets 2008*, Sept. 2008.

Non-Refereed Technical Papers

1. Y. Ohsita, S. Ata, and M. Murata, "Detecting Distributed Denial-of-Service Attacks by Analyzing TCP SYN Packets Statistically," *Technical Reports of IEICE(IN2003-201)*, pp. 23–28, Feb. 2004 (*in Japanese*).

2. Y. Ohsita, S. Ata, and M. Murata, "Deployable Overlay Network for Defense against Distributed SYN Flood Attacks," *Technical Reports of IEICE(IN2004-125)*, pp. 13–18, Dec. 2004 (*in Japanese*).

3. Y. Ohsita, S. Ata, and M. Murata, "Traffic Matrix Estimation for Identification of Attack Sources," *IEICE Society Conference*, BS-3-1, Sept. 2005 (*in Japanese*).

4. Y. Ohsita, S. Ata, and M. Murata, "Identification of Attack Nodes from Traffic Matrix Estimation," *Technical Reports of IEICE(NS2005-86, IN2005-86, CS2005-32)*, Sept. 2005 (*in Japanese*).

5. Y. Ohsita, T. Miyamura, S. Arakawa, S. Ata, E. Oki, S. Shiomoto, and M. Murata, "Increasing the Accuracy of Traffic Matrix Estimation for Gradual Reconfiguration of Virtual Network Topologies," *Technical Reports of IEICE(PN2006-90)*, pp. 37–40, Mar. 2007 (*in Japanese*).

Preface

Network operators design their networks according to the predicted traffic so as to accommodate all traffic efficiently (e.g., without congestion or large delays). However, if the current traffic significantly differ from the predicted one, the previously constructed network becomes no longer suitable to the current traffic; for example, it may happen that utilizations of some links are extremely high and cause congestion or large delays. Similarly, if a server receives unexpectedly many requests, the server cannot respond to the requests.

Thus, when the current traffic becomes significantly different from the predicted one, we need to handle the changes of traffic so as not to degrade the performances of the network. Such significant changes of traffic are caused either by the malicious traffic or by the increases of legitimate traffic. In this thesis, we propose methods to handle both cases.

One of the typical malicious traffic is traffic of denial-of-service (DoS) attacks in which attackers send a very large number of packets to prevent users from communicating with service providers. Especially, in the distributed denial-of-service (DDoS) attacks often seen recently, multiple distributed nodes attack a single server concurrently. Even if the rate of attack from each node is small, the attack traffic can cause serious damage at the victim server when the number of hijacked nodes is large. Therefore, if the significant changes of traffic are caused by the malicious traffic, we need to identify and block attack packets immediately and protect the legitimate traffic so as to keep legitimate users able to connect the services. Against this kind of attacks, we propose the following methods.

First, we propose a method to detect attacks. In our detection method, we focus on SYN flood attacks which are the most frequent attacks among DDoS attacks. One of the problems in detecting SYN flood traffic is that server nodes or firewalls cannot distinguish the SYN packets of normal TCP connections from those of SYN flood attack. Moreover, since the rate of normal network traffic may vary, we cannot use an explicit threshold of SYN arrival rates to detect SYN flood traffic. Thus, we investigate the difference of statistics of arrival rates of normal TCP SYN packets and SYN flood

attack packets by using the traffic data monitored at the gateway of our university. According to the results, the arrival rate of normal TCP SYN packets can be modeled by a normal distribution while the arrival rate of TCP SYN packets when attack starts is far from a normal distribution. Based on the analytical results, our detection method detects attacks by checking the difference between the sampled SYN rates and the normal distribution. The simulation results show that our method can detect attacks whose rates are even lower than 20 SYNs/sec. In addition, the results also show that our method can detect attacks faster than the existing detection method.

Then, we propose a method to identify attack nodes which can work with legacy routers unlike the traditional traceback methods. In our identification method, we identify the egress routers that attack nodes are connecting to by estimating the traffic matrix between arbitral source-destination edge pairs from the traffic volumes of each link which can be monitored by legacy routers. By monitoring the traffic variations obtained by the traffic matrix, we identify the edge routers that are forwarding the attack traffic, which have a sharp traffic increase to the victim. According to the simulation results, even when we can monitor only the link loads, our method can identify attack sources accurately and limit the total attack rate from unidentified attack sources by setting parameters adequately.

Finally, we propose a method to protect legitimate traffic. Our protection method also focuses on SYN flood attacks. In our protection 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 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. According to our simulation results, our method can make the probability of dropping legitimate SYN packets less than 0.1 even when the attack rate exceeds 600,000 SYNs/sec.

On the other hand, if the significant changes of traffic are caused by the increases of legitimate traffic, we should not block any traffic unlike the case of the malicious traffic. Thus, we reconfigure the network settings (e.g., routes or logical topologies) so as to accommodate all current traffic efficiently. This type of activity is often called *traffic engineering* (TE). In TE methods, new network settings are configured according to a traffic matrix, which indicates traffic volumes between all pairs of edge nodes, so that constraints such as maximum link utilization are satisfied. However, it

is difficult to monitor traffic matrices directly. Therefore, we need to estimate traffic matrices from the information which can be monitored more easily than directly monitoring traffic matrices and then perform TE methods using the estimated traffic matrices.

One of the important issues about using the estimated traffic matrices is estimation errors which may degrade the performances of TE methods. For example, due to estimation errors, TE method may not mitigate the congestion. Thus, we propose methods that reduce estimation errors during performing TE methods.

First, we propose a gradual reconfiguration method in which the reconfiguration of network settings is divided into multiple stages instead of reconfiguring the suitable settings at once. By dividing the reconfiguration into multiple stages and assuming that no or few elements of the true traffic matrix change significantly throughout the TE method execution, we can calibrate and reduce the estimation errors in each stage by using information monitored in prior stages. We evaluate the effectiveness of the gradual reconfiguration by simulation. According to the results, our gradual reconfiguration can reduce the root mean squared relative error (RMSRE) to 0.1 and achieve adequate network settings as is the case with the reconfiguration using the actual traffic matrices.

However, when it takes a long time to achieve the sufficient network settings, the current traffic can differ from the initial traffic monitored before the beginning of the reconfiguration. This violates the fundamental assumption of the above method to reduce estimation errors. Therefore, we also propose a new estimation method, with which we can accurately estimate current traffic matrices even when it takes a long time to achieve the sufficient network settings. In this method, we first estimate the long-term variations of traffic by using the link loads monitored the last M times. Then, we adjust the estimated long-term variations so as to fit the current link loads. In addition, when the traffic variation trends change and the estimated long-term variations cannot match the current traffic, our method detects mismatches between the estimated long-term variations and the current traffic. Then, our method re-estimates the long-term variations after removing information about the end-to-end traffic causing the mismatches, so as to capture the current traffic variations. According to our simulation results, our estimation method can estimate current traffic matrices accurately without RMSRE larger than 0.1 even when traffic changes significantly.

Acknowledgements

First of all, I would like to express my sincere gratitude to Professor Masayuki Murata of Graduate School of Information Science and Technology, Osaka University, under whose direction the studies described in this thesis have been carried out. I also thank to Professor Koso Murakami, Professor Makoto Imase, and Professor Teruo Higashino of Graduate School of Information Science and Technology, Osaka University, and Professor Hirotaka Nakano of Cyber Media Center, Osaka University, for reviewing this thesis.

This work would not have been possible without Associate Professor Shingo Ata of Graduate School of Engineering, Osaka City University and Assistant Professor Shin'ichi Arakawa of Graduate School of Information Science and Technology, Osaka University. Their critical comments and unerring guidance are always informative and helpful. I'm also indebted to Associate Professor Naoki Wakamiya of Graduate School of Information Science and Technology, Osaka University for giving me helpful comments.

I would also like to thank Professor Naoaki Yamanaka of Keio University, Associate Professor Eiji Oki of the University of Electro-Communications, Dr. Kohei Shiomoto and Mr. Takashi Miyamura of NTT Corporation, for their useful suggestions and fruitful discussion.

I am thankful to all the members of Advanced Network Architecture Laboratory in Graduate School of Information Science and Technology, Osaka University, for their detailed and valuable instructions.

I want to heartily thank my parents for their endless love and invaluable support. Finally, I am deeply grateful to my wife, Masako for her support. She has always supported and encouraged me to do my best.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Section 1 Background and Overview

1.1 Background

Network operators design their networks according to the predicted traffic so as to accommodate all traffic efficiently (e.g., without congestion or large delays). However, if the current traffic significantly differ from the predicted one, the previously constructed network becomes no longer suitable to the current traffic; for example, it may happen that utilizations of some links are extremely high and cause congestion or large delays. Similarly, if a server receives unexpectedly many requests, the server cannot respond to the requests.

Thus, when the current traffic becomes significantly different from the predicted one, we need to handle the changes of traffic so as not to degrade the performances of the network. Such significant changes of traffic are caused either by the malicious traffic or by the increases of legitimate traffic. In this thesis, we discuss methods to handle both cases.

One of the typical malicious traffic is traffic of denial-of-service (DoS) attacks in which attackers send a very large number of packets to prevent users from communicating with service providers. Especially, in the distributed denial-of-service (DDoS) attack often seen recently, multiple distributed nodes attack a single server concurrently. A malicious user tries to hack remote nodes by exploiting the vulnerabilities of software running on them, installs an attacking program on hijacked nodes, and keeps them waiting for an order to attack a victim server. When the malicious user sends a signal to them, they begin to attack to the same server. Even if the rate of attack from each node is small, the attack traffic can cause serious damage at the victim server when the number of hijacked nodes is large. Therefore, if the significant changes of traffic are caused by the malicious traffic, we need to identify and block attack packets immediately and protect the legitimate traffic so as to keep legitimate users able to connect the services.

On the other hand, if the significant changes of traffic are caused by the increases of legitimate traffic, we should not block any traffic unlike the case of the malicious traffic. Thus, we reconfigure the network settings (e.g., routes or logical topologies) so as to accommodate all traffic efficiently. This type of activity is often called *traffic engineering* (TE). In TE methods, new network settings are configured according to a traffic matrix, which indicates traffic volumes between all pairs of edge nodes, so that constraints such as maximum link utilization are satisfied.

One approach to obtain the traffic matrix directly is to construct fully meshed label switched paths using MPLS. However, this approach does not scale because it requires N-square number of label switched paths. Another approach is to tally the number of packets of each end-to-end

traffic flow at all the edge nodes. However, this is also difficult to apply in large-scale networks, because tallying the number requires a non-negligible amount of CPU resources of edge nodes, and gathering the tallied data of all end-to-end traffic consumes a non-negligible amount of network resources such as bandwidths.

Therefore, several methods to estimate traffic matrices have been proposed [1–10]. In such methods, a whole traffic matrix is estimated by using the information (e.g., link utilizations) that can be collected much more easily than directly monitoring traffic matrices. However, according to Ref. [11], if we use the estimated traffic matrices as an input of a TE method, estimation errors in traffic matrices have large impacts on the performance of the TE methods and may cause the significant large link utilizations. Thus, we need a TE method which can properly work with estimated traffic matrices by cooperating with traffic matrix estimation methods.

1.2 Outline of Thesis

As discussed above, so as not to degrade the performances of networks even when network traffic changes significantly, we need to handle the following two kinds of cases; the cases of (1) attacks and (2) increases of legitimate traffic. Against attacks, we propose methods which can detect attacks immediately, identify the sources of the attacks accurately and protect the legitimate connections accurately. On the other hand, against the increases of legitimate traffic, we propose methods which estimate current traffic matrices and reconfigure the adequate network settings by cooperating each other.

Detection, Identification and Defense against Denial-of-Servce Attacks

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.

There are many kinds of DDoS attacks such as Smurf attacks [12], UDP floods [13], and SYN flood attacks [14]. In Smurf and UDP attacks, attackers generate many ICMP or UDP packets to exhaust the capacity of the victim's network link. In SYN flood attacks, attackers send so many connection requests to one victim server that users cannot connect to that server. Because attackers can easily put servers into a denial-of-service state this way, about 90% of all DoS attacks are SYN flood attacks [15]. In addition, according to Ref. [16], recently many bot networks use SYN flood

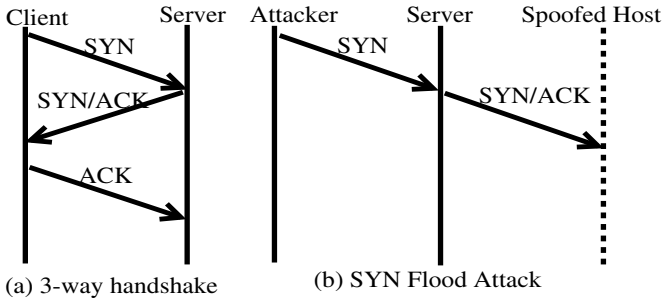


Figure 1.1: Overview of a 3-way handshake and a SYN flood attack

attacks to shut down target servers. As a result, the number of SYN flood attacks is still increasing.

SYN flood attacks exploit the TCP (Transmission Control Protocol) specification. In the TCP, a client node communicates with a remote node (i.e., server) by way of a virtual connection established by a process called a 3-way handshake. As shown in Figure 1.1(a), a client first sends a server a SYN requesting to establish a connection. Then the server sends the client a SYN/ACK packet acknowledging receipt of the SYN packet. When the client receives the SYN/ACK packet, the client sends the server an ACK packet acknowledging receipt of the SYN/ACK packet and begins to transfer data.

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

Figure 1.1(b) shows an overview of a SYN flood attack. Attackers send SYN packets whose source address fields are spoofed. The server receiving these SYN packets sends the SYN/ACK packets to spoofed addresses. If the node having the spoofed address actually exists, it sends a RST packet for the SYN/ACK packet because it didn't send the SYN packet. If there is no host having the spoofed address, however, the SYN/ACK packet is discarded by the network and the server waits in vain for an ACK packet acknowledging it. For losses of SYN/ACK packets, the server has a timer in the backlog queue, and half-open states exceeding the timer are removed. When the backlog queue is filled with spoofed SYN packets, however, the server cannot accept SYN packets

from users trying to connect to the server.

To defend servers and networks against these kinds of attacks, we propose methods which can detect attacks immediately, identify the sources of the attacks accurately and protect the legitimate connections accurately in Chapter 2.

Detection of Distributed Denial-of-Service Attacks by Analyzing TCP SYN Packets Statistically [17–19] To defend servers and networks against DDoS attacks, we first need to detect attacks. In this paragraph, we discuss how to detect attacks especially focusing on SYN flood attacks which are the most frequent attacks among DDoS attacks.

It is difficult to distinguish the packets of SYN flood attacks from normal TCP SYN packets at the victim server because the packets of the attacks do not differ from normal TCP SYN packets except in the spoofing of the source addresses. In addition, since the rate of normal network traffic may vary, we cannot use an explicit threshold of SYN arrival rates to detect SYN flood traffic. This is why SYN flood attacks are hard to detect.

Therefore, several methods for detecting this kind of attacks have been proposed, and one is to detect the mismatch between bidirectional packets [20]. When a server is not under attack, packet arrival rates for both directions are almost the same or at least of the same order, because the TCP needs an ACK packet for each packet that is sent. If the packet arrival rate for one direction is much higher than that for the other direction, the traffic in the high-rate direction might include some attack packets. In this mechanism, however, the router cannot detect the attack until the server can reply SYN/ACK packets for spoofed SYN packets. MULTOPS [21] is one of similar versions which checks asymmetries of traffics for both directions with the granularity of subnets. Another method proposed in Ref. [22] is to use the difference between the rates of SYN packets (i.e., the head of the connection) and FIN/RST packets (i.e., tail of the connection). If the rate of SYN packets is much higher than that of FIN or RST packets, the router recognizes that the attacking traffic is mixed into the current traffic. The method proposed in Ref. [23] detects attacks by the number of source addresses. If the number of source addresses increase rapidly, the current traffic might include attack packets.

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

of any variance of normal traffic, but require long time. Attack traffic should be identified more accurately and faster by considering the variance of normal traffic.

Therefore, in Section 2, we propose a method that detects attacks more quickly and more accurately by taking the time-of-day variance of traffic into consideration. For this purpose, we first collect all packets passing through the gateway of our university, and analyze the statistical characteristics of both normal and attack traffics. According to the analytical results, the arrival rate of normal TCP SYN packets can be modeled by a normal distribution. Then, we propose a new detection method based on the analytical results. Finally, we conduct trace-driven simulations. According to the simulation results, our detection method can detect attacks whose rates are even lower than 20 SYNs/sec. In addition, the results show that our method can detect attacks faster than the existing detection method.

Identification of Attack Nodes from Traffic Matrix Estimation [24–27] Identifying the attack sources is one of effective (and ideal) solutions to cut off the link to the attacker or filter attack packets by an edge router connected to the attacker. However, because attackers can easily spoof the source address fields of the attack packets, it is quite hard to identify the attack sources by only checking the source address of the attack packets.

For this reason, several methods for identifying the attack sources are proposed. In general, these methods are called *IP tracebacks*. One of them is proposed in Ref. [28], where a router generates an ICMP traceback packet to the destination of the packet with a low probability at the event of packet forwardings. The victim identifies the actual source of the packet by using the received ICMP traceback packets. Other methods are proposed in Ref. [29–31], in which a router marks IP packets to be forwarded with identification information instead of generating ICMP packets. The victim can identify the source of the packets using the identification information. Another method is proposed in Ref. [32,33], where each router stores packet digests. The victim queries its upstream routers to see whether an attack packet has passed through them or not.

However, these methods have several problems. One of them is that they cannot work with legacy routers because they need router support. Another is that they may erroneously identify legitimate clients as attack sources. This is because these methods can only identify the source nodes of IP packets. Since there is no difference between legitimate and attack packets, identifying attack packets from the mixture of attack and legitimate traffic is difficult.

In DoS attacks, attackers send a large number of packets to exhaust the network resources. When an attack starts, there is a rapid increase in the traffic from the attack sources to the victim.

Several methods use such increase in traffic in the network to detect attacks [34–37]. By using traffic volumes which can also be monitored by legacy routers, we identify edge routers connecting to the attackers without any change in core network. Then, deploying only edge routers supporting IP traceback, we identify attack nodes by using IP traceback from the identified edge routers. In addition, identification of the attack sources by monitoring the increased traffic can distinguish the attackers from the legitimate clients, which do not sharply increase traffic. However, there are only a few papers about identification of attack sources by monitoring the increase in traffic.

Lakhina et al propose a method for identifying the attack sources by monitoring the traffic on each link in the network [35]. In this method, the measured loads of all links are separated into normal and abnormal subspaces. The normal subspace indicates the time-of-day variation of the traffic. Other variations are categorized into the abnormal subspace. We then identify the attack source that explains the largest amount of anomalous subspace. Although this method can identify the attack source in a single attacker case, this method has difficulty in identifying multiple attack sources such as DDoS, because we need to consider all possible cases by changing the number of attackers. It requires a huge computation overhead.

The anomaly detection methods using traffic volumes between all source/destination pairs are also proposed. The traffic volumes between every pair of ingress and egress points are typically described as a traffic matrix. The method proposed in Ref. [37] uses the compact summary of the per-flow statistics and detect anomalies by comparing the difference between the actual summary and the forecasted summary obtained by the forecast models. Another method proposed in Ref. [36] separates the traffic matrix into normal and abnormal subspaces. Since this method separate traffic volumes between all source/destination pairs into normal and abnormal subspaces, we can identify traffic between source/destination pairs having large abnormal subspaces as attack traffic.

However, these methods assume that the traffic matrix can be monitored accurately. Though Cisco's NetFlow [38] can monitor the flow statistics and periodically export the monitored statistics to the central storage device, the process of NetFlow in routers consumes CPU time to identify flows of received packets. The performance analysis of NetFlow [39] says the resource consumption would increase according to the number of active flows passing the router. Especially, DDoS attack traffic contains many of spoofed packets which lead the large number of flows having a single packet. As a result, during DDoS attacks, the activation of NetFlow has possibility to consume very large amount of CPU time. Though Ref. [40] proposes the distributed method to monitor traffic data and separate them into normal and abnormal subspaces, this method needs router support and cannot work with legacy routers. On the contrary, to avoid the high loads on routers, the objective

of our identification method is to identify attack sources accurately under the assumption that we cannot monitor the amount of traffic for all edge pairs. Therefore, we only measure the load of network links.

Estimating traffic matrix from the measurement of link loads is also proposed in some literatures e.g., Ref. [6], however, existing traffic matrix estimation methods are not suitable to apply the identification of attacks because the assumption used in these methods may decrease the accuracy of estimation of traffic volumes including the attack traffic.

Therefore, in Section 3, we propose a new method for identifying attack sources by estimating the increase in traffic between each source and destination. In this method, we can estimate the increase in traffic accurately by focusing not on the total rate of traffic but on the increase in traffic. In addition, our method can work with existing routers because we can obtain link load data through Simple Network Management Protocol (SNMP). We also evaluate the effectiveness of our method through simulation. According to the results, even when we can monitor only the link loads, our method can identify attack sources accurately and limit the total attack rate from unidentified attack sources by setting parameters adequately.

Overlay Network Against Distributed SYN Flood Attacks [41–43] After the detection of attacks, we need a defense method which can protect legitimate traffic so that end users can connect the target servers during attacks. We discuss the defense method, especially focusing on SYN flood attacks which are the most frequent attacks among DDoS attacks. Prior to mention about the defense methods, 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 DDoS attacks have been proposed so far. SYN cookie [44] and SYN cache [45] 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 [20] 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 [46], 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.

DefCOM [47] proposes another framework to mark suspicious packets at edge nodes and limit the rates for the suspicious packets at core nodes. PacketScore [48] and ALPi [49] are similar methods. In these methods, 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.

Another framework is proposed in Ref. [50]. 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 [51] 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.

The frameworks using the overlay network to protect legitimate traffic have also been proposed as SOS [52–54], Mayday [55] or FON [56]. WebSOS [54] 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. 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.

As described above, none of the existing methods fulfill all of four requirements described above. Thus, in Section 4, we propose a defense method which fulfills all of the requirements.

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 eliminating 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 mechanisms 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 defense 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.

We also conduct simulations to evaluate our defense method. According to our simulation results, our method can make the probability of dropping legitimate SYN packets less than 0.1 even when the attack rate exceeds 600,000 SYNs/sec.

Measurement, Estimation and Topology Control to Changes of Traffic

Network operators design their networks according to the predicted traffic so as to accommodate all traffic efficiently (e.g., without congestion or large delays). However, if the current traffic significantly differ from the predicted one due to the changes of legitimate traffic, the previously constructed network becomes no longer suitable to the current traffic. Therefore, in this case, we need to reconfigure the networks settings by TE methods so as not to degrade the performances of the network. However, as discussed above, TE methods require the traffic matrices which are hard to monitor directly. Therefore, we propose methods which estimate traffic matrices accurately and reconfigure the adequate network settings by cooperating each other in Chapter 3.

Gradual Reconfiguration of Virtual Network Topology [57, 58] One of efficient traffic engineering methods to accommodate traffic that changes unpredictably is optical layer traffic engineering (TE) [59–66]. Optical layer TE assumes that a network consists of IP routers and optical cross-connects (OXCs), as illustrated in Fig. 1.2. Each outbound port of an edge IP router is connected to an OXC port. Lightpaths (hereafter called optical layer paths) are established between two IP routers by configuring OXCs along the route between the routers. A set of optical layer paths forms a VNT (virtual network topology). Traffic between two routers is carried over the VNT using IP layer routing. In these conditions, optical layer TE accommodates time-varying traffic by dynamically reconfiguring VNTs.

Several methods to reconfigure the VNT have been proposed. They can be categorized as either full [59, 60] or partial reconfiguration algorithms [61–66]. In full reconfiguration, the new VNT is computed with no limitation on the number of reconfigured optical layer paths. For example, Mukherjee *et al.* [59] formulated the full reconfiguration VNT design problems as optimization problems and proposed heuristic algorithms based on the simulated annealing approach that find nearly optimal solutions. The full reconfiguration approach can yield a solution that is optimal in

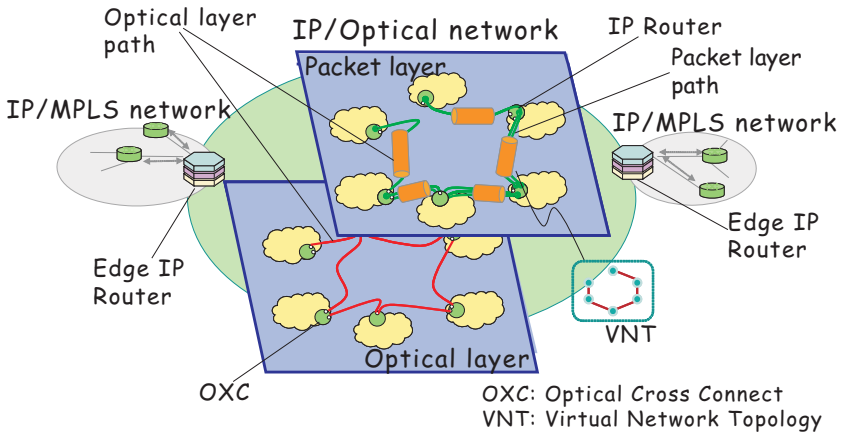


Figure 1.2: IP/Optical network.

terms of network performance such as maximum utilization of optical layer paths when using the actual traffic matrix. On the other hand, in partial reconfiguration [61–66], the new VNT is computed with limitations on the number of reconfigured optical layer paths. Banerjee and Mukherjee [61] present an integer linear programming (ILP) formulation for optimal VNT design. Their approach assumes that the future traffic matrix is given; the future VNT is determined by adapting the current one to accommodate the change in the traffic matrices. Gieselman *et al.* [66] proposed a heuristic reconfiguration algorithm that efficiently adapts to fluctuations in traffic. In the partial reconfiguration method, we can construct the adequate VNT by changing only a small number of optical layer paths.

However, as described above, both algorithms require a traffic matrix as an input and it is difficult to monitor traffic matrices directly. Therefore, several methods to estimate traffic matrices have been proposed. Zhang *et al.* [6] have proposed a estimation method called the tomogravity method. Tomogravity method first obtains the initial traffic matrices by using a gravity model that assumes that the amount of traffic from a source to a destination node is proportional to the total of the incoming/outgoing traffic for each edge node. Then, the method estimates the current traffic matrix by adjusting the initial traffic matrices so as to fit all the monitored link loads. Results reported by Refs. [6] and [67] show that tomogravity can follow rapid changes in amounts of traffic, and can estimate the traffic matrix on a tier-1 ISP network within 5 seconds. Recently, a method to estimate traffic matrices so as to follow the gravity model even in the case without complete observation of the edge link loads has been proposed [7].

Another type of estimation methods [2, 3] has also been proposed; they assume that edge-to-edge traffic demand follows a Gaussian distribution. Additionally, Tan *et al.* [4, 5] have proposed

methods to estimate traffic matrix which fits the monitored link loads and is nearest to the initial traffic matrix obtained by assuming the Gaussian distribution.

However, the estimated traffic matrices include estimation errors due to the differences between the actual traffic and these models (i.e., gravity model or Gaussian distribution) as shown in the results described in Ref. [68] and these estimation errors have impacts on a TE method.

One way of handling such estimation errors is to reconfigure the VNT redundantly, taking the estimation errors into consideration. However, if the estimation errors are large, the redundant reconfiguration may require an unacceptable amount of resources such as wavelengths. To avoid the impact of estimation errors, therefore, reduction of estimation errors is necessary.

One approach to increase the estimation accuracy is to use additional information. Refs. [8] and [9] obtain additional information by measuring a part of end-to-end traffic. However, in a large network, we may not measure enough number of end-to-end traffic to estimate the traffic matrices accurately. Another method to obtain the additional information has been proposed by Soule *et al.* [10]. In this method, additional information is obtained by changing the routes of packet layer paths and observing the difference between utilization of links before and after routes are changed. However, this method requires changes in packet layer paths that are initially unnecessary.

Therefore, in Section 5, we propose a method to reduce the estimation errors while reconfiguring the VNT. In this method, the VNT reconfiguration is divided into multiple stages instead of reconfiguring the suitable VNT at once. By dividing the VNT reconfiguration into multiple stages and assuming that no or only few elements of the true traffic matrix change significantly throughout the TE method execution, our traffic matrix estimation method calibrates and reduces estimation errors at each stage by using information (e.g., packet layer routing and utilizations of optical layer paths) monitored at prior stages with no unnecessary route changes. By reducing the estimation errors, our method can achieve the sufficient VNT (e.g., making the link utilizations less than a threshold) without directly monitoring traffic matrices as is the case with the reconfiguration using the actual traffic matrices. We evaluate the effectiveness of the gradual reconfiguration by simulation. According to the results, the gradual reconfiguration can reduce the root mean squared relative error (RMSRE) to 0.1 and achieve an adequate VNT.

Estimation of Current Traffic Matrices from Long-term Traffic Variations [69,70] The method proposed in Section 5 obtains additional measurements by using the route changes caused by a TE method. By performing TE a sufficient number of times, this approach obtains a sufficient number of measurements and then estimates the traffic matrix by assuming that no or only few elements

of the true traffic matrix change significantly throughout the TE method execution. However, if it takes a long time to change routes sufficient times, the current traffic can differ from the initial traffic monitored before the first route change. This violates the fundamental assumption of the above method. Therefore, we need a traffic matrix estimation method that considers the time variations of traffic matrices. Ref. [10] proposes a method for modeling traffic variations by using periodic functions and estimates these functions' parameters. However, when traffic changes unpredictably, the traffic matrices estimated by this approach cannot fit the current traffic matrices since it can only estimate the average variations of traffic for a period of a day by monitoring link loads for several days. As a result, a TE method cannot configure routes suitable for the current traffic.

Therefore, in Section 6, we also propose a new estimation method, with which we can accurately estimate current traffic even when traffic changes. Unlike in Ref. [10], the purpose of our method is to estimate not the long-term variations of traffic but the current traffic matrix, which consists of both long-term variations and short-term variations. By using the accurate traffic matrix, a TE method can properly work to configure routes suitable for the current traffic.

In this method, we first estimate the long-term variations of traffic by using the link loads monitored the last M times. Then, we adjust the estimated long-term variations so as to fit the current link loads. In addition, when the traffic variation trends change and the estimated long-term variations cannot match the current traffic, our method detects mismatches between the estimated long-term variations and the current traffic. Then, our method re-estimates the long-term variations after removing information about the end-to-end traffic causing the mismatches, so as to capture the current traffic variations. According to our simulation results, our estimation method can estimate current traffic matrices accurately without RMSRE larger than 0.1 even when traffic changes significantly.

Chapter 2

Detection, Identification and Defense against Denial-of-Service Attacks

The recent rapid growth and the increasing utility of the Internet are making security issues increasingly important. Denial-of-service (DoS) attacks are one of the most serious problems and must be resolved as soon as possible. These attacks prevent users from communicating with service providers and have damaged many major web sites all over the world. The number of attacks has been increasing, and the techniques used to attack servers have become more complex. In the distributed denial-of-service (DDoS) attacks often seen recently, multiple distributed nodes attack a single server concurrently. A malicious user tries to hack the remote nodes by exploiting the vulnerabilities of the software running on them, installs an attack program on the 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 to the victim server when the number of hijacked nodes is large.

To defend servers and networks against these attacks, in this chapter we propose methods which can detect attacks immediately, identify the sources of the attacks accurately and protect the legitimate connections accurately.

Section 2 Detection of Distributed Denial-of-Service Attacks by Analyzing TCP SYN Packets Statistically

To defend servers against DDoS attacks, we need to detect attacks immediately and accurately. One of the problems in detecting attack traffic is that server nodes or firewalls cannot distinguish the packets of normal traffic from those of attack traffic. Moreover, since the rate of normal network traffic may vary, we cannot use an explicit threshold of packet arrival rates to detect attack traffic.

In this section, we focus on SYN flood attacks which are the most frequent attacks among DDoS attacks and introduce a mechanism for detecting SYN flood traffic more accurately by taking into consideration the time variation of arrival traffic. For this purpose, we first investigate the statistics of the arrival rates of both normal TCP SYN packets and SYN flood attack packets. According to the analytical results, the arrival rate of normal TCP SYN packets can be modeled by a normal distribution. Then, we propose a new detection method based on the analytical results. According to the simulation results, our detection method can detect attacks whose rates are even lower than 20 SYNs/sec. In addition, the results show that our method can detect attacks faster than the existing detection method.

2.1 Statistical Analysis of Traffic and Attack Detection Method

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

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

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

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

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

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

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

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

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*.

Time-dependent variation of normal traffic and its statistical modeling

In the work shown in this section, 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.

Table 2.1: 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

The numbers of flows classified into each of the five groups are listed in Table 2.1. These values were obtained using 180 seconds as the timeout. That is, if there are more than 180 seconds after the last packet in of the flow, we considered the flow to be terminated.

The time-dependent variations of SYN arrival rates of all flows, the flows in *normal traffic* and the flows in *incomplete traffic* are shown in Figures 2.1. Points where the arrival rate rises sharply (e.g., 28,000 sec and 57,000 sec) seem to be due to *incomplete traffic*. These results also show that we would mistakenly identify many points as attacks if we set a single threshold for the SYN arrival rates because the arrival rates of the normal traffic change over time. We can also see that the distribution of SYN arrival rates seems to be different in *incomplete traffic* from in the *normal traffic* especially at the tail.

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

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

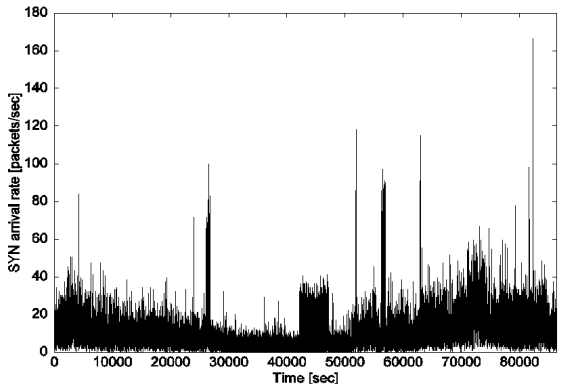
$$F(x) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}\sigma} exp[\frac{-(y-\zeta)^2}{2\sigma^2}]dy. \tag{2.1}$$

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

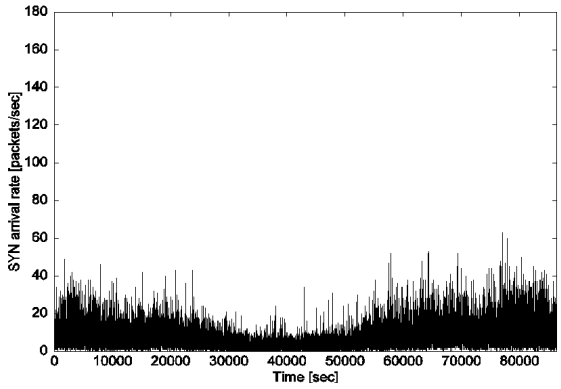
$$F(x) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}\sigma y} exp[\frac{-(\log y-\zeta)^2}{2\sigma^2}]dy. \tag{2.2}$$

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

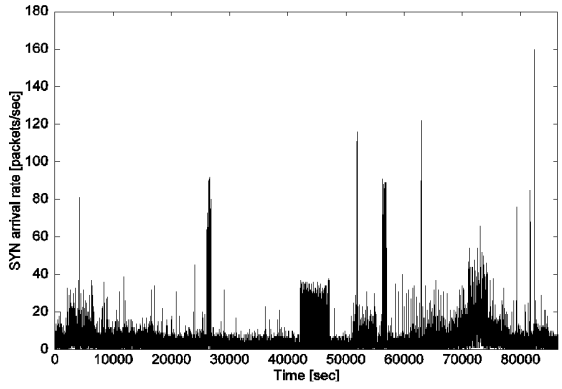
$$\hat{\zeta} = \frac{1}{n}\sum_{i=0}^{n} \log x_i \tag{2.3}$$



(a) all flows



(b) normal traffic



(c) incomplete traffic

Figure 2.1: Time-dependent variation of SYN arrival rates

$$\hat{\sigma}^2 = \frac{1}{n}\sum_{i=0}^{n}(\log x_i - \hat{\zeta}). \tag{2.4}$$

where n is the number of samples.

The equation for the Pareto distribution is

$$F(x) = \ 1 - (\tfrac{x}{k})^{\alpha}, \quad x \geq k \tag{2.5}$$

Parameters (α,k) in Pareto distribution are obtained from [72].

$$\hat{k} = min(x_1, x_2, \ldots, x_n), \tag{2.6}$$

$$\hat{\alpha} = n\left[\sum_{i=1}^{n}\log\frac{x_i}{\hat{k}}\right]. \tag{2.7}$$

The equation for the gamma distribution is

$$\Gamma(\lambda) = \int_0^{\infty} x^{\lambda-1}e^{-x}dx, \tag{2.8}$$

$$f(x) = \begin{cases} \frac{1}{\Gamma(\alpha)\beta^{\alpha}}x^{\alpha-1}e^{-\frac{x}{\beta}}, & 0 < x < \infty \\ 0, & -\infty < x < 0 \end{cases} \tag{2.9}$$

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

$$\alpha = \frac{E(X)^2}{V(X)} \tag{2.10}$$

$$\beta = \frac{V(X)}{E(X)}. \tag{2.11}$$

Figure 2.2 shows the result of fitting the normal traffic to four distributions. This figure compares the cumulative distribution of SYN packet arrival rates with the cumulative distributions described above. This curve is for the data obtained in 10-second intervals. We used 10,000 samples to obtain the SYN rate distributions. From this figure we can see that tail of the SYN rate distribution of the *normal traffics* is quite different from Pareto distribution. Among rest three distributions, the gamma distribution is most suitable for the normal traffic in the region of 99-percentile and higher. On the other hand, the normal distribution is most appropriate in the area of less than 95-percentile. The lognormal distribution can also be fit to the normal traffic at 90-percentile and below.

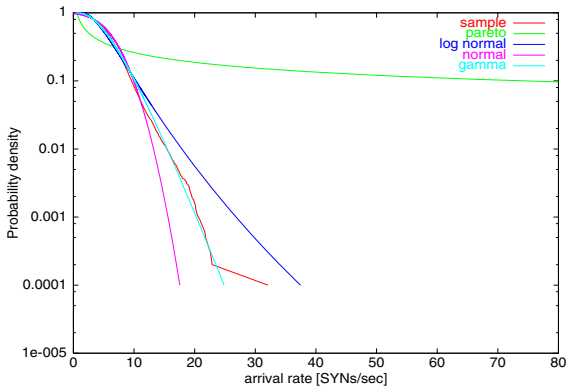


Figure 2.2: Comparisons between the distributions of SYN rates and the four distributions(*normal traffic)*

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

$$D = \frac{\sum_{i=n-\lceil nX_t \rceil}^{n} (F^{-1}(\frac{i}{n}) - r_i)^2}{\lceil nX_t \rceil - 1}. \tag{2.12}$$

We calculated the value of D for each of our measurements of the SYN arrival rate (i.e., for every 10 seconds in our experiment). We used 10,000 samples to obtain the SYN rate distributions and the samples are obtained in 10-second intervals. That is, we need total 100,000 seconds to obtain the entire distribution. We then calculate the average of squared differences for each sample by using 10,000 histories of samples. Fig. 2.3 shows the time-dependent variation of average of squared difference of normal traffic from normal, lognormal and gamma distributions. From this figure we can see that lognormal distribution is sometimes quite different from sample distribution. D on the gamma distribution is the smallest at any time, and its variation is also small. The variation of D on the normal distribution also does not vary regardless of time. From this observation, we can conclude that the gamma distribution is most appropriate to model the normal traffic statistically. The normal distribution is also useful for modeling, and the lognormal distribution gives a fair

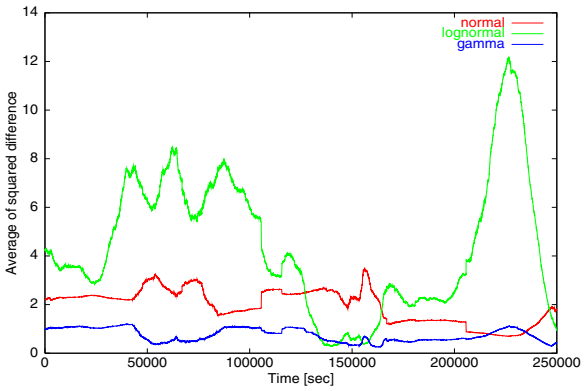


Figure 2.3: Variation of average of squared differences between the sampled SYN rates and the three distributions

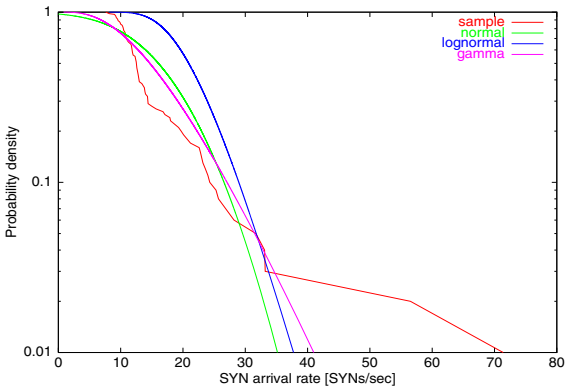
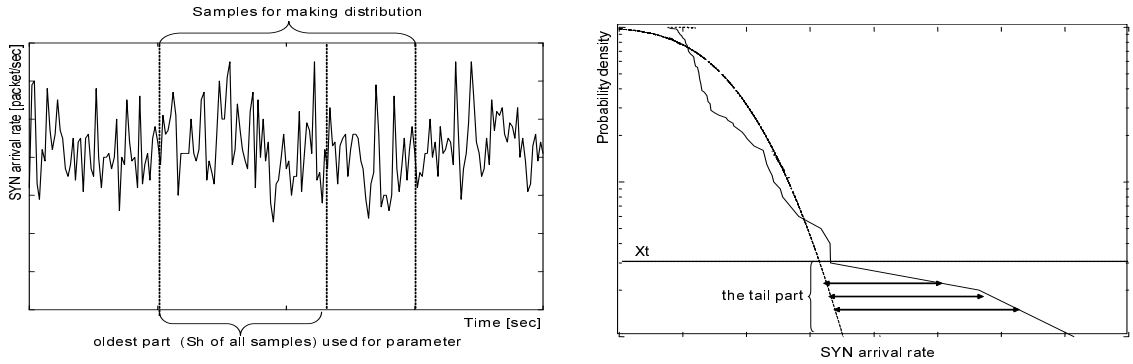


Figure 2.4: Distribution of SYN packet arrival rate when attacks started.

appropriateness.

We next evaluate for fitting statistical distributions with all traffic (i.e. the traffic including both normal and attack traffics). The results are shown in Fig. 2.4. Fig. 2.4 compares the distribution of SYN arrival rates of all flows three distributions used above. From this figure, we can observe a clear difference from the normal traffic case (Fig. 2.2). Even in gamma and normal distributions the actual traffic is far from the modeling functions. It is because the attack traffic included in the all traffic gives a strong impact to the statistics, and clearly different from human-generated characteristics (e.g., constantly high rate for a long period). Especially, the influence of the attack traffic is significantly appeared at the tail part of the distribution. This is the reason why we focus on the tail part of the distribution for distinguish the attack traffic.



(a) oldest part of samples

(b) tail of distribution

Figure 2.5: Outline of the average squared difference calculation

Attack detection method based on statistics of SYN arrival rates

As described above, 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. Based on this motivation, we propose a new detection method.

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

In calculating the average squared difference, we introduce two ratio values S_h and X_t, which

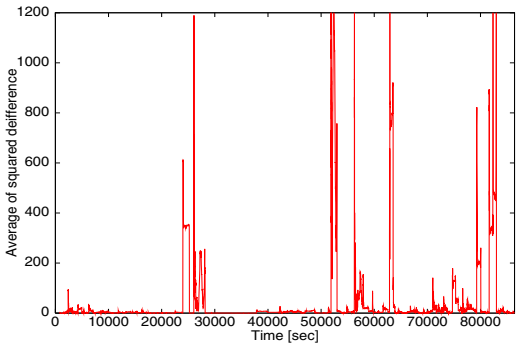
are the ratio of the oldest part of samples and the tail part of the distribution, respectively. Fig. 2.5 shows the outline of the average squared difference calculation. First, we calculate the parameter of the model function by using the S_h oldest part of sampled SYN rates. The reason why we use S_h is as follows. We calculate the value of D for each event of SYN rate calculation. The oldest one in M SYN rates are identified as the normal traffic in $M - 1$ times. That is, if no attack traffic is detected previously, the older SYN rate has a tendency to be identified as normal traffic. We then calculate the squared difference D at the range of the X_t tail part of the distribution. In this section, we set $X_t = 1 - S_h$ for simplicity.

Figures 2.6(a), 2.6(c) and 2.6(e) show the variation of the averages of squared differences for all flows and Figs. 2.6(b), 2.6(d) and 2.6(f) show the ones for *normal traffic*. According to these results, the averages of the squared differences for the *normal traffic* are quite small and stable regardless of time. The averages of the squared differences for all flows, on the other hand, rise rapidly at several points (we call them *spikes* throughout this section). Comparing Figures 2.6(a) with Figures 2.6(b) and Figures 2.6(c) with Figures 2.6(d) suggest that these *spikes* are caused by the *incomplete traffic* including attack traffic. Therefore, we can detect attacks by setting a threshold for the average of squared difference as the boundary between normal traffic and attack traffic.

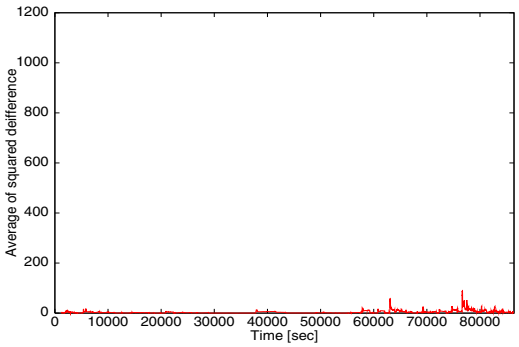
2.2 Evaluation of Attack Detection Method

Definition of attack traffic

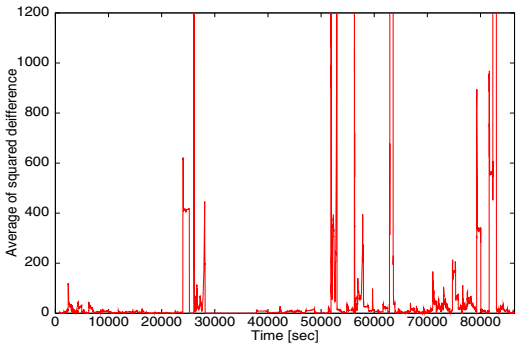
Prior to the performance evaluation, we define the attack traffic that must be detected as traffic that can put a server into a denial-of-service state. This state occurs when the backlog queue is full and new SYN packets arrive at the server. The length of the backlog queue is configured by a kernel parameter in the operating system, and the default parameters of the backlog queue on some major operating systems are listed in Table 2.2. 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 which do not complete 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.



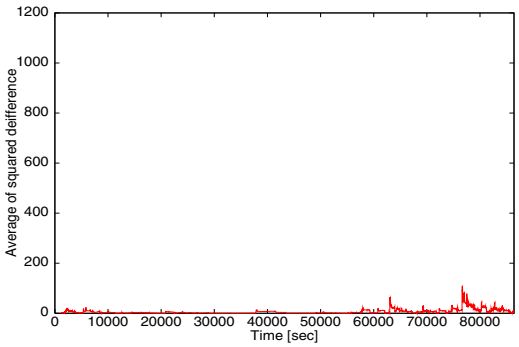
(a) all flows (compared with gamma distribution)



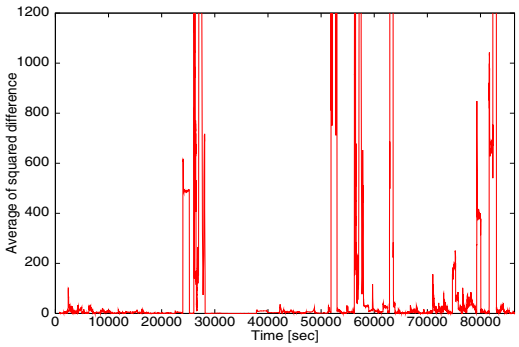
(b) normal traffic (compared with gamma distribution)



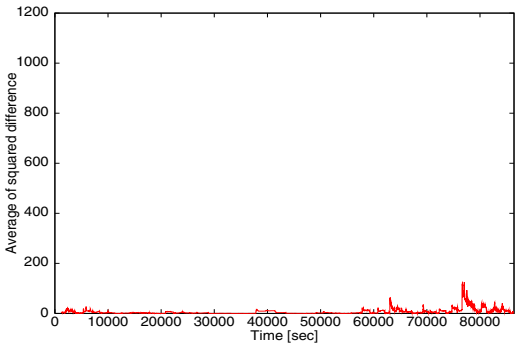
(c) all flows (compared with normal distribution)



(d) normal traffic (compared with normal distribution)



(e) all flows (compared with lognormal distribution)



(f) normal traffic (compared with lognormal distribution)

Figure 2.6: Variation of average of squared differences between the sampled SYN rates and the modeled distributed functions

Table 2.2: Default configuration of backlog queue

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

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:

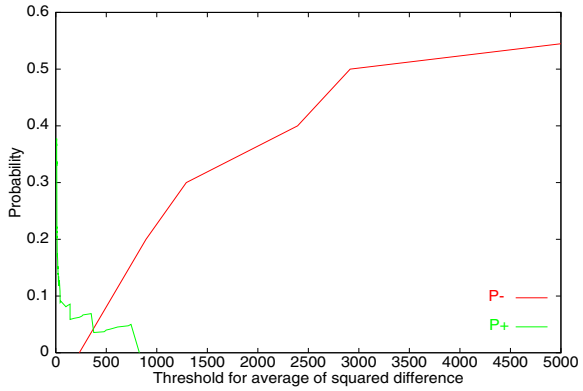
$$\begin{aligned} P_- &= \frac{\sharp \text{ of attacks not detected}}{\sharp \text{ of attacks satisfying the definition}} \\ P_+ &= \frac{\sharp \text{ of points erroneously detected as attacks}}{\sharp \text{ of points detected as attacks}} \end{aligned} \tag{2.13}$$

Probabilities of P_- and P_+ are shown in Fig. 2.7 respectively as a function of the threshold for the average of squared difference. In this regard, we set N to 100, S_h to 90 and M to 100. These figures show that both distributions could detect all attacks when we set the threshold to less than 250. Though probability of detecting erroneously was 5 % when the threshold was 250, these erroneous detections were caused by a single client sending about 20 SYNs/sec. From the viewpoints of fairness and resource managements, this relatively 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.

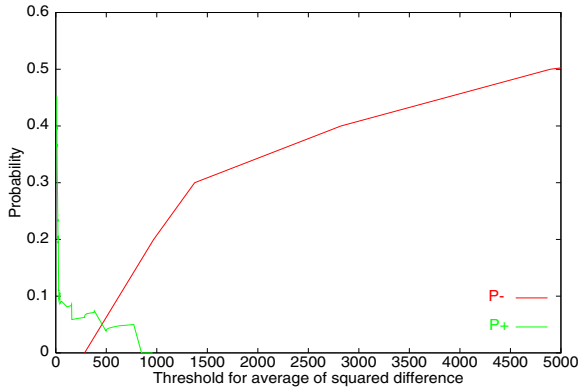
Detectable SYN rate of attack traffic

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

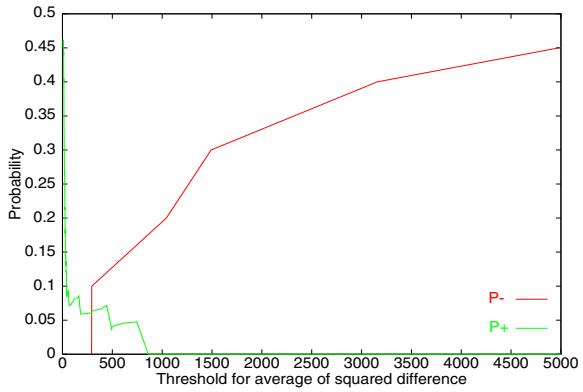
Effect of parameters in our detection method Figure 2.8 shows the SYN rates of attacks can be detected as a function of parameter S_h. We can detect lower-rate attacks by setting S_h to 75 than to 70. That is because when S_h is smaller, the number of samples used to estimate the parameters



(a) gamma distribution



(b) normal distribution



(c) lognormal distribution

Figure 2.7: Relation between threshold for average of the squared difference and the probabilities of not detecting an attack and of erroneously detecting an attack

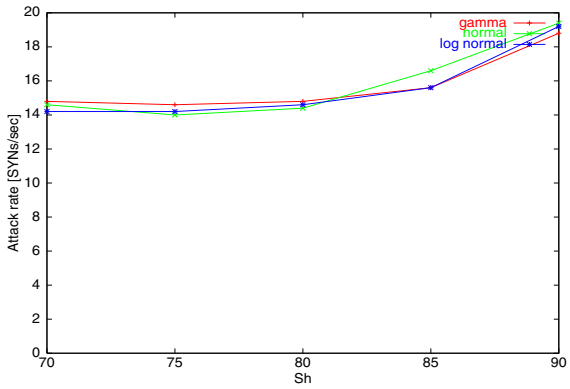


Figure 2.8: Relation between the detectable SYN rate of attack traffic and parameter S_h.

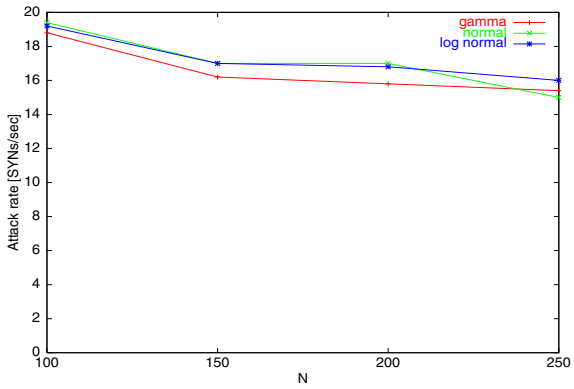


Figure 2.9: Relation between the detectable SYN rate of attack traffic and parameter N.

is smaller and we cannot model accurately. On the other hand, we can detect lower-rate attacks by setting S_h to 85 than by to 90. Too small X_t makes detection too sensitive because the number of samples compared with the models is small.

Figure 2.9 shows the SYN rates of attacks can be detected as a function of parameter N. In this regard, we set X to 90 and M to 100. When we set too small N, momentary high rates are detected erroneously. On the other hand, larger N makes attack detection duller and it takes more time to detect attacks.

Figure 2.10 shows the SYN rates of attacks can be detected as a function of parameter M. In this regard, we set S_h to 90 and N to 100. When we set M to larger value, we can model more accurately. However, we can detect lower-rate attacks by setting M to 200 than by to 250. That

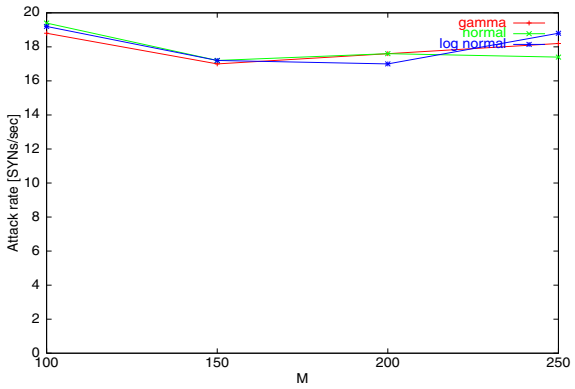


Figure 2.10: Relation between the detectable SYN rate of attack traffic and parameter M.

is because large M makes effect of attack traffic on the distribution of SYN arrival rates small and makes low-rate attacks difficult to be detected.

These results show also our method can detect smaller attacks than a single threshold cannot detect. Fig. 2.1(b) shows the SYN arrival rates vary between 10 and 50 SYNs/sec. Therefore, to avoid erroneous detection, we should set a single threshold of SYN arrival rate more than 50 SYNs/sec though the threshold cannot detect low-rate attacks which occur in hours when the traffic is relatively low. Time-of-day variation of SYN rates influences methods using a single threshold. On the other hand, our method can detect attacks regardless of time-of-day variation of SYN rates. Therefore Figures 2.8 through 2.10, we can see that our method can detect attacks whose rates are lower than 20 SYNs/sec.

Comparison among three distribution functions Figures 2.8 through 2.10 also show that there is no significant observation among three distribution functions (normal, lognormal, and gamma). So we can use any of these functions to detect the attack traffic in case of our simulation. But if we focus on the deployment of our detection mechanism, the calculation complexity is also important. It is clear that the calculation of the lognormal distribution is more complex than the one of the normal distribution. Both the normal and the gamma distributions require much computational overhead, however, the calculation of parameters in the normal distribution is very easy. Also, the calculation of the normal distribution function can be simplified by using a table of standard normal distribution. In summary, the normal distribution is most appropriate to detect the attack traffic on considering both accuracy and implementation issues.

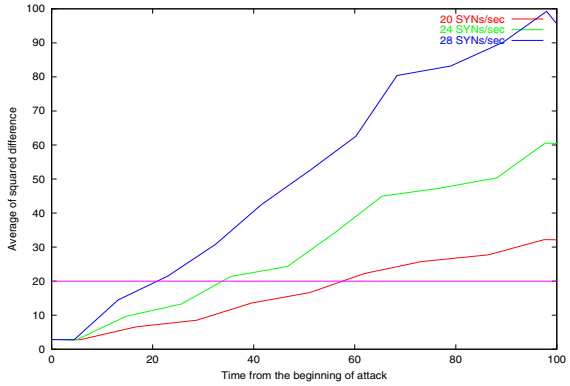


Figure 2.11: Average of squared differences versus time after the beginning of attacks with various SYN rates.

Time needed to detect the attack traffic

Figure 2.11 shows the dynamics of average of squared difference from the beginning of the attacks. In this figure, the SYN rates of the attacks are 20 SYNs/sec, 24 SYNs/sec and 28 SYNs/sec. In this figure N is 200, M is 100 and S_h is 90. We use the normal distribution as the model distribution. This figure shows that the averages of squared differences increase gradually after the beginning of attacks. When the threshold is set to 20, which detect attacks without detecting any attacks erroneously, attacks with SYN rates higher than 28 SYNs/sec can be detected within 20 seconds. In this case, the number of half-open states caused by attack is 560, which is smaller than the length of backlog queue in Linux.

To show that our mechanism can detect attacks faster, we compare the time needed to detect attacks on our method with the time on the method proposed in [22]. Throughout this section, we refer it as SYN-FIN method.

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

$$y_i = \begin{cases} 0 & (y_{i-1} + x_{i-1} - \alpha \leq 0) \\ y_{i-1} + x_{i-1} - \alpha & (otherwise) \end{cases} \tag{2.14}$$

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

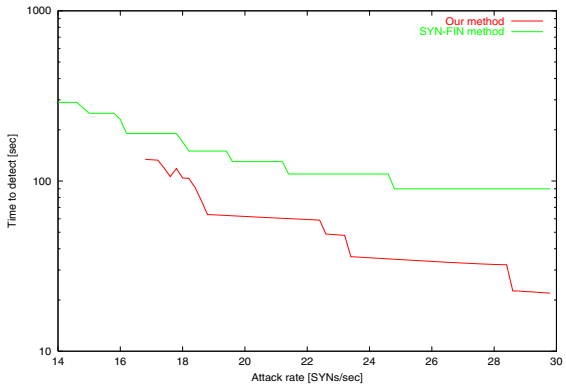


Figure 2.12: Time to detect attacks with our method and with SYN-FIN method

T.

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

Figure 2.12 compares the time to detect attacks between our method and SYN-FIN method. We varied the rate of attacking traffic and measure the time needed to detect the attacking traffic. From this figure, we can observe that our method is much faster to detect attacks than SYN-FIN method. One of the reasons is because SYN-FIN method uses a non-parametric approach to estimate the difference the characteristic of normal from the one of attacking traffics, while our method adopts a parametric approach (i.e., we model that the SYN rate of the normal traffic follows the normal distribution) to estimate it. The parametric approach can detect faster and more accurate than the non-parametric approach in the cases if the model is appropriate. However, SYN-FIN method has an advantage that it can also detect attacks with lower rate (e.g., less than 14 SYNs/sec). Our method cannot detect them because the traffic having the low rate attacks still follows the normal distribution.

Resource needed by detection method

From above results, our method can work with only 100 samples of SYN rates. If we monitor D for each destination address, we need 100 samples for each address. The captured traffic has 1,000 destination addresses in 1,000 seconds of inbound traffic, and 10,000 destination addresses in 1,000

seconds of outbound traffic. According to Fig. 2.1(b), arrival rates are not so large and we can then assume a small range of integer value (i.e., 16 bits) is enough for counting SYN rates. Then we need 200 KBytes for incoming traffic and 2 Mbytes for outgoing traffic.

2.3 Conclusion

We have analyzed the traffic at an Internet gateway and the results showed that we can model the arrival rates of normal TCP SYN packets as a normal distribution. Using this result, we described a new attack detection method taking the time variance of arrival traffic into consideration. Simulation results show that our method can detect attacks quickly and accurately regardless of the time variance of the traffic.

Section 3 Identification of Attack Nodes from Traffic Matrix Estimation

The most effective way to prevent attack traffic is to identify the attack nodes and detach (or block) attack nodes at their egress routers. However, existing traceback mechanisms are currently not widely used for several reasons, such as the necessity of replacement of many routers to support traceback capability, or difficulties in distinguishing between attacks and legitimate traffic. In this section, we propose a new scheme that enables a traceback from a victim to the attack nodes. More specifically, we identify the egress routers that attack nodes are connecting to by estimating the traffic matrix between arbitral source-destination edge pairs. By monitoring the traffic variations obtained by the traffic matrix, we identify the edge routers that are forwarding the attack traffic, which have a sharp traffic increase to the victim. We also evaluate the effectiveness of our identification method through simulation. According to the results, even when we can monitor only the link loads, our method can identify attack sources accurately and limit the total attack rate from unidentified attack sources by setting parameters adequately.

3.1 Overview of Identification Method

Our method identifies attack sources by estimating the increases in traffic between every pair of sources and destinations. We estimate the increases in traffic from the monitored link load. In the estimation of the traffic matrix, we don't focus on the total amount of traffic, but only focus on the amount of increase from the previous measurement. The reason why we use only the increases in

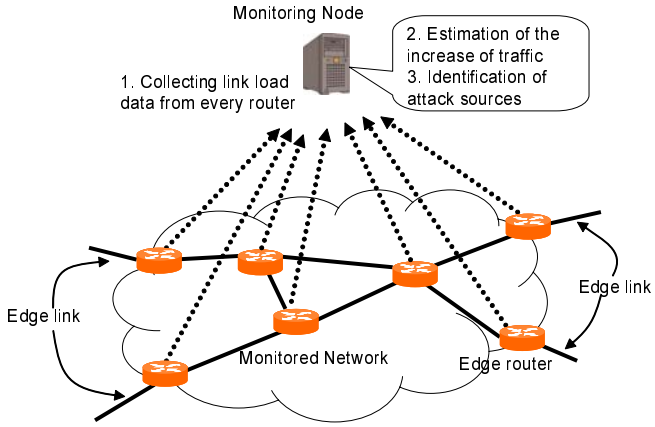


Figure 2.13: Overview of our identification method

traffic for the traffic estimation is discussed in the next paragraph. In this subsection, we first show a brief overview of our identification method.

Figure 2.13 shows an overview of our identification method. In our method, we introduce a control node to perform the process of identification of attack sources. We call this node a *monitoring node*, and we also define the region where the monitoring node controls as a *monitored network*. The monitoring node identifies the attack sources by periodically performing the following operations.

1. Obtains the statistics of the link load data from all routers in the monitored network.

2. Estimates a matrix of the increase in traffic between all arbitrary pairs of edge routers in the monitored network.

3. Identifies the attack sources from the estimated traffic increase matrix.

We can obtain link load data through SNMP. SNMP is supported by essentially every device in IP networks and is used to monitor or manage the device. That is, our method can work with existing routers.

The interval for obtaining the statistics affects the time for identifying the attack sources. If we set the interval to a larger value, the identification takes more time. On the other hand, if we set the interval to a smaller value, the loads on the routers increase though we can identify attack the sources soon after the attack starts. Thus, we should set this interval to as small value as possible without high loads on routers. In general, to avoid high loads on routers, the interval of SNMP

is set to 5 minutes. Therefore, we set this interval to 5 minutes in our evaluation described in Subsection 3.2.

In the following paragraphs, we describe the details about how to estimate the increase in traffic and how to identify the attack sources.

Estimation of Increase in Traffic

First, we assign a set of links outside the monitored network as E. We call these links *edge links*. The router, which is connected by an edge link, is called the *edge router*. We assign a set of all the links in the monitored network, including the edge links, as L.

Traffic matrix T is defined as the $|E| \times |E|$ sized matrix, whose element $t_{i,j} \quad (i, j \in E)$ indicates the amount of traffic traversing from edge link i to edge link j. We can obtain the link loads from each router through SNMP. The link loads can be denoted by the $2|L|$-size link load matrix X as follows:

$$X = \begin{bmatrix} x_1^{\mathrm{f}} \\ x_1^{\mathrm{b}} \\ x_2^{\mathrm{f}} \\ x_2^{\mathrm{b}} \\ \vdots \\ x_{|L|}^{\mathrm{f}} \\ x_{|L|}^{\mathrm{b}} \end{bmatrix}. \tag{2.15}$$

In matrix X, elements $x_l^{\mathrm{f}} \quad (l \in L)$ and $x_l^{\mathrm{b}} \quad (l \in L)$ indicate the amount of traffic on link l in the forward and backward directions respectively, because most of the network links are bidirectional. We only use the words forward/backward to distinguish the direction of the link. Therefore, there is no policy for determining the forward or backward direction of each link. However, we must distinguish between the ingress and egress traffic. To distinguish between them, we denote the ingress traffic measured on edge link i as $x_i^{\mathrm{in}} \quad (i \in E)$ and egress traffic measured on the edge link j as $x_j^{\mathrm{out}} \quad (j \in E)$.

Traffic Matrix Estimation using *Gravity Model* We estimate the traffic matrix of each pair of edge links from the link loads and routing information in monitored network. Ref. [6] uses a *gravity model* to estimate the traffic matrix. The gravity model assumes that traffic from a source to a destination is proportional to the total traffic at the source and at the destination. Using this

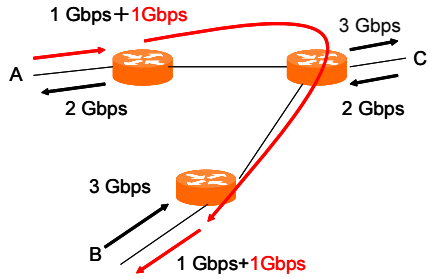


Figure 2.14: Simple example of DoS attack

model, we estimate the traffic matrix from

$$t_{i,j} = x_i^{\text{in}} \frac{x_j^{\text{out}}}{\sum_{\forall k \in E} x_k^{\text{out}}} \quad (i, j \in E) \ , \tag{2.16}$$

where x_i^{in} is the element of X corresponding to the amount of ingress traffic to the monitored network measured on the edge link i and x_j^{out} is the egress traffic measured on the edge link j.

However, we cannot estimate increases in traffic accurately using Eq. (2.16) as follows. We assume that an attack traffic whose rate is t_{attack} traverses from i to j. We also assume legitimate traffic $t_{i,j}$ can be accurately estimated by Eq. (2.16). Traffic from i to j, including the attack traffic is estimated from

$$t'_{i,j} = (x_i^{\text{in}} + t_{\text{attack}}) \frac{x_j^{\text{out}} + t_{\text{attack}}}{\sum_k x_k^{\text{out}} + t_{\text{attack}}}, \tag{2.17}$$

where $t'_{i,j}$ is the traffic traversing from i to j including attack traffic. Then, the increased traffic by the attack is estimated by

$$t'_{i,j} - t_{i,j} = \frac{t_{\text{attack}}^2 + t_{\text{attack}}(x_i^{\text{in}} + x_j^{\text{out}})}{\sum_{k \in E} x_k^{\text{out}} + t_{\text{attack}}}, \tag{2.18}$$

where $t_{i,j}$ is the legitimate traffic from i to j. Fig. 2.14 shows a simple example. In this example, we assume the total rate of traffic in the monitored network is 6 GBytes/sec, both x_A^{in} and x_B^{out} are 1 GBytes/sec. We also assume the attack traffic from the edge link A to B has the rate of 1 GBytes/sec. From Eq. (2.18), the total traffic, including the attack traffic from edge link i to j is estimated as 0.55 GBytes/sec, which is quite different from the attack rate (1 GBytes/sec).

As previously mentioned, when attack traffic is injected, the estimated increase in traffic is proportional to the total rate of traffic monitored at the source. That is, the gravity model is infeasible

for directly estimating the attack traffic because the impact of the attack traffic is distributed among the edge links that have legitimate traffic to the victim. As a result, the estimated attack rate is significantly lower than the rate of the attack traffic that is really generated.

Traffic matrix estimation focusing on increased traffic To accurately estimate the increase in traffic, we propose a matrix estimation method focusing not on the total rate of traffic but on the increase in traffic.

First, we calculate the increases in traffic on each link from

$$G_n = X_n - \bar{X}_n, \tag{2.19}$$

where G_n is the $2|L|$-size vector in which the elements $g^{\mathrm{f}}_{i,n} \quad (i \in L)$ and $g^{\mathrm{b}}_{i,n} \quad (i \in L)$ indicate the increase in traffic on link i in the forward and backward directions at time n, respectively. X_n is the link load vector at time n and $\bar{X}_n$ is the $2|L|$-size vector in which $\bar{x}^{\mathrm{f}}_{i,n}$ is the average rate of legitimate traffic on the link i in the forward direction before time n and $\bar{x}^{\mathrm{b}}_{i,n}$ is one on the same link in the backward direction. We explain how to calculate $\bar{X}_n$ later.

Then, by using G_n, we estimate the increases in traffic between every pair of sources and destinations. The increase in traffic can be shown as a $|E|\times|E|$ matrix F_n whose element $f_{i,j,n} \quad (i,j \in E)$ indicates the increase in traffic traversing from edge link i to edge link j.

Eq. (2.16) cannot be used to estimate the traffic increase matrix from G_n, which may include negative values, because it supports only positive values. Therefore, we modify Eq. (2.16) to support negative values. We define the traffic increase matrix F_n, having the traffic increase $f_{i,j,n}$, from edge link i to j between the time $n-1$ and n. The value of $f_{i,jn}$ is calculated from

$$f_{i,j,n}=\begin{cases} g^{\mathrm{in}}_{i,n}\dfrac{g^{\mathrm{out}}_{j,n}}{\sum_{\{k:(g^{\mathrm{out}}_{k,n}>0)\}} g^{\mathrm{out}}_{k,n}} & (g^{\mathrm{in}}_{i,n}>0, g^{\mathrm{out}}_{j,n}>0) \\ -\left| g^{\mathrm{in}}_{i,n}\dfrac{g^{\mathrm{out}}_{j,n}}{\sum_{\{k:(g^{\mathrm{out}}_{k,n}<0)\}} g^{\mathrm{out}}_{k,n}} \right| & (g^{\mathrm{in}}_{i,n}<0, g^{\mathrm{out}}_{j,n}<0) \\ 0 & (\text{others}). \end{cases} \tag{2.20}$$

Focusing on the increase in the traffic, we can eliminate the effect of the amount of legitimate traffic and estimate the increase in the traffic more accurately. That is, we can estimate that the increase in traffic from attack sources to the victim is large by checking the increase in traffic when the attack starts. If the monitored network suffers from multiple attacks whose sources and victims are different, some traffic from different sources to different destinations concurrently increases.

In this case, the estimated increase in traffic is proportional to the increase in traffic measured at the sources. That is, traffic from a source of an attack to a victim of another attack is estimated as increased. However, we can identify the attack sources that generate the attack traffic, even if we could not identify the victim node exactly where the attack source sends the attack traffic to.

Modification of traffic matrix Although F_n is a $|E| \times |E|$ matrix, F_n can be denoted as following the $|E|^2$-size vector;

$$F_n = \begin{bmatrix} f_{1,1,n} \\ f_{1,2,n} \\ \vdots \\ f_{1,|E|,n} \\ f_{2,1,n} \\ \vdots \\ f_{|E|,|E|,n} \end{bmatrix} \tag{2.21}$$

Due to the fact that the total amount of traffic on the link is the summation of the traffic of flows that are passing the link, F_n and G_n satisfy

$$G_n = AF_n, \tag{2.22}$$

where A is a $2|L| \times |E|^2$ routing matrix which is given by

$$A = \begin{bmatrix} a^{\mathrm{f}}_{1,1,1} & a^{\mathrm{f}}_{1,2,1} & \cdots & a^{\mathrm{f}}_{|E|,|E|,1} \\ a^{\mathrm{b}}_{1,1,1} & a^{\mathrm{b}}_{1,2,1} & \cdots & a^{\mathrm{b}}_{|E|,|E|,1} \\ a^{\mathrm{f}}_{1,1,2} & a^{\mathrm{f}}_{1,2,2} & \cdots & a^{\mathrm{f}}_{|E|,|E|,2} \\ a^{\mathrm{b}}_{1,1,2} & a^{\mathrm{b}}_{1,2,2} & \cdots & a^{\mathrm{b}}_{|E|,|E|,2} \\ \vdots & \vdots & \ddots & \vdots \\ a^{\mathrm{f}}_{1,1,|L|} & a^{\mathrm{f}}_{1,2,|L|} & \cdots & a^{\mathrm{f}}_{|E|,|E|,|L|} \\ a^{\mathrm{b}}_{1,1,|L|} & a^{\mathrm{b}}_{1,2,|L|} & \cdots & a^{\mathrm{b}}_{|E|,|E|,|L|} \end{bmatrix}. \tag{2.23}$$

$a^{\mathrm{f}}_{i,j,k} \quad (i, j \in E, k \in L)$ is equal to 1 if the traffic from edge link i to edge link j traverses on link k in the forward direction, or set to zero otherwise. In a similar way, $a^{\mathrm{b}}_{i,j,k} \quad (i, j \in E, k \in L)$ is equal to 1 if the traffic from edge link i toedge link j traverses on link k in the backward direction or zero otherwise. Matrix A can be obtained by monitoring the routing messages, such as the Link

State Advertisement (LSA) of OSPF [73] or by simulating routing [74].

The traffic matrix estimated by the gravity model cannot satisfy Eq. (2.22) because Eq. (2.20) does not use the traffic statistics on the internal links of the monitored network, but uses only the traffic measurements of the edge links. Therefore, we adjust the traffic matrix estimated by the gravity model to satisfy Eq. (2.22). We can obtain the final estimation results for F_n from

$$F_n = F'_n + A^+(G_n - AF'_n), \tag{2.24}$$

where F'_n is the $|E|^2$-size vector indicating the results estimated by Eq. (2.20), and A^+ is a pseudo-inverse matrix of A. We can obtain the least squares solution of $X = AT$ by multiplying X by the pseudo-inverse matrix of A. That is, by Eq. (2.24), we obtain F_n which satisfies $Gn = AF_n$ and minimizes $|F_n - F'_n|^2$. In our evaluation, we obtain A^+ by using a function of Scilab [75].

How to estimate average of legitimate traffic Our method for estimating the increase in traffic uses the average rate of legitimate traffic. The rate of legitimate traffic varies according to the time of day. To follow the time-of-day variation of this traffic, we assume that the average rate of legitimate traffic $\bar{X}_n$ is basically estimated by the exponentially weighted average of the monitored traffic rate from

$$\bar{X}_{n+1} = \ \alpha X_n + (1-\alpha)\bar{X}_n \quad (0 < \alpha < 1) \ . \tag{2.25}$$

Note here, other estimation method (e.g. AutoRegressive Integrated Moving Average (ARIMA) model) can also be used to estimate the average rate of legitimate traffic. However, according to [37], exponentially weighted average is almost as accurate as ARIMA model though it is very simple. For this reason, we use the exponentially weighted average described above.

However, when the traffic increases suddenly and rapidly (we call these *spikes* throughout the rest of this chapter), $\bar{X}_n$ becomes large after the spike. The large $\bar{X}_n$ value causes difficulties in the identification of the increase in traffic after the spike, because the larger $\bar{X}_n$ value makes the impact of $(X_n - \bar{X}_n)$ small, even for cases of increases in traffic. For this reason, we must estimate the average of the legitimate traffic without the effect of spikes.

We can eliminate the effect of spikes by updating only the elements of $\bar{X}_n$ corresponding to the link on which the increase in traffic is under a threshold. However, as described in the previous paragraph, our method assumes the situation covered by Eq. (2.22). For this reason, we should update $\bar{X}_n$ by satisfying Eq. (2.22).

For this purpose, we update $\bar{X}_n$ using an element from estimated F_n, which is not rapidly

increasing. First, we extract the element not increasing rapidly from F_n. We denote the $|E| \times |E|$ matrix of the extracted elements as $\hat{F}_n$. Each element $\hat{f}_{i,j,n} \quad (i,j \in E)$ is defined by

$$\hat{f}_{i,j,n} = \begin{cases} f_{i,j,n} & (f_{i,j,n} < \mu_{i,j} + \beta\sigma_{i,j}) \\ 0 & (\text{others}) \end{cases} . \tag{2.26}$$

where $\mu_{i,j}$ is the average of the last J values of $f_{i,j,k} \quad (i,j \in E, n-J < k \leq n)$ and $\sigma_{i,j}$ is the variance of the last J values of $f_{i,j,k} \quad (i,j \in E, n-J < k \leq n)$. β is the parameter by which we can set the threshold. By Eq. (2.26), when the traffic from i to j sharply increases at time n beyond the threshold, $\hat{f}_{i,j,n}$ is zero, while in other cases, $\hat{f}_{i,j,n}$ is $f_{i,j,n}$.

After that, we update $\bar{X}_{n+1}$ with the following equation.

$$\bar{X}_{n+1} = \alpha(\bar{X}_n + A\hat{F}_n) + (1-\alpha)\bar{X}_n \tag{2.27}$$

In Eq. (2.27), we calculate the increase in traffic on each link from $\hat{F}_n$ by $A\hat{F}_n$. Using the increase in traffic, we calculate the amount of traffic at time n as $\bar{X}_n + A\hat{F}_n$. Then, we update $\bar{X}_{n+1}$ as the weighted average of the monitored traffic using the amount of traffic at time n.

With the above stated equations, we can update $\bar{X}_{n+1}$ without the effect of any spikes in F_n. By deciding whether each element of F_n should be used to update, we can satisfy Eq. (2.22).

The above model to estimate the average of legitimate traffic uses three parameters, α, β and J. If the change of legitimate traffic causes large entries in G_n, the legitimate traffic is erroneously identified as an attack. To avoid this erroneous detection, we should set the parameters to the value which can minimize G_n during no attack times.

α indicates the weight to current measurement. Setting α to a large value, we cannot eliminate the impact of temporal change of traffic on the estimated average of legitimate traffic. As a result, the impact enlarges the increase of legitimate traffic from the estimated average of legitimate traffic. However, setting α to a small value, the estimated average of legitimate traffic cannot follow the periodic change of traffic. Therefore, we use the traffic data monitored before to set α to adequate value. By using the traffic data monitored before, we set α to the value which can minimize the squared errors between monitored traffic rate and its estimated rate.

By β, we can set the sensitivity to detect the spikes. If we set β to a large value, the spike also affect the estimated average of legitimate traffic. On the other hand, if we set β to a small value, the periodical change of traffic may mistakenly be identified as a spike. As a result, we cannot update

the estimated average of legitimate traffic. Therefore, we use the traffic data monitored before and set β to as small value as possible without identifying the periodic change of the monitored traffic as a spike.

J is the number of monitored data used for setting a threshold to detect spikes. By setting J to large value, we use more monitored data. However, setting J to large value needs more memories to store the monitored data. Therefore, we set J to as large value as possible.

Identification of attack sources

When an attack starts, the traffic sharply increases from the attackers to the victim. Moreover, the larger the increase is, the more serious the impact on the network resources is. We identify the sources increasing the traffic on the victim as attack sources. However, when many attack sources are widely distributed, the impact of the attack is serious, even if each attack source generates a small rate of attack traffic. Thus, the identification of the attack sources, by setting a static threshold to the increase in traffic, is not sufficient. Instead of setting a threshold, we identify the attack sources by comparing the increase in traffic from each edge link to the victim. When the victim detects an attack, it is reasonable enough to assume that the source generating more traffic to the victim has more likelihood of being considered an attack source. With this assumption, we identify attack sources from the nodes generating a lot of traffic to the victim node. We also use the total rate of traffic to detect the event of an attack. By using the total rate of attack traffic, we can identify the attack sources even in cases of DDoS. The total rate of attack traffic can be estimated from the increase of the egress traffic to the victim.

When an attack starts, the egress traffic increases with the rate of the attack traffic. However, the rate of legitimate traffic may also change according to the time-of-day. Assuming the increase of egress traffic to the victim is attack traffic may be an overestimation of the attack traffic, because an increase in egress traffic includes both legitimate and attack traffic. As a result of this overestimation, the source node sending only legitimate traffic may be misled as an attack source. For this reason, we estimate the rate of the attack traffic $\tilde{g}^{\mathrm{out}}$ from results of traffic estimation. When an attack to edge link j starts at the time n, $\tilde{g}^{\mathrm{out}}$ is estimated from

$$\tilde{g}^{\mathrm{out}} = g^{\mathrm{out}}_{j,n} - \mu^{\mathrm{out}}_{j} - \gamma, \tag{2.28}$$

where $g^{\mathrm{out}}_{j,n}$ is the egress traffic on edge link j to the outside of the monitored network, μ^{out}_{j} is the average of the last J values of $g^{\mathrm{out}}_{j,k}$ $(n - J \leq k < n)$, and γ is the parameter indicating

the variation in the rate of the legitimate traffic. In this equation, $\mu_i^{\rm out}$ represents the effect of the time-of-day variation of the legitimate traffic and γ mitigates the effect of the other variations of the legitimate traffic. By adequately setting γ, we can estimate $\tilde{g}^{\rm out}$ as the value which may be a little smaller than the actual attack rate, but is never larger than the actual attack rate.

Then, we identify source i as attack source when source i satisfies

$$\sum_{(k:f_{k,j,n}>f_{i,j,n})} f_{k,j,n} \leq \tilde{g}^{\rm out}, \tag{2.29}$$

where $f_{i,j,n}$ is the element of the estimated traffic increase matrix F_n corresponding to the traffic from edge link i to victim edge link j. Before using Eq. (2.29), we must first sort out the set of $f_{k,j,n}(1 \leq k \leq N)$ by descending order based on their values. We then calculate the total of the top m traffic to the victim node. We compare the total top m traffic with the estimated egress traffic $\tilde{g}^{out}$. We increment m by one and calculate the total top m traffic until the total traffic exceeds $\tilde{g}^{out}$. Finally, we identify these m nodes as the attack sources.

Let us denote the actual rate of attack traffic as $t^{\rm attack}$ and that the sum of the top m increases of the egress traffic to the victim as $t^{{\rm top}(m)}$. If $t^{{\rm top}(m)}$ is smaller than $\tilde{g}^{\rm out}$ and $t^{{\rm top}(m+1)}$ is larger than $\tilde{g}^{\rm out}$, then we can identify $m+1$ attack sources. In this case, the total rate of attack traffic from the identified attack sources is $t^{{\rm top}(m+1)}$, which is larger than $\tilde{g}^{\rm out}$. That is, the rate of the attack traffic from the unidentified attack sources is at most $t^{\rm attack} - \tilde{g}^{\rm out}$, which is calculated from $\gamma+\mu-f^{\rm normal}$ where $f^{\rm normal}$ is the increase in legitimate traffic. Therefore, by setting γ adequately, we can identify most of the attack sources and limit the rate of attack traffic from the unidentified attack sources.

Calculation time of our method

In our method, we estimate the traffic increase matrix from Eq. (2.20), Eq. (2.24) and Eq. (2.27). The calculation time for Eq. (2.20) is $O(|E|^2)$ because we should estimate the traffic transmitted between every pair of ingress and egress points. Although Eq. (2.24) needs the value of A^+, we do not have to calculate A^+ each time, since A seldom varies. The calculation times for Eq. (2.24) and Eq. (2.27) are $O(|L||E|^2)$, because they include the products of a $2|L| \times |E|^2$ matrix and a $|E|^2$-sized vector. That is, the calculation time for estimating the traffic increase matrix is $O(|L||E|^2)$.

To identify the attack sources, we check whether the candidate satisfies Eq. (2.29). The number of candidates is $|E|$. If $f_{k,j,n}(1 \leq k \leq N)$ are sorted by descending order, we check the condition at most $|E|$ times. Using a quicksort algorithm, we can sort $|E|$ elements by less than $O(|E|^2)$

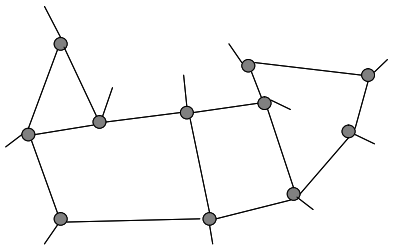


Figure 2.15: Backbone Topology of the Abilene

comparisons. That is, the calculation time for identifying the attack sources using the estimated matrix is $O(|E|^2)$.

Consequently, the calculation time for our method is $O(|L||E|^2)$. However, in a large network, we can reduce the calculation time by using a link load on only a part of the links, not on all links. This can be done by taking A and X_n from a part of links.

3.2 Evaluation of Identification Method

We evaluate our method by using simulations. In our simulation, we use the backbone topology of Abilene (11 nodes and 14 links) shown in Fig. 2.15 for the monitored network. We assume that each node in Fig. 2.15 has one edge link. That is, in this simulation, the purpose of our method is to extract nodes connecting to attackers from 11 nodes in Fig. 2.15. We use the traffic data captured for 24 hours with 5 minutes interval on the Abilene backbone network for the legitimate traffic in the simulation. The sampling rate of the data is 1:100 (that is, one out of every 100 packets is sampled). In this simulation, we use packets/sec to measure the traffic rate, because attacks sending a number of small packets (including SYN flood attacks, which are the most frequent attacks [16]) affect packets/sec more significantly than byte/sec.

In our simulations, we set α to 0.3 and β to 3, which allows a time-of-day variation of the traffic.

Accuracy in estimating the increase of traffic

First, we validate that our method can accurately estimate the increase in traffic. Fig. 2.16 shows the time-dependent variation of the arrival rate of each packet between a source and a destination. Fig. 2.17 compares the actual time-dependent variation of the increase in arrival traffic with its estimated rate. Comparing Fig. 2.16 and Fig. 2.17, we can see that by monitoring the increase in traffic, we can eliminate the time-of-day variation of the traffic. That is, by monitoring the increase

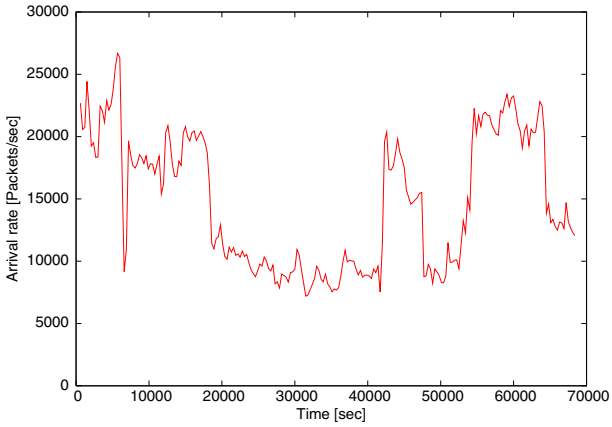


Figure 2.16: Time-dependent variation of arrival rate of packets

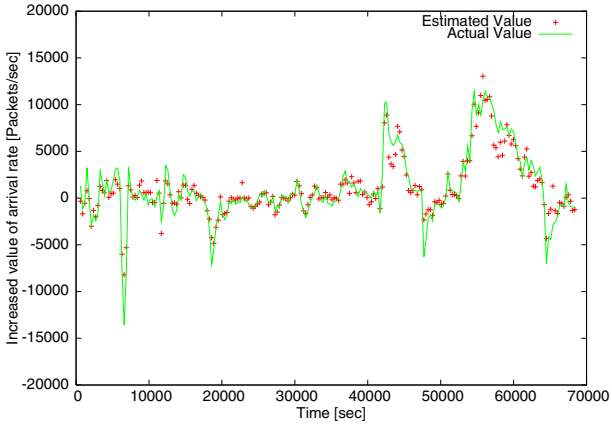
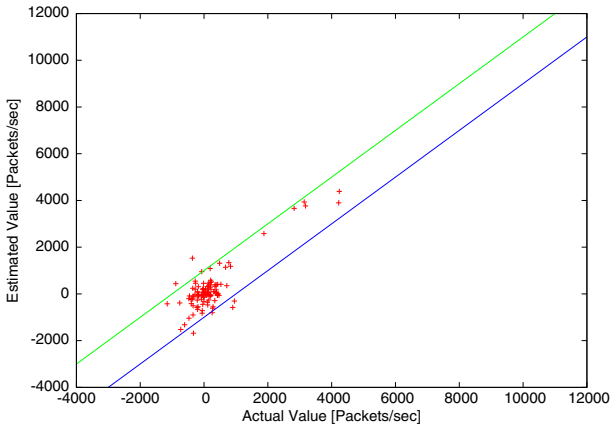


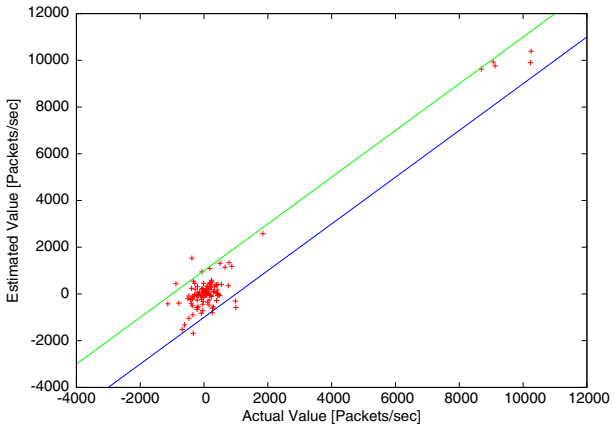
Figure 2.17: Time-dependent variation of increase of arrival rate of packets between source and destination.

in traffic, we can identify the attack sources without the effect of a time-of-day variation in the traffic. From Fig. 2.17, we also see that in the cases where a rapid increase in traffic occurs, our method can accurately estimate it.

We perform another simulation to evaluate accuracy when attacks from several sources start. We inject attack traffic from randomly selected five sources to a single destination. Fig. 2.18 compares the results of the estimations with actual values. The horizontal axis is the actual rate of traffic and the vertical axis is the estimated value. In Fig. 2.18(a), the attack rate from each source is 4000 packets/sec. In this case, 25% of all packets to the victim are attack packets. In Fig. 2.18(b),



(a) 4000 packets/sec attack injected



(b) 10000 packets/sec attacks injected

Figure 2.18: Estimated increase vs. Actual increase when attack started

the attack rate from each source is 10000 packets/sec. In this case, about a half of all packets to the victim are attack packets. The lines in both figures show $x \pm 1000$. These figures show we can accurately estimate the increase in traffic. Even for large attacks, we can estimate the increase in traffic with an error rate of less than 1000 packets/sec.

Table 2.3: Number of attack sources vs. false-positives and false-negatives

# of attack sources (total # of attack sources)	# of false-negatives (false-negative rate)	# of false-positives (false-positive rate)
1 (14)	0 (0.00)	12 (0.09)
2 (28)	0 (0.00)	6 (0.05)
3 (42)	2 (0.04)	12 (0.12)
4 (56)	6 (0.14)	14 (0.16)
5 (70)	14 (0.20)	16 (0.21)

Accuracy of identification of attack sources

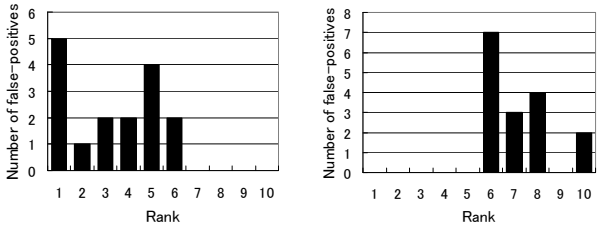
Definition of false-positive and false-negative The accuracy of our method for identifying attack sources is evaluated by two metrics, false-positive and false-negative. We define false-positive as a case where a source not generating attack traffic is erroneously identified as an attack sources. We define false-negative for cases where an attack source cannot be identified. That is, the number of false-positives indicates the number of sources erroneously identified as attack sources and the number of false-negatives indicates the number of attack sources that cannot be identified. We also define the false-negative and false-positive rates as follows:

$$\text{false-negative rate} = \frac{\text{\# of false-negative}}{\text{total \# of attack sources}}$$

$$\text{false-positive rate} = \frac{\text{\# of false-positive}}{\text{total \# of sources not generating attack traffic}}$$

Number of attack sources vs. false-positives and false-negatives We simulate our method to identify attack sources, changing the number of attack sources. We inject attack packets at 14 different times which are randomly selected. We change the number of attack sources from one to five and attack sources are randomly selected. In this simulation, we set the attack rates so that 25% of all packets to the victim are attack packets and the attack rate from each attack source is equal. We set γ to 6000 packets/sec.

Table 2.3 shows the total number of false-positives and false-negatives of 14 attacks and their rates. From these results, we can accurately identify the attack sources regardless of the number of attack sources.



(a) Rank of estimated increase vs. number of false-positives

(b) Rank of actual increase vs. number of false-positives

Figure 2.19: Rank of increase in traffic vs. number of false-positives

However, there are a few false-positives. Therefore, we investigate such false-positives. Fig. 2.19 shows where these false-positives are ranked in estimated increase in traffic and actual increase in traffic when the number of attack sources is five. In this figure, the horizontal axis is the rank order in estimated increase in traffic and actual increase in traffic and vertical axis is the number of false-positives corresponding to the rank order. From this figure, though actual increases in traffic from the sources mistakenly identified are ranked 6th or lower, estimated ones are ranked 5th or higher. That is, the reason of these false-positives is estimation errors. These estimation errors are caused by the rapid increase in traffic traversing to the link that is near the link to the victim. In these cases, the rapid increases cause errors because most of the path of the increased traffic is common with the path from the source of the increased traffic to the victim.

γ vs. false-positive and false-negative We evaluate the relationship between γ and the false-positives or false-negatives in our method by using a simulation with various values of γ. In this simulation, we inject attack packets from randomly selected five sources at randomly selected 14 different times. Fig. 2.20 shows the results. In this figure, we inject two kinds of attacks. First, the attack rate from each source is 4000 packets/sec. In this case, 25 % of all packets to the victim are attack packets. In the second one, the attack rate from each source is 10000 packets/sec. In this case, about a half of all packets to the victim are attack packets. From Fig. 2.20, we can see that the proposed method can reduce the number of false-positives by setting γ to a larger value. However, a large γ causes many false-negatives. In addition, when comparing two kinds of attacks, we can also see that if we set γ to the same value, we have less false-negatives in cases of larger

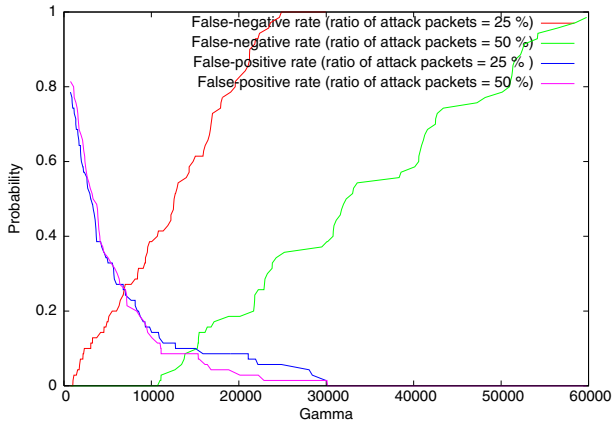


Figure 2.20: γ vs. false-negative and false-positive

attacks than in smaller attacks. From this figure, we can also see that the number of false-positives is almost the same, regardless of the injected attack rate. That is, the attack rate does not affect the number of false-positives.

γ vs. attack rate from unidentified attack sources To evaluate the relationship between γ and the total rate of attacks from unidentified attack sources, we simulate our method to identify attack sources, changing the attack rate. In this simulation, we inject attack packets from randomly selected five sources at randomly selected 14 different times and the attack rate from each source is equal.

In Fig. 2.21, the horizontal axis is the total rate of the attack traffic. Each line shows γ, which can identify one of the five attack sources, two of the five attack sources, three of the attack sources, four of the attack sources and all of the attack sources at all time. From this figure, we can see that a smaller γ is needed to identify attack sources for smaller attacks or to identify more attack sources. This figure also shows that even when we set γ to the same value, we can identify more attack sources for large attacks. For example, by setting γ to 10000 packets/sec, we can identify only one attack source when the attack rate from each attacker is 2000 packets/sec. However, by setting γ to the same value, we can identify four attack sources when the attack rate is 6000 packets/sec.

Figure 2.22 shows the relationship between γ and the total rate of attack traffic from unidentified attack sources. In this figure, the three lines indicate the false-positive rate and the maximum and average of the total rate of attack traffic from the unidentified attack sources. From this figure, we can see that by setting γ to a smaller value, the attack rate from unidentified attack sources can be

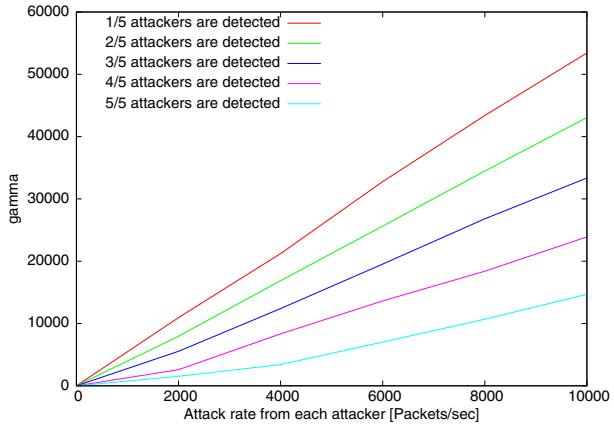


Figure 2.21: Relationship between attack rate and γ to identify attachk sources

small while a smaller γ causes more false-positives. We can also see that the average of the total rate of attack traffic from unidentified attack sources is near γ. That is, the total rate of attack traffic from unidentified attack sources is closely related to γ.

However, in some cases, the total rates of attack traffic from unidentified attack sources are higher than γ. There are two reasons for this. First one is caused by the decrease of legitimate traffic to the victim. In this case, our method underestimates the total attack rates to the victim. Another reason is caused by errors in our method for estimating the increases in traffic. Our method for estimation has errors in the range of ± 1000 packets/sec. That is, the estimated increase in traffic from an attack source may be 1000 packets/sec less than the actual increase, while the difference from one to another attack source may be 1000 packets/sec larger than the actual increase. In this case, this error causes 1000 packets/sec attack traffic from unidentified attack sources. However, we can accurately identify attack sources sending attack traffic whose estimated rate is larger than $\gamma + \mu - f^{\mathrm{normal}}$. That is, by adequately setting γ, we can identify attack sources even when the estimated increases have several errors.

As previously mentioned, the total rate of attack traffic from unidentified attack sources is closely related to γ. That is, by defining the maximum attack rate that does not affects the network resources, we can adequately set γ to limit the total attack rate from unidentified attack sources to the defined maximum attack rate.

Comparison with existing method Finally we compare our method with existing method.

The method proposed in Ref. [36] separates traffic matrix into normal and abnormal subspaces

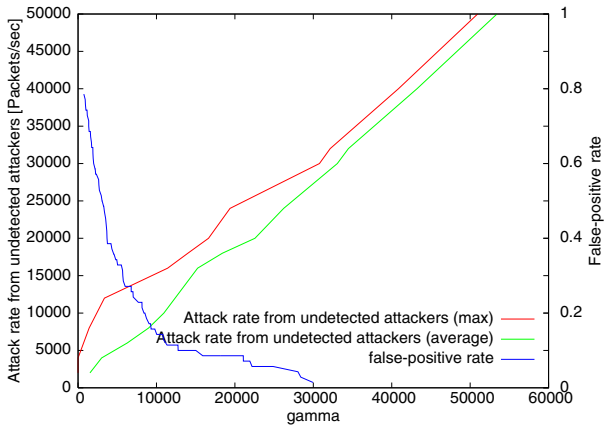


Figure 2.22: γ vs. total rate of traffic from unidentified attack sources

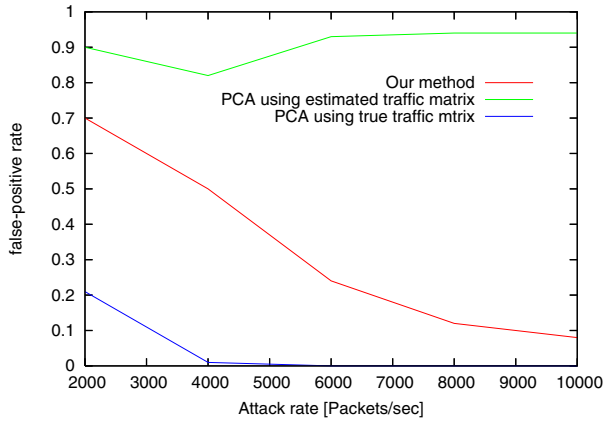


Figure 2.23: Comparison of our identification method and PCA method using estimated traffic matrix

by applying Principal Component Analysis (PCA). When the square sum of abnormal subspaces is larger than the threshold, it is detected as attacks. These abnormal subspaces also can be used to identify attack sources. Using abnormal subspaces, we can identify all attack sources by identifying the sources having the largest abnormal subspaces, until the squared sum of the abnormal subspaces of traffic which are not identified as attack becomes less than the threshold. In this simulation, we use two traffic matrices for PCA method. One is the true traffic matrix. Another is estimated by the method proposed in Ref. [6]. By using estimated traffic matrix, we compare our method with PCA method in the conditions that we can only monitor link loads.

We compare our proposed and PCA methods by simulation. In this simulation, we inject attack packets from randomly selected five sources at randomly selected 14 different times and the attack rate from each source is equal. We compare the false-positive rates when we set the thresholds so that false-negative rates are less than 0.1. Fig. 2.23 shows the results. In this figure, the horizontal axis is the attack rate from one attack source and the vertical axis is the false-positive rate. From this figure, false-positive rate of PCA method with the true traffic matrix is low. That is, in the case of monitoring traffic matrix accurately, PCA method identifies attack sources accurately. However, false-positive rate of PCA method using estimated traffic matrix is quite high. This is because the method proposed in Ref. [6] cannot estimate the traffic matrix accurately in the case of attacks. Because the increase in traffic matrix estimated by the method proposed in Ref. [6] is proportional to the total rate of traffic monitored at the source, we mistakenly identify sources having large amount of traffic or cannot identify attack sources having small amount of traffic. As a result, PCA method cannot identify attack sources accurately in the case that we can monitor only the link utilizations.

On the other hand, our method can identify attack sources accurately. This is because our method estimates not the total amount of traffic but the increase in traffic. By focusing on the increase in traffic, we can accurately estimate the increase in traffic caused by the attacks and identify attack sources. From this figure, we can also see that our method can identify attack sources more accurately when the attack rate is larger. This is because larger attack causes the significant increase in traffic. As a result, because the increase in traffic caused by the attack is much larger than the time-dependent variations of legitimate traffic, we can identify the sources easier.

This way, to detect attack sources, traditional traffic matrix estimation method is insufficient and we need to use the estimation method focusing on the changes in traffic caused by attacks. In our method, we can identify attack sources accurately by focusing on the increase in traffic.

3.3 Conclusion

In this section, we have proposed a new method for identifying attack sources by estimating traffic matrices. Our method periodically collects link load data from each router through SNMP and estimates the increase in traffic between each source and destination. When attacks start, our method identifies the sources of the attack using the estimated increase. We have also shown that our method can accurately identify attack sources without any false-positives by setting the adequate parameters of γ.

Section 4 Overlay Network Against Distributed SYN Flood Attacks

DDoS attacks prevent the legitimate users from connecting the victim servers by consuming the resources of the servers and networks. Because the attack succeeds when legitimate packets are also dropped even if no attack packets reach the server, we need a method to protect legitimate packets accurately from attack packets.

In this section, 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. We evaluate our defense method by simulation and show that our method can make the probability of dropping legitimate SYN packets less than 0.1 even when the attack rate exceeds 600,000 SYNs/sec.

4.1 Defense Method to Protect Legitimated Traffic from a Domain by Using Overlay Networks

Overview of defense method to protect legitimate traffic

Figure 2.24 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. 2.25). 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 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

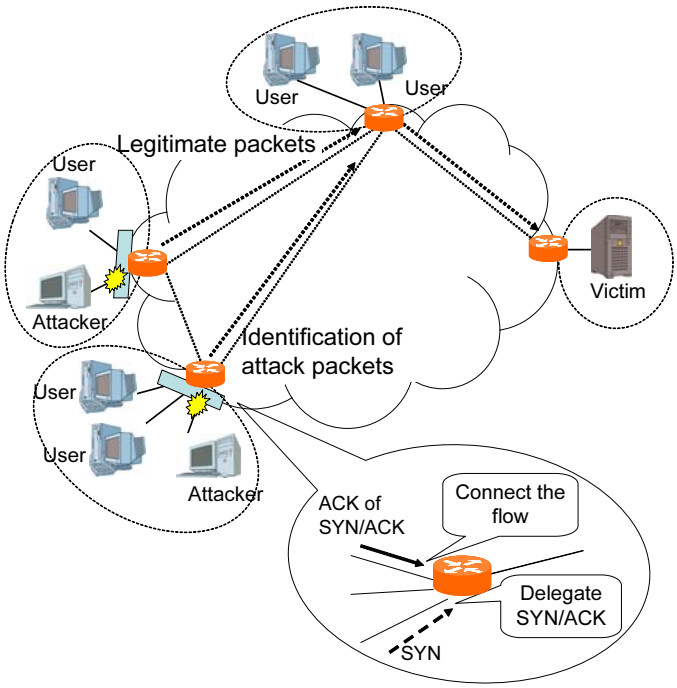


Figure 2.24: Overview of our defense method

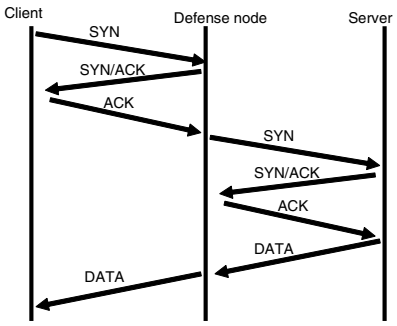


Figure 2.25: Delegation of SYN/ACK packets

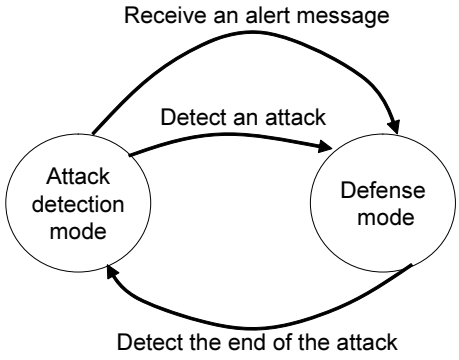


Figure 2.26: State transition diagram between attack detection mode and defense mode

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. 2.26.

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 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 Subsection 4.3, we use the method proposed in Section 2.

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 paragraphs, we describe the details of the above operations.

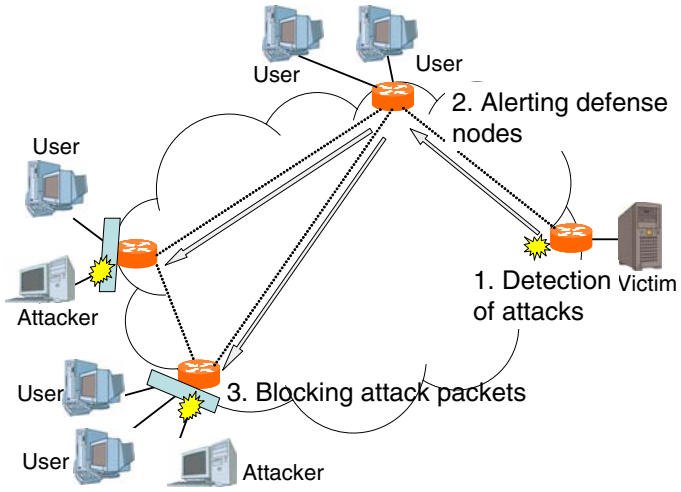


Figure 2.27: Steps to start defense mode

Changing the modes of defense nodes

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 2.27 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 availability.

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

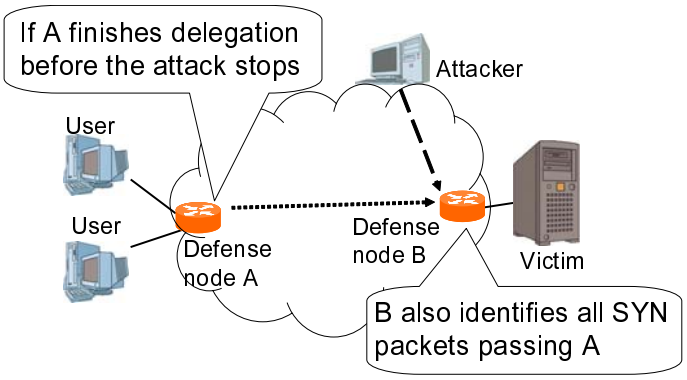


Figure 2.28: Problem in finishing the defense mode

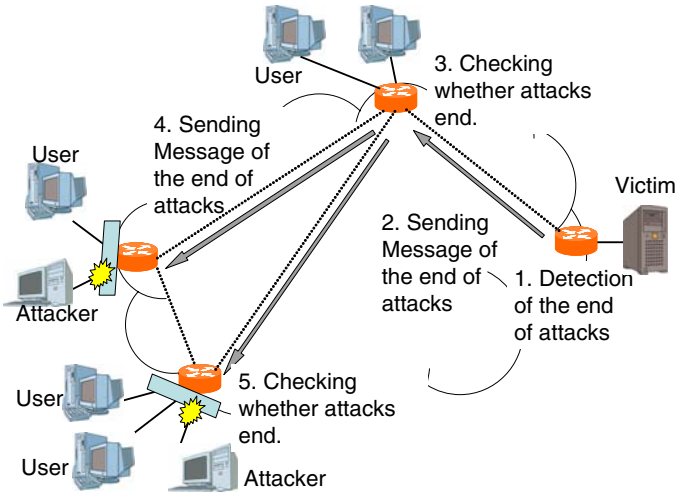


Figure 2.29: Steps to finish the defense mode

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. 2.28. 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 to the victim node.

Figure 2.29 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.

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 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

Table 2.4: 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

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 paragraph, we describe these operations in detail.

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 2.4 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 section, we use the SYN cache [45] 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.

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 [76], 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. Ref. [76] 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 Ref. [77] can search flows quickly by using a hash table. To apply this method to our defense nodes, though, we need to adjust it. While Ref. [77] 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 connections to the next hop at the same time as searching the connections.

Figure 2.30 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.

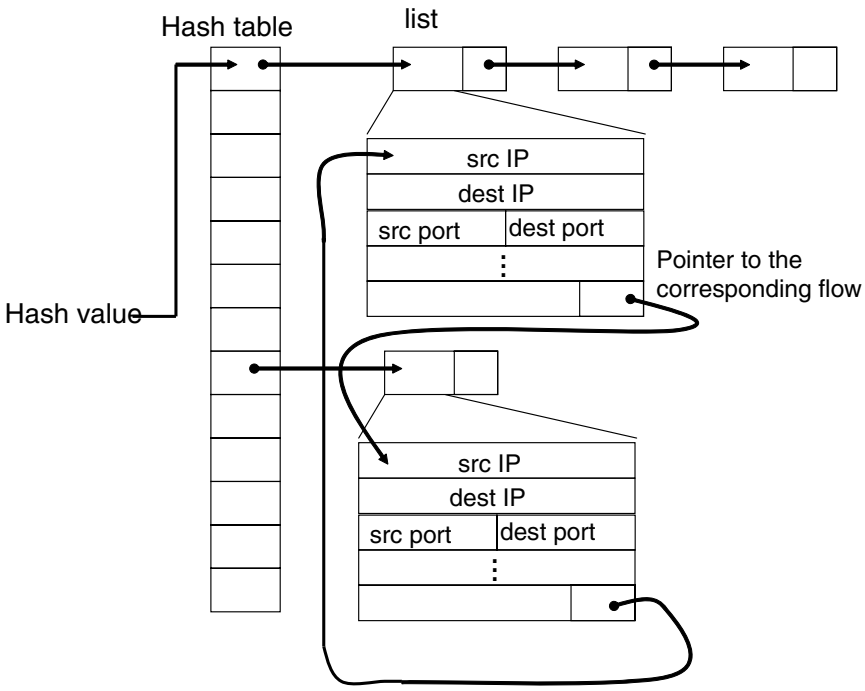


Figure 2.30: Data structure to hold normal flows

The size of data structure shown in Fig. 2.30 is only 240 Bytes per flow because a TCP connection can be held in a 120 Byte control block [78]. We also need sender/receiver buffers. According to Ref. [76], 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 [52–54], 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 [79] or the QoS-Aware routing method [80]. 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 Subsection 4.3, we use the minimum hop routing to simplify the evaluation model.

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. 2.30 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 our method, 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 [52] 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.

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 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 the previous paragraph.

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.

4.2 Deployment Scenario

In this subsection, 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 section, 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*. Figs. 2.31 through 2.33 show the strategic scenario for the deployment of our defense mechanism. There are three stages as follows.

1st stage (Fig.2.31): Only one domain is *protected*. Others are *unprotected*.

2nd stage (Fig.2.32): Several domains are *protected*.

Final stage (Fig.2.33): All domains are *protected*.

At the first stage, we consider our method to be deployed in only one domain, as shown in Fig. 2.31. 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. 2.31, 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. 2.32), our method is deployed in several domains which cooperate with each other. In the case shown in Fig. 2.32, Domain 1, Domain 3, and Domain 6 are

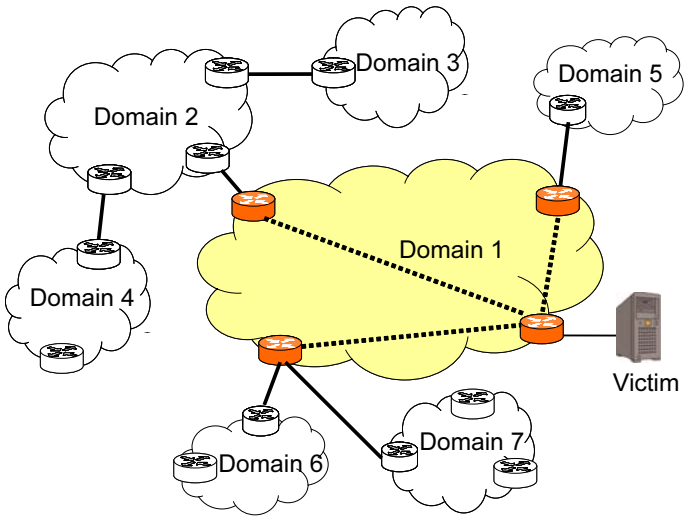


Figure 2.31: First stage of deployment

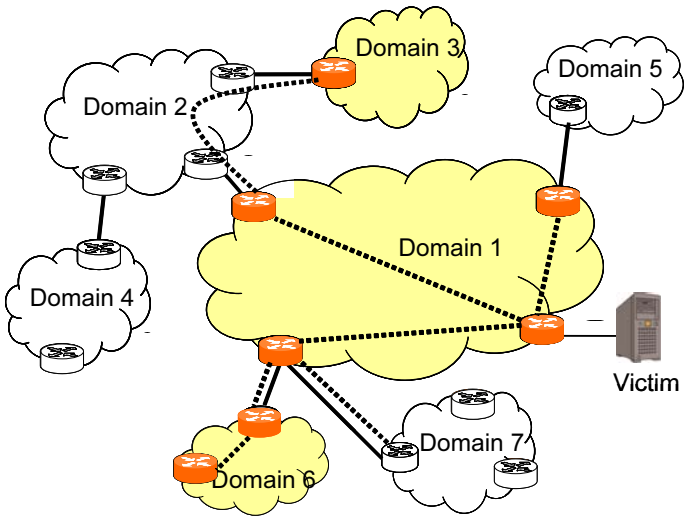


Figure 2.32: Seccond stage of deployment

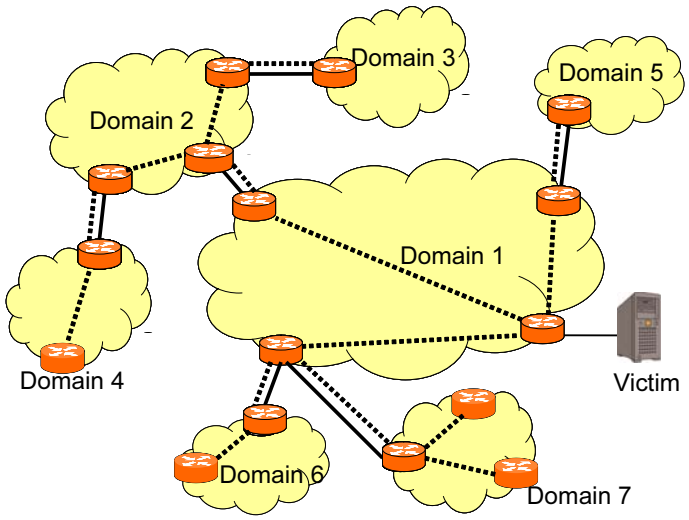
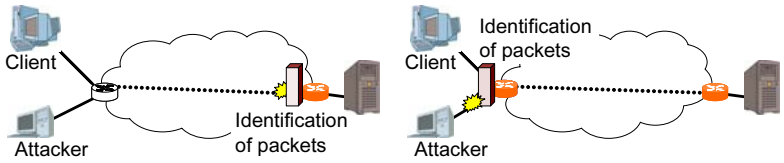


Figure 2.33: Final stage of deployment

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. 2.32). 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. 2.33), all domains are *protected*. In the case shown in Fig. 2.33, 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.



(a) Server-side defense (b) Client-side defense

Figure 2.34: Server-side defense and client side defense

4.3 Evaluation of Defense Method

In this subsection, 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.

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. 2.34(b)) with that when deploying a server-side defense mechanism (Fig. 2.34(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 Section 2, 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 2.35 shows the probabilities of legitimate SYN packets being dropped based on the rate of attack traffic. As shown, the client-side defense protects 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.

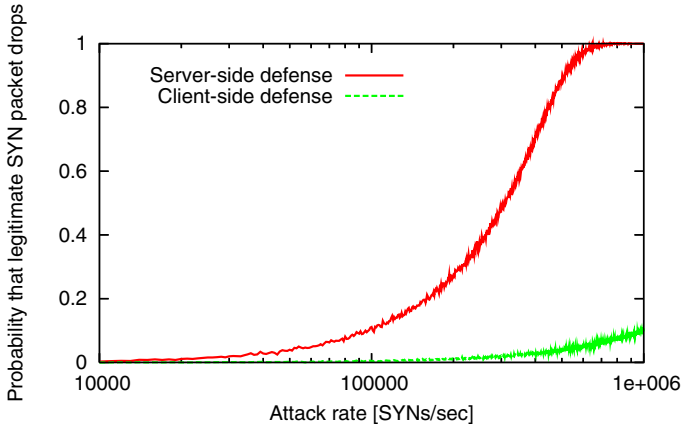


Figure 2.35: Probability of dropping legitimate SYN packets vs. attack rate

Ref. [81] 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.

Effectiveness of our method to protect legitimate traffic

We next considered the effectiveness of our method 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. 2.36. 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.

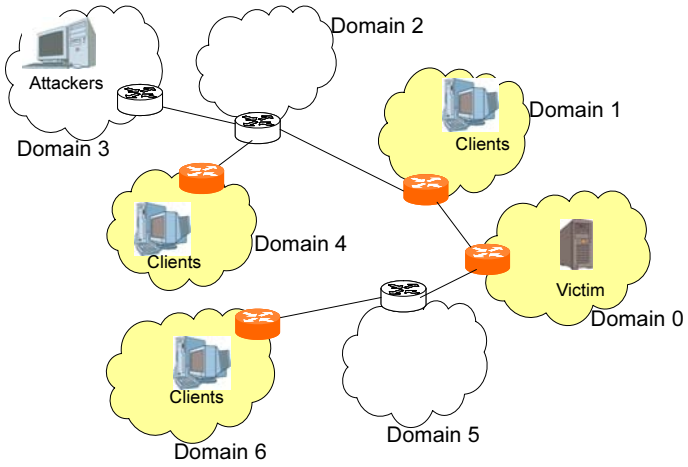


Figure 2.36: Enviroment used in our simulation

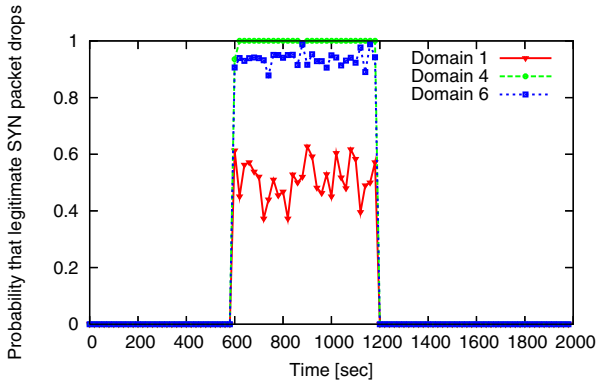
Server-side defense Only server-side defense node (i.e., the defense node deployed at the gateway of Domain 0 in Fig. 2.36) identifies the legitimate packets and blocks attack packets.

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. 2.36, 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.

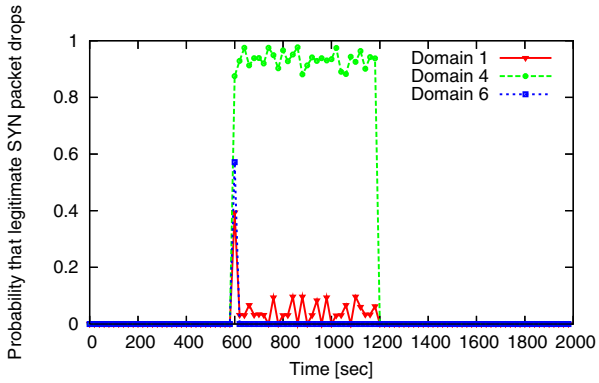
In all cases, defense nodes identify the legitimate traffic by delegating the SYN/ACK packets.

Figure 2.37 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 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.

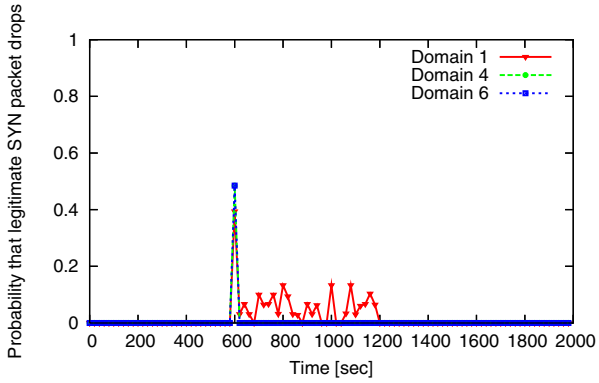
In the case of attacker-side defense, the probabilities of packet loss for Domain 1 and 6 become very low soon after the attack starts. This is because the attacks are immediately detected, and the defense node at Domain 1 begins to distinguish legitimate packets from attack packets. Then, none of the 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



(a) Server-side defense



(b) Attacker-side defense



(c) Our method

Figure 2.37: Probability of dropping legitimate SYN packets (when attack rate is constant)

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).

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 paragraph. 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. Fig. 2.38 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 paragraph. 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.

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.

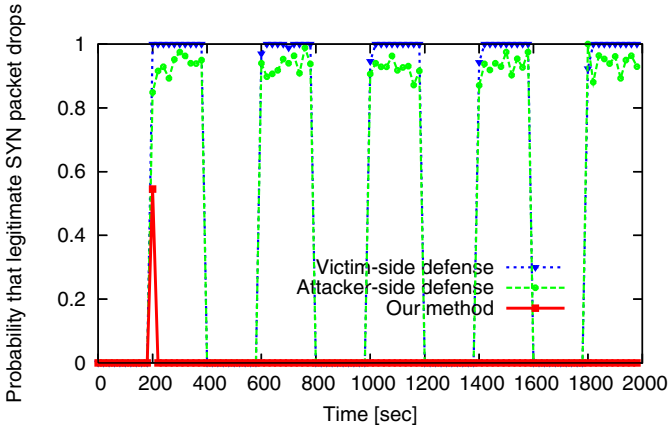
Loads on defense nodes

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 of the case of Fig. 2.37 and the length of the legitimate TCP connections was set so as to follow the Pareto distribution whose mean is 5 sec.

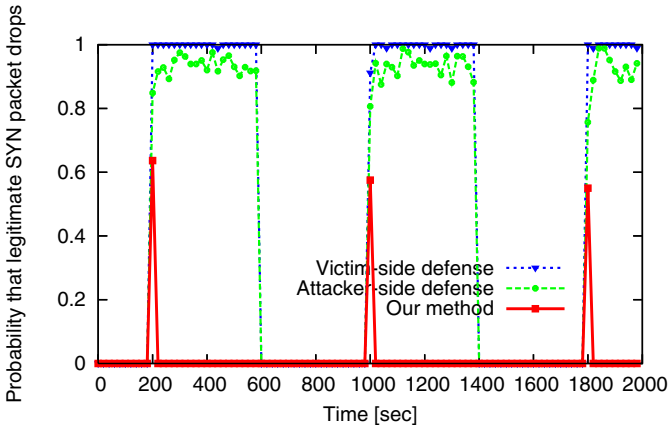
Figure 2.39 compares the number of TCP connections held by the defense node at Domain 6 in following three cases, our method with the extension described in Subsection 4.1, our method without the extension, and a method similar to SOS [52–54] which does not stop the identification of legitimate traffic 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 4.1 does not require



(a) Attack rate changes every 200 sec



(b) Attack rate changes every 400 sec

Figure 2.38: Probability of dropping legitimate SYN packets (the case of pulsing attacks)

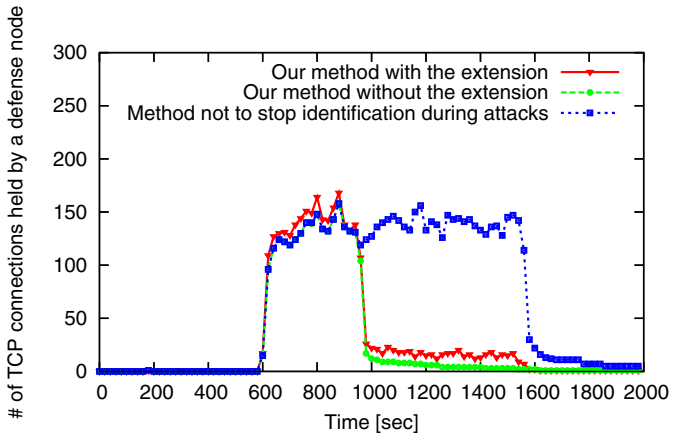


Figure 2.39: Number of TCP connections held by a defense node

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.

4.4 Conclusion

In this section, 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.

Chapter 3

Measurement, Estimation and Topology Control to Changes of Traffic

Network operators design their networks according to the predicted traffic so as to accommodate all traffic efficiently (e.g., without congestion or large delays). However, if the current traffic significantly differ from the predicted one due to the changes of legitimate traffic, the previously constructed network becomes no longer suitable to the current traffic.

Optical layer traffic engineering (TE) [59–66] is one efficient way of accommodating traffic that changes unpredictably. Optical layer TE accommodates time-varying traffic by dynamically reconfiguring VNTs which are formed by the optical layer paths.

In optical layer TE, a traffic matrix, which indicates traffic volumes between all pairs of edge nodes, is required as an input. By using the traffic matrix, the VNT reconfiguration methods configure a new VNT in which constraints such as maximum utilization of optical layer paths are satisfied. Because it is difficult to monitor traffic matrices directly, several methods for estimating them from link loads [1–10] have been proposed. However, estimated traffic matrix includes estimation errors which degrade the performance of TE significantly.

Therefore, in this chapter, we propose methods which estimate traffic matrices accurately and reconfigure the adequate network settings by cooperating each other.

Section 5 Gradual Reconfiguration of Virtual Network Topology

Traffic matrix is essential to traffic engineering (TE) methods. Because it is difficult to monitor traffic matrices directly, several methods for estimating them from link loads have been proposed. However, estimated traffic matrix includes estimation errors which degrade the performance of TE significantly. In this section, we propose a method that reduces estimation errors while reconfiguring the VNT.

To reduce the estimation errors, we propose a gradual reconfiguration method in which the VNT reconfiguration is divided into multiple stages instead of reconfiguring the suitable VNT at once. By dividing the VNT reconfiguration into multiple stages, our traffic matrix estimation method calibrates and reduces the estimation errors in each stage by using information monitored in prior stages.

We also investigate the effectiveness of our gradual reconfiguration using simulations. According to the results, the gradual reconfiguration can reduce the root mean squared relative error (RMSRE) to 0.1 and achieve an adequate VNT as is the case with the reconfiguration using the actual traffic matrices.

5.1 Terminology

Before presenting overviews of our gradual reconfiguration, we explain our terminology.

Traffic matrix

A matrix indicating the amount of traffic between all pairs of IP routers.

Physical topology

A topology physically constructed in the optical layer that consists of OXCs and WDM optical fibers. Two OXCs are connected by a single optical fiber.

Optical layer path

A lightpath configured between two indirectly/directly connected OXCs. An optical layer path is a set of optical fibers between the two OXCs determined by the optical layer TE. An optical layer path occupies one wavelength of each optical fiber on the route of the optical layer path.

VNT

A topology constructed with optical layer paths. From the packet layer, an optical layer path is regarded as a single directly connected link between IP routers.

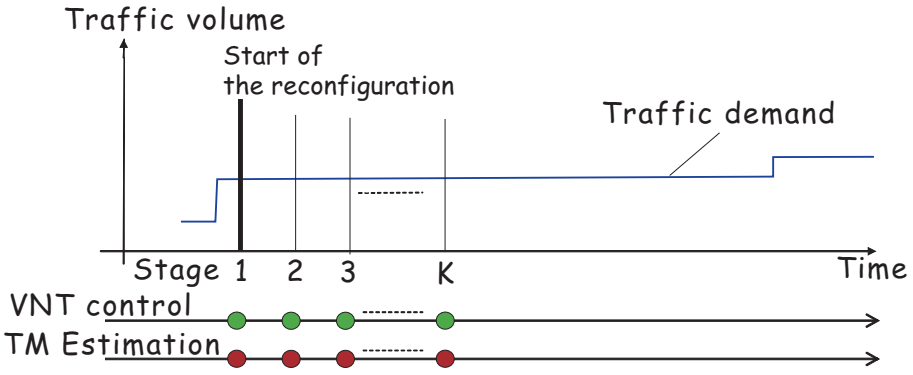


Figure 3.1: Overview of gradual reconfiguration of VNT

Packet layer path

An end-to-end packet-layer traffic traversing the VNT. Packet layer paths traversing the same optical layer path share the optical layer path bandwidth.

Route of a packet layer path

A set of optical layer paths passed by the packet layer path.

Utilization of an optical layer path

Amount of traffic traversing the optical layer path divided by the capacity of the optical layer path.

5.2 Overview of Gradual Reconfiguration

Our goal is to develop a method that reduces the estimation errors while reconfiguring the VNT by cooperating with the VNT reconfiguration. To achieve this goal, we deploy the gradual reconfiguration approach shown in Fig. 3.1. First, when the VNT reconfiguration is needed (e.g., the link utilizations exceed a threshold), we start the gradual reconfiguration. In the gradual reconfiguration, the VNT reconfiguration is divided into several stages. In each stage of the gradual reconfiguration, we assume that routes of both packet and optical layer paths are calculated by a path computation element (PCE) [82]. Then, the OXCs and the routers in the network are configured according to the calculated routes.

The operations performed in each stage are shown in Fig. 3.2. At the beginning of each stage, the PCE collects the monitored link loads and estimates the traffic matrix from them. Then, the

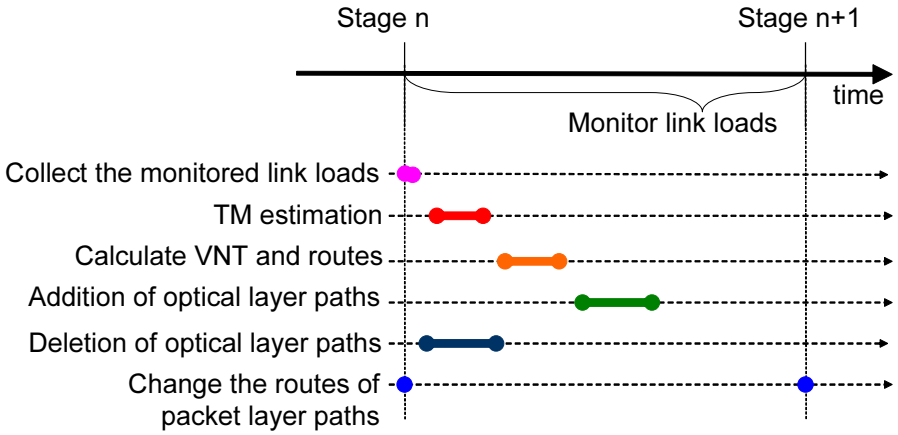


Figure 3.2: Operations in each stage

PCE calculates the VNT and the routes of packet layer paths by using the estimated traffic matrix. The calculated routes are used from the beginning of the next stage.

However, in order to avoid dropping packets, the addition of optical layer paths should be done before the change of packet layer paths and the deletion of optical layer paths should be done after the change of packet layer paths. Thus, in each stage, optical layer paths not included in the VNT of the current stage are deleted after the change of routes of packet layer paths at the beginning of the current stage. Optical layer paths used at the next stage are added in advance by using the resources not used at the current stage after the calculation of the VNT for the next stage.

By iteratively performing the above operations, we reconfigure more suitable VNT as stages go on. Finally, when the goal of the reconfiguration (e.g., making the maximum link utilization less than a threshold) is achieved, the gradual reconfiguration is finished. Once the gradual reconfiguration is finished, the VNT is fixed unless another VNT reconfiguration is needed. Therefore, the performance degradations which may occur when changing the routes are only temporary.

When estimating the traffic matrix at each stage, we use the additional information obtained by monitoring the amount of traffic on each optical layer path both before and after the VNT reconfiguration. By using the additional information, our estimation method improves the accuracy of the estimation. Then, the reconfiguration method can use more accurate traffic matrix at the next stage.

The basic idea of our gradual reconfiguration is to avoid adding or deleting many optical layer paths before the estimation errors are sufficiently reduced. Thus, any partial reconfiguration method can be applied to our gradual reconfiguration. The challenges are how to reduce the estimation

errors of traffic matrices during the gradual reconfiguration. We discuss the details of our estimation method in Subsection 5.3.

5.3 Traffic Matrix Estimation Method Suitable for Gradual Reconfiguration

When the VNT is reconfigured, several routes of packet layer paths are also changed. The change in routes of packet layer paths directly impacts the amounts of traffic on optical layer paths that are passed by the packet layer paths whose routes are changed. Assuming that the network is stable (i.e., the variation in traffic is small) between two continuous stages, if the measured amount of traffic on an optical layer path is changed by VNT reconfiguration, it is safe to conclude that the difference is caused by the change in routes of packet layer paths. These differences can yield additional equations for solving the traffic matrix calculation.

Hereafter, we call our estimation method the *additional equation method*. In this subsection, we first describe its basic idea. In a real network, the traffic may change from the beginning of the reconfiguration. A significant variation in traffic causes estimation error because it violates the fundamental assumption that the network is stable. Thus, to deal with obvious variations in traffic, we propose a method of eliminating the impact of the non-negligible change in traffic. In addition, we propose a method to reduce the size of the data of the previous stages in order to save the resources such as CPU and memories of a PCE.

Basic Idea

In our method, we use the amounts of traffic on optical layer paths monitored from the beginning of the reconfiguration. That is, to estimate the traffic matrix at stage n, we use the amounts of traffic monitored from stage 0 to stage n. In stage i, T_i, A_i, and X_i denote the actual traffic matrix, the routing matrix (i.e., the matrix in which an element corresponding to a packet layer path and an optical layer path is 1 if the packet layer path passes the optical layer path or is set to 0 if not) and the matrix indicating amounts of traffic on optical layer paths, respectively.

Because amount of traffic on an optical layer path is the sum of the traffic for the packet layer paths using the optical layer path, we have

$$\begin{aligned} X_i &= A_i T_i \\ &= A_i T_n + \epsilon_{i,n}, \end{aligned} \tag{3.1}$$

where $\epsilon_{i,n}$ denotes the change in traffic between stages i and n. At stage n, by combining all

relations from X_0 to X_n, we also have

$$\begin{bmatrix} X_0 \\ \vdots \\ X_i \\ \vdots \\ X_{n-1} \\ X_n \end{bmatrix} = \begin{bmatrix} A_0 \\ \vdots \\ A_i \\ \vdots \\ A_{n-1} \\ A_n \end{bmatrix} T_n + \begin{bmatrix} \epsilon_{0,n} \\ \vdots \\ \epsilon_{i,n} \\ \vdots \\ \epsilon_{n-1,n} \\ 0 \end{bmatrix}. \tag{3.2}$$

According to Ref. [83], backbone IP traffic is stationary with a period of 1 to 1.5 hours. Thus, unless the gradual reconfiguration takes more than 1.5 hours, $\epsilon_{i,n}$ is considered to be small.

To estimate the traffic matrix $\hat{T}_n$ from Eq. (3.2), we apply the pseudo-inverse calculation method described in [84]. The traffic matrix $\hat{T}_n$ is obtained from

$$\hat{T}_n = \bar{A}_n^+ \bar{X}_n, \tag{3.3}$$

where $\bar{A}_n^+$ is the pseudo-inverse of matrix $\bar{A}_n$, and $\bar{A}_n$ and $\bar{X}_n$ are the matrices defined as,

$$\bar{X}_n = \begin{bmatrix} X_0 \\ \vdots \\ X_i \\ \vdots \\ X_n \end{bmatrix} \tag{3.4}$$

and

$$\bar{A}_n = \begin{bmatrix} A_0 \\ \vdots \\ A_i \\ \vdots \\ A_n \end{bmatrix}. \tag{3.5}$$

A pseudo-inverse matrix is a generalized inverse matrix and by using pseudo-inverse matrix, Eq. (3.3) can estimate $\hat{T}_n$ so as to minimize the squared sum of $\epsilon_{i,n} (0 \leq i \leq n)$.

However, if we simply apply the pseudo-inverse of $\bar{A}_n$ to solve Eq. (3.3), some elements in $\hat{T}_n$ may have negative values, which are nonexistent as regards the traffic matrix. The following

iteration eliminates such negative values. We define the estimated traffic matrix for the i-th iteration as $\hat{T}_n^{(i)}$.

Step 1 Let $\hat{T}_n^{(0)} \leftarrow \hat{T}_n$

Step 2 Calculate $\hat{T}_n^{(i)}$ from $\hat{T}_n^{(i-1)}$ by using

$$\hat{T}_n^{(i)} = \hat{T}_n^{(i-1)} + \bar{A}_n^{+}(\bar{X}_n - \bar{A}_n \hat{T}_n^{\prime(i-1)}), \tag{3.6}$$

where $\hat{T}_n^{\prime(i)}$ is a matrix in which we replace all negative values of $\hat{T}_n^{(i)}$ with zero.

Step 3 If all elements in $\hat{T}_n^{(i)}$ are non-negative values, go to *Step 4*, or else back to *Step 2*.

Step 4 Let $\hat{T}_n^{(i)}$ be the final result of traffic matrix $\hat{T}_n$

Dealing with non-negligible changes in traffic

The basic idea described above assumes that the network is stable (i.e., the variation in traffic is small). However, in real networks, the traffic may change from the beginning of the reconfiguration. Significant variation in traffic causes estimation error because it violates the fundamental assumption of the additional equation method.

Therefore, we remove the information about non-negligible change in traffic from the information monitored at previous stages in order to avoid violating the assumption. Our proposed estimation method contains the following steps.

Step 1 Identify the packet layer paths including non-negligible changes

Step 2 Remove the information about the traffic of the identified paths from the information monitored at previous stages

Step 3 Estimate the traffic matrix by using the information in which information about the traffic of the identified paths is removed

In the rest of this paragraph, we describe these steps in detail.

Identify the packet layer paths, including non-negligible change First, we identify the packet layer paths including non-negligible changes. To do this, we use the method proposed in Section 3,

which identifies source nodes of DDoS attacks based on increase in traffic. We identify the packet layer paths including non-negligible changes in stage n as follows.

First, we calculate differences D_n between the amounts of traffic monitored at stage n and the utilization forecasted using the estimated traffic matrix from stage $n-1$.

$$D_n = X_n - A_n \hat{T}_{n-1} \tag{3.7}$$

Then, we estimate matrix G_n indicating the increases in traffic flows between all pairs of edge nodes using D_n. Finally, if there are elements in G_n that are larger than a threshold Γ, we identify the packet layer paths that correspond to the elements as the paths that include non-negligible changes.

Though this step requires estimation of G_n, we do not have to estimate it accurately. The aim of this step is to identify packet layer paths that include traffic flows increasing significantly more than others. Therefore, when estimating G_n, we have only to estimate the elements corresponding to the packet layer paths with significantly increasing traffic as large values. In this study, we used the tomogravity method to estimate G_n from D_n. In the tomogravity method, the elements of G_n are estimated as the value proportional to the increase of incoming/outgoing traffic for each edge node. Because elements in D_n corresponding to the incoming/outgoing traffic for the source and destination nodes of the packet layer paths with significantly increasing traffic are large, the tomogravity method can estimate the elements in G_n that correspond to the packet layer paths with significantly increasing traffic as large values.

Removal of information about non-negligible changes We remove the information about packet layer paths with non-negligible changes from the information monitored in the previous stages as follows.

We first remove the information of the identified packet layer paths from the routing matrices A_i by replacing the elements corresponding to the identified paths by 0. We denote the routing matrix after the replacement as A'_i, in which the element corresponding to the packet layer path from n to m and the optical layer path between k and l is given by

$$a_i^{'n,m,k,l} = \begin{cases} 0, & \text{if traffic from } n \text{ to } m \text{ changes significantly} \\ a_i^{n,m,k,l}, & \text{otherwise} \end{cases}, \tag{3.8}$$

where $a_i^{n,m,k,l}$ is the element of A_i indicating whether or not the packet layer path from n to m passes the optical layer path between k and l.

Then, we create the matrix of traffic amount X'_i in which the information about the identified packet layer paths are removed, which is given by

$$X'_i = X_i - (A_i - A'_i)\hat{T}_{n-1}. \tag{3.9}$$

In this equation, $(A_i - A'_i)\hat{T}_{n-1}$ indicates the matrix of the identified traffic on each link calculated by using the traffic matrix estimated at stage $n-1$.

Estimate the traffic matrix To estimate the traffic matrix, we use the equation,

$$X'_i = A'_i T_n + \epsilon_{i,n} \tag{3.10}$$

instead of Eq. (3.1). Similar to Eq. (3.2), by combining all relations from X_0 to X_n, we also have

$$\begin{bmatrix} X'_0 \\ \vdots \\ X'_i \\ \vdots \\ X'_{n-1} \\ X_n \end{bmatrix} = \begin{bmatrix} A'_0 \\ \vdots \\ A'_i \\ \vdots \\ A'_{n-1} \\ A_n \end{bmatrix} T_n + \begin{bmatrix} \epsilon_{0,n} \\ \vdots \\ \epsilon_{i,n} \\ \vdots \\ \epsilon_{n-1,n} \\ 0 \end{bmatrix}. \tag{3.11}$$

Because we remove the information about the traffic with non-negligible changes, X'_i does not include the information about the traffic with non-negligible changes. That is, $\epsilon_{i,n}$ is small. Therefore, as in Eq. (3.3), we can estimate $\hat{T}_n$ as

$$\hat{T}_n = \bar{A'}^{+}_{n} \bar{X'}_n, \tag{3.12}$$

where $\bar{A'}^{+}_{n}$ is the pseudo-inverse of the matrix $\bar{A'}_n$, and $\bar{A'}_n$ and $\bar{X'}_n$ are the matrices defined as

$$\bar{X'}_n = \begin{bmatrix} X'_0 \\ \vdots \\ X'_i \\ \vdots \\ X'_n \end{bmatrix} \tag{3.13}$$

and

$$\bar{A'}_n = \begin{bmatrix} A'_0 \\ \vdots \\ A'_i \\ \vdots \\ A'_n \end{bmatrix}. \tag{3.14}$$

Reduction of the size of matrix

Partial reconfiguration changes only a small number of paths at each stage. That is, most elements of $\Delta A'_i = A'_i - A'_{i-1}$ are expected to be 0. Thus, we introduce following procedure that reduces the size of the matrix.

First, we transform X'_i and A'_i into the matrices, $\Delta X'_i = X'_i - X'_{i-1}$ and $\Delta A'_i = A'_i - A'_{i-1}$. Second, we remove rows $\Delta a_i^{'(j)}$ in $\Delta A'_i$ if $\Delta a_i^{'(j)}$ = O. We also remove the same position of the row $\Delta x_i^{'(j)}$ in ΔX_i. We denote $\Delta X'_i$ and $\Delta A'_i$ after row removals as $\Delta X''_i$ and A''_i respectively. Finally, we use

$$\hat{T}_n = \Delta \bar{A''}_n{}^{+} \Delta \bar{X''}_n, \tag{3.15}$$

where $\Delta \bar{A''}_n{}^{+}$ is the pseudo-inverse of the matrix $\Delta \bar{A''}_n$, and $\Delta \bar{A''}_n$ and $\Delta \bar{X''}_n$ are the matrices defined as

$$\Delta \bar{X''}_n = \begin{bmatrix} \Delta X''_1 \\ \vdots \\ \Delta X''_i \\ \vdots \\ \Delta X''_n \\ X'_n \end{bmatrix} \tag{3.16}$$

and

$$\Delta \bar{A''}_n = \begin{bmatrix} \Delta A''_1 \\ \vdots \\ \Delta A''_i \\ \vdots \\ \Delta A''_n \\ A'_n \end{bmatrix}. \tag{3.17}$$

instead of Eq. (3.12)

5.4 Evaluation of Gradual Reconfiguration

In this subsection, we describe the simulation conditions, explain how the simulations demonstrate the effectiveness of the gradual reconfiguration method with the existing traffic matrix estimation, and evaluate the gradual reconfiguration method using the additional equation method.

The partial reconfiguration method

Our gradual reconfiguration can use any partial reconfiguration method at each stage. In this evaluation, we use a heuristic method proposed in Ref. [66]. This method adds a new optical layer path to mitigate congestion and, if possible, deletes a currently underutilized optical layer path for reclamation. This method allows only one optical layer path to be added or deleted in each stage. However, in our method, the number of optical layer paths added or deleted at each stage may affect the number of additional equations and the estimation errors. We therefore extend the reconfiguration methods in Ref. [66] such that it can add or delete multiple optical layer paths at each stage. N denotes the maximum number of paths to be added or deleted. We set $N = \infty$ to obtain results for the full reconfigurations.

The extended method uses two thresholds for the utilization of each optical layer path to define the *congested* and *underutilized* states. T_H and T_L denote thresholds for *congested* and *underutilized*, respectively. In our evaluations, we set T_L to $0.5 \times T_H$. The general sequence of the algorithm to calculate the VNT at each stage is as follows:

Step 1 Check the utilization of all optical layer paths. If at least one congested optical layer path (i.e., a path whose utilization exceeds the threshold T_H) is found, go to the optical layer path addition phase (Step 2). If there is an optical layer path whose utilization is less than threshold T_L, go to *Step 3*

Step 2 Execute the *optical layer path addition phase* described below, and then go to *Step 4*.

Step 3 Execute the *optical layer path deletion phase* described below, and then go to *Step 4*.

Step 4 Calculate the routes of packet layer paths over the new VNT and obtain the expected utilization of all optical layer paths of the new VNT.

Step 5 Decrement the number of optical layer paths to be added/deleted in this stage, i.e., N = $N - 1$. If $N = 0$, go to *End*. Otherwise, go back to *Step 1*.

End

Finally, when utilizations of all optical layer paths become less than T_H and any optical layer paths cannot be deleted, the gradual reconfiguration finishes.

In the above steps, the routes of packet layer paths over the VNT are calculated so as to make the maximum link utilizations less than $T_H - \eta$, where η indicates the margin for the fluctuations of traffic. In our evaluations, we set η to $0.1 \times T_H$. The calculation of the routes of packet layer paths is performed as follows. 1) When utilizations of all optical layer paths on the route of a packet layer path in the previous stage are lower than $T_H - \eta$, the route is kept in the current stage because there is no need to change it. 2) Otherwise, the route is calculated using constraint-based shortest path first (CSPF) [85] so as to limit maximum utilizations to less than $T_H - \eta$.

The details of optical layer path addition/deletion phases are as follows.

1. *Optical layer path addition phase:*
   If the utilization of an optical layer path exceeds T_H, a new optical layer path is set up to reroute traffic away from the *congested* optical layer path. First, we collect a set of packet layer paths that pass the most congested optical layer path. Then, we select the busiest of the collected packet layer paths. Finally, we add the direct optical layer path (i.e., a single directly connected link) from ingress to egress nodes of the selected packet layer path.

2. *Optical layer path deletion phase:*
   If the utilization of an optical layer path is less than T_L and the deletion of the optical layer path is shown not to cause congestion, the path is torn down so the IP router ports and wavelengths can be reclaimed for future use. The optical layer path is checked for potential for its deletion to cause congestion by calculating the utilization of optical layer paths after deletion using the traffic matrix estimated in the current stage. If there is more than one candidate for deletion, each candidate path is tested in ascending order of utilization.

In our evaluations, we implement the above algorithm and investigate the VNT reconfigured at each stage by using C++.

Simulation Conditions

In our simulation, we use the European Optical Network (EON) (19 nodes, 37 links) shown in Fig. 3.3 as the physical topology. In this figure, circles represent OXCs, and lines represent optical fibers. The number of wavelengths for each optical fiber is set to 16.

In our simulation, we implement estimation methods by using LAPACK [86] which is a software library for numerical computing. In the additional equation method, we use a parameter, Γ

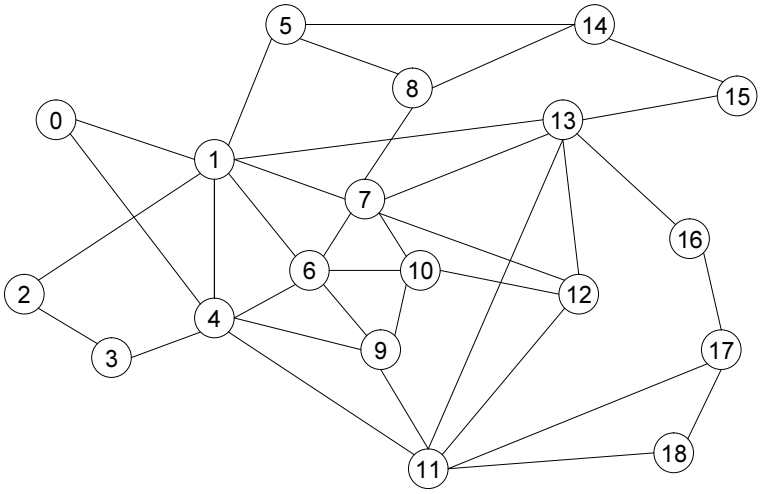


Figure 3.3: EON topology

which indicates sensitivity for detection of change in traffic; the traffic whose increase is larger than Γ is identified as traffic including non-negligible changes, and the previously monitored information about the traffic is not used for estimating traffic matrix. If we set Γ to too large a value, there is still a non-negligible change in traffic in the information used by the additional equation method. This causes estimation errors. However, a too small Γ causes misdetection of traffic with no non-negligible changes (false positives). As a result, the additional equation method cannot use many of the amounts of traffic monitored in the previous stages. Therefore, we should set the Γ to as small a value as possible such that it will not detect traffic with no non-negligible change by monitoring the traffic in advance. In our simulations, Γ is set to $0.2 \times T_H$ times of the bandwidth of an optical layer path.

In our evaluations, we generate the initial traffic matrix where the elements of it follow the lognormal distribution [83]. We set the parameters of lognormal distribution to be the same in Ref. [83], and scale each element such that its average is $0.3 \times T_H$. The initial VNT is configured using the initial traffic matrix by the above VNT reconfiguration method with $N = \infty$. The changes are randomly generated within the range from -0.4 to 0.4 times the elements of the initial traffic matrix, based on the observation that, in the Abilene [87], the amount of traffic increases about 1.4 times in 2 hours when the daily change of the traffic is the largest.

In our evaluations described below, we first use the static traffic matrix during the gradual reconfiguration so as to focus on the impact of estimation errors. Then, we use the fluctuating traffic matrix as more realistic case.

The case without change of traffic

In this paragraph, to focus on the impact of estimation errors, we simulate the ideal case that the traffic is constant in all stages.

Improvement of accuracy of the estimation First, we compare the accuracy of our estimation method with tomogravity method [6] where N is set to 1, 3, and 5. We conducted the simulations using ten different traffic matrices, and the averaged results are presented in following figures. In our simulation, the estimated traffic matrix is obtained in 20 sec at stage 30 by using a computer with 2.66 GHz Intel XEON Processor and 4 GB RAM.

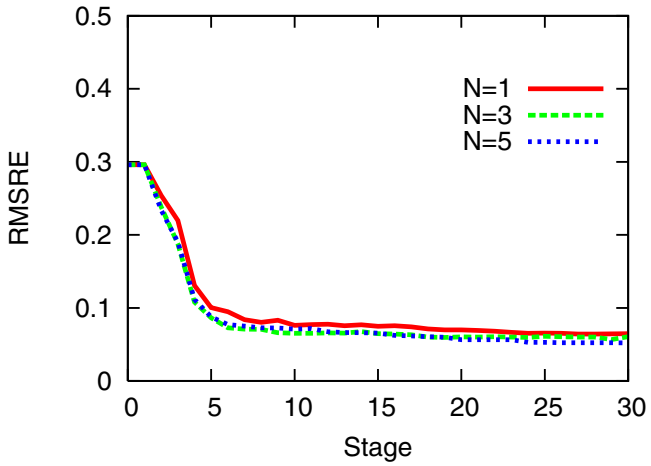
To compare the accuracy of the estimated traffic matrices, we use the root mean squared relative error (RMSRE) as follows,

$$\mathrm{RMSRE} = \sqrt{\frac{1}{N_{\tilde{t}}^2} \sum_{1 \le i,j \le N, t_{i,j} > \tilde{t}} \left(\frac{\hat{t}_{i,j}(n) - t_{i,j}(n)}{t_{i,j}(n)} \right)^2} \tag{3.18}$$

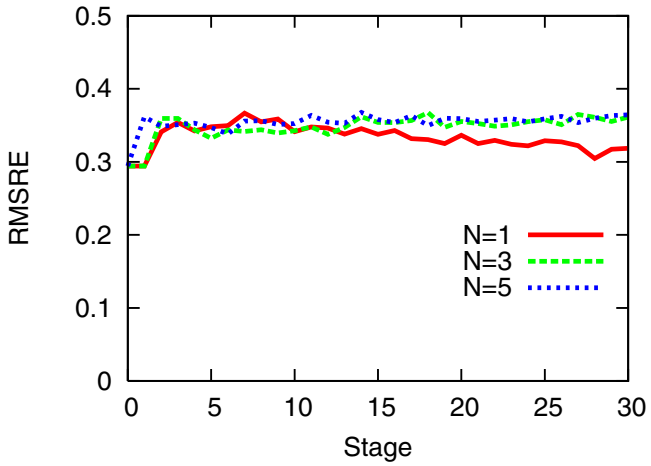
where $\hat{t}_{i,j}$ and $t_{i,j}$ are the estimated and actual amount of traffic from i to j, respectively. The RMSRE gives a relative measure. For small matrix elements, however, the relative errors are not really important. Thus, in computing the RMSRE, we consider only matrix elements greater than a threshold $\tilde{t}$. $N_{\tilde{t}}$ is the number of elements greater than $\tilde{t}$ in a traffic matrix. In the following simulations, $\tilde{t}$ was set so that the sum of the end-to-end traffic whose actual rate was greater than $\tilde{t}$ composed 75 % of the total traffic.

Figure 3.4 shows the RMSREs for each stage. In this graph, the horizontal axis represents the number of stages after the beginning of the VNT reconfiguration, and the vertical axis represents the RMSREs. As can be seen in this figure, the additional equation method reduces estimation errors as the stages are completed, while estimation errors of the tomogravity method are large in all stages. This is caused by the difference in the number of equations used in traffic matrix calculation. The tomogravity method uses only the amounts of traffic monitored at each stage. That is, the tomogravity method uses only the same number of equations as the number of optical layer paths. On the other hand, when some routes of packet layer paths are changed, the additional equation method adds the equations about the packet layer paths whose routes are changed. As a result, the number of equations used by the additional equation method increases as it progresses through the stages.

Figure 3.4 also shows that the estimation errors are reduced to the same level regardless of



(a) Additional equation method



(b) Tomogravity

Figure 3.4: RMSREs of each stage (the case without changes of traffic)

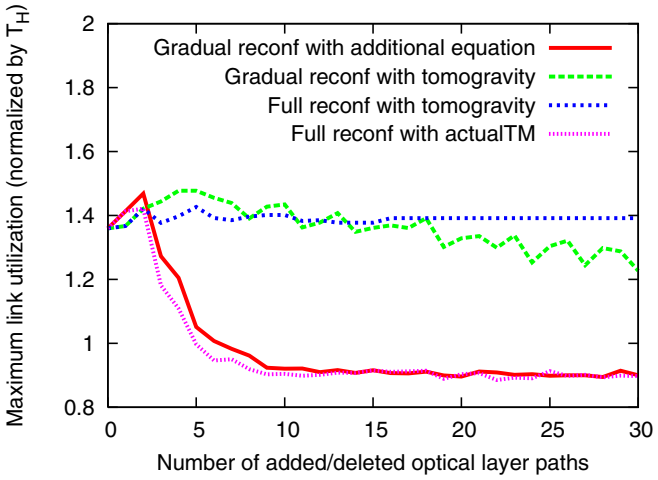


Figure 3.5: Number of added/deleted paths vs maximum utilization

N at each stage. This is because the number of packet layer paths whose routes are changed are almost the same regardless of N. In this simulation, a packet layer path is routed on the VNT according to the following policy; when a packet layer path can be accommodated to the same route as the previous stages without causing utilizations higher than $T_H - \eta$, the packet layer path is accommodated on the same route as the previous stage, otherwise the routes are re-calculated using CSPF so as to limit the maximum link utilization to less than $T_H - \eta$. When one or more optical layer paths are added, the packet layer path traversing the optical layer paths whose utilizations are higher than $T_H - \eta$ moves to the optical layer paths whose utilizations are low. Because the number of packet layer paths traversing optical layer paths whose utilizations are higher than $T_H - \eta$ before the VNT reconfiguration is the same regardless of N, the number of packet layer paths whose routes are changed is also almost the same regardless of N.

Effectiveness of gradual reconfiguration In this paragraph, we investigate the VNT reconfigured by our gradual reconfiguration. First, we compare the following four methods.

- Gradual reconfiguration using the additional equation method
- Gradual reconfiguration using the tomogravity method
- Full reconfiguration using the traffic matrix estimated by tomogravity method.
- Full reconfiguration using the actual traffic matrix (i.e., ideal case)

Figure 3.5 compares the maximum utilization when optical layer paths of the same number are added or deleted. In this figure, the horizontal axis represents the number of added or deleted optical layer paths and the vertical axis represents maximum utilization normalized by the threshold T_H. We set N to 1 for the gradual reconfigurations.

From this figure, full reconfiguration using the traffic matrix estimated by tomogravity method can never make the link utilization less than T_H. This is because the full reconfiguration decides whether additional optical layer paths are needed based on utilization calculated using the traffic matrix estimated at the beginning of the VNT reconfiguration. Therefore, the optical layer path whose actual utilization is more than T_H is mistakenly identified as a path whose utilization is less than T_H. As a result, though the maximum utilization is still over the threshold, the VNT reconfiguration is completed.

In the gradual reconfiguration, because we re-estimate traffic matrix based on the monitored link loads at each stage, the VNT reconfiguration is never completed until the maximum link utilization becomes less than T_H. However, the gradual reconfiguration using the tomogravity method cannot make the maximum utilization less than T_H due to estimation errors of the tomogravity.

On the other hand, similarly to the case using the actual traffic matrices, the gradual reconfiguration using the traffic matrix estimated by the additional equation method can reconfigure an adequate VNT whose maximum utilization is under T_H because the additional equation method can reduce the estimation errors as it progresses through the stages.

Next, we investigate the impact of parameter N on the gradual reconfiguration using the additional equation method. Figure 3.6 compares the maximum utilization normalized by T_H in the case of setting N to 1, 3 and 5. From this figure, regardless to N, we can reduce the maximum utilization as stages go on. In each stage, though the maximum link utilization of the case of $N = 3$ or $N = 5$ is slightly less than that of the case of $N = 1$, the difference is only small. This is because at the early stages before the estimation errors are not reduced enough, we cannot set the VNT suitable to the current traffic due to estimation errors. As a result, we cannot reduce link utilizations as much as expected even when adding more optical layer paths.

.

Figure 3.7 shows the number of optical layer paths added or deleted by the gradual reconfiguration using the additional equation method until the VNT becomes stable. In this figure, we compare the cases where $N = 1$, 3, and 5. In this figure, if we set N to a smaller value, we can reconfigure the adequate VNT by adding or deleting a small number of optical layer paths. Especially, if we set N to 1, the number of added or deleted optical layer paths is significantly smaller than the cases of

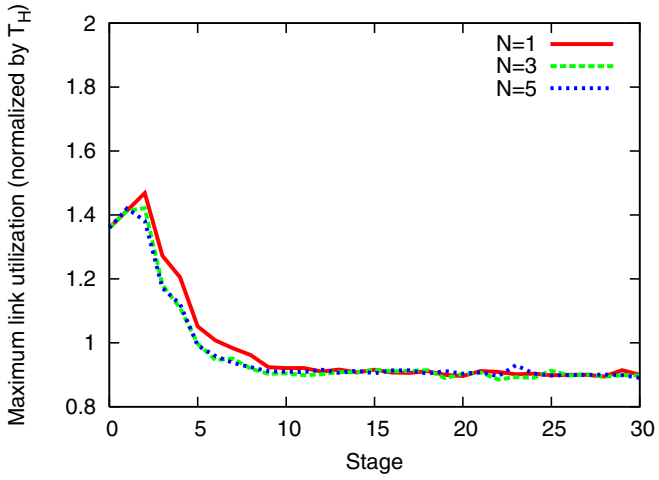


Figure 3.6: Maximum utilization (with various N)

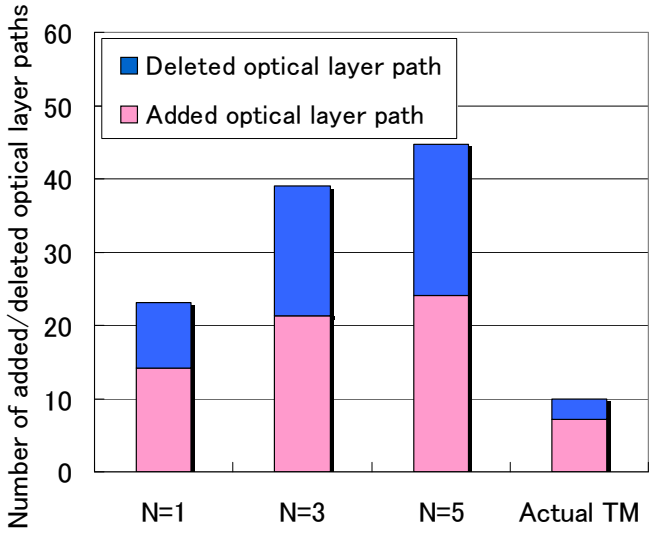


Figure 3.7: Number of added/deleted optical layer paths of additional equation method

$N = 3$ and $N = 5$ though the number is larger than the case using the actual traffic matrices. This is because many optical layer paths are added or deleted before the estimation errors are reduced when there is a large N. As a result, due to estimation errors, many unnecessary optical layer paths are added.

According to the above results in this paragraph, the gradual reconfiguration method can reduce the estimation errors dramatically even when we allow only one optical layer path to be added or deleted in each stage. In addition, by using the traffic matrices whose accuracies are improved at each stage, the gradual reconfiguration can achieve the adequate VNT as is the case with the reconfiguration using the actual traffic matrices. Especially, by limiting the number of added or deleted optical layer paths at each stage, we can achieve the adequate VNT by adding or deleting a small number of optical layer paths.

The case that traffic changes

The evaluation described above assumes that the traffic is constant after the beginning of the VNT reconfiguration. However, real traffic changes over time. Therefore, in this paragraph, we evaluate our methods in the case that traffic changes.

Adaptability to the fluctuations in traffic According to the dynamic stationary model proposed in Ref. [83] which models traffic fluctuations with a period of 1 to 1.5 hours, we generate changes in traffic as follows.

$$t_{i,j,n} = \alpha_{i,j} + \gamma_{i,j,n} \tag{3.19}$$

where $t_{i,j,n}$ is the amount of traffic from node i to node j at stage n, $\alpha_{i,j}$ is the average of traffic from node i to node j, and $\gamma_{i,j,n}$ is the factor by which traffic fluctuates. In this simulation, we generated the value of $\gamma_{i,j,n}$ based on Gaussian random values whose average is 0 and variance is $(\lambda\alpha_{i,j})^2$. Then, by changing the value of λ, we changed the size of the fluctuation in traffic.

Figure 3.8 shows the results for $\lambda = 0.01$, 0.05, and 0.10. In this simulation, we set N to 1. We set $\alpha_{i,j}$ equal to the element of the matrix used in the previous paragraph. Similar to the previous paragraph, we simulated our gradual reconfiguration ten times by using different traffic matrices and Fig. 3.8 shows the averages of results.

In this graph, the horizontal axis represents number of stages from the beginning of the VNT reconfiguration, and the vertical axis represents the RMSREs. From this figure, the smaller λ is, the faster we can reduce the estimation errors. However, even when $\lambda = 0.10$, the additional

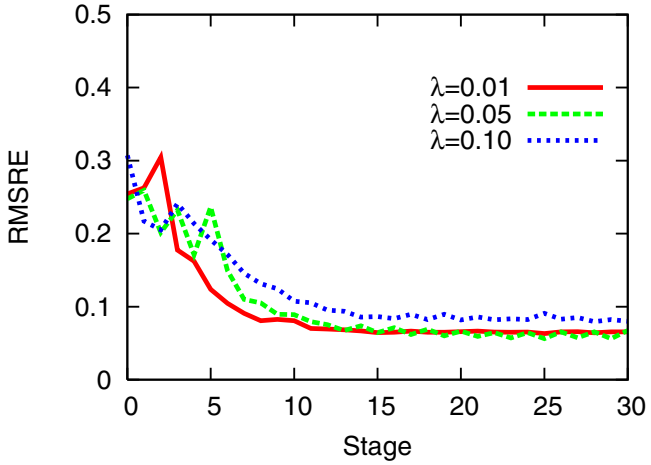


Figure 3.8: RMSREs of additional equation method (the case with changes of traffic)

equation method can reduce estimation errors significantly, because in this method, the number of equations used to estimate the traffic matrix increases as it progresses through the stages. The additional equations constrain the solution of the traffic matrix estimation. Because we generated the fluctuation component of traffic according to Gaussian random values whose average is 0, no traffic amount monitored in any stage is far from the traffic in the current stage. In addition, the traffic amounts monitored in stages with temporally high or low traffic are balanced out. As a result, since the constraints from the additional equations are appropriate to the current traffic, we can increase the accuracy of the traffic matrix estimation as we progress through the stages.

Figure 3.9 shows the maximum link utilizations in each stage. Since the accuracy of the estimated traffic matrix is dramatically improved as stages go on, the gradual reconfiguration can make the maximum link utilizations less than T_H after several stages from the beginning of the reconfiguration as is the case with the previous paragraph. That is, the gradual reconfiguration can efficiently work even in the case traffic fluctuates.

Adaptability to sudden change in traffic There is another type of change of traffic. According to the results described in Ref. [88], some end-to-end traffic may change suddenly. We evaluated our additional equation method when such sudden changes in traffic occur during the gradual reconfiguration. In this simulation, we generated sudden changes in traffic by doubling five randomly selected elements in the traffic matrix described in the previous paragraph. We set λ to 0.05 and N to 1 and added a sudden change in Stage 4.

Figure 3.10 compares estimation errors for the additional equation method with and without

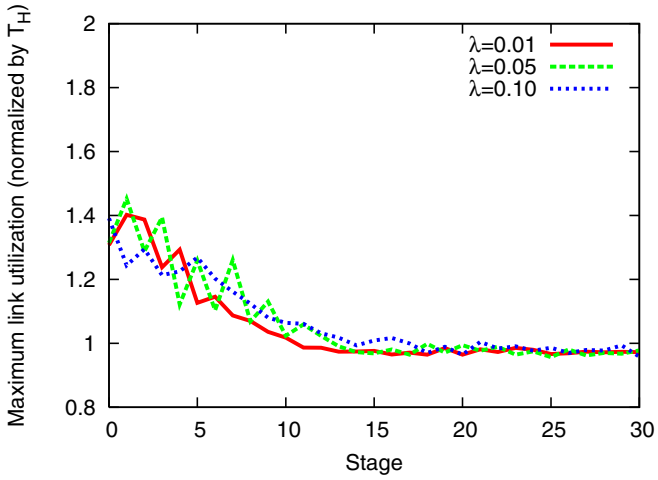


Figure 3.9: Maximum utilization (the case with changes of traffic)

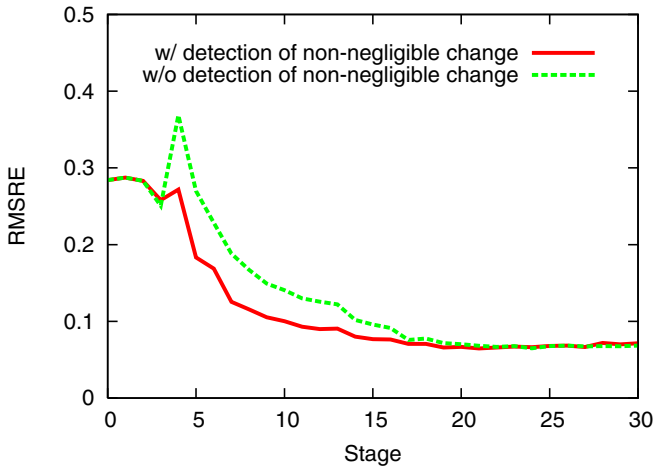


Figure 3.10: RMSREs of the additional equation method when some traffic change suddenly

detection of non-negligible change. In this figure, the horizontal axis represents stages since the beginning of the VNT reconfiguration and the vertical axis represents the RMSREs. From this figure, at Stage 4, the estimation errors of the additional equation method without detection of non-negligible change increase significantly. This is because the traffic amount before the sudden change, which is far from the current traffic, is used to estimate the traffic matrix. On the other hand, the additional equation method with detection of non-negligible change can reduce the estimation errors even after Stage 4 because it can remove the information about the traffic with non-negligible change. That is, even when there is a sudden change in traffic, by identifying and deleting the

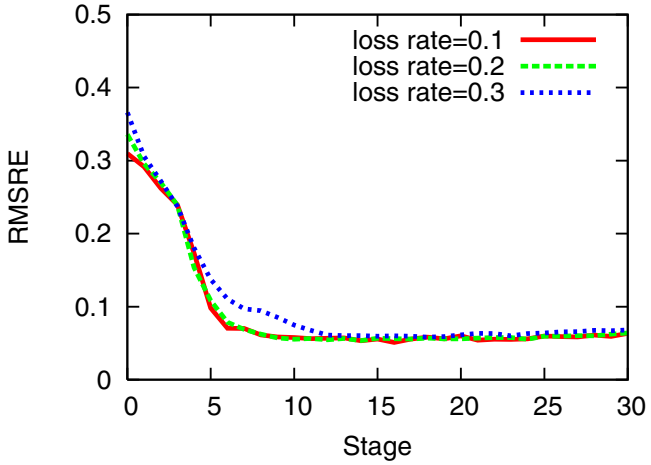


Figure 3.11: RMSREs of the additional equation method when some of link loads cannot be monitored

information about the traffic with such changes, the additional equation method can estimate a traffic matrix accurately. As a result, an accurate traffic matrix is available for VNT reconfiguration.

The case that monitoring link loads is less reliable

Finally, we investigate the robustness of the additional equation method to the loss or inaccuracy of the monitored link loads.

Robustness to the loss of the link load information Typical methods to collect link loads (e.g., SNMP) use UDP transport. Thus, some of the link loads may not be collected due to the packet loss. In this paragraph, we discuss how the additional equation method works in the case of the loss of the link load information.

Figure 3.11 shows the RMSREs for each stage when the loss rates of the link load information are 0.1, 0.2 and 0.3. In this simulation, we used the same traffic matrices in the case of $\lambda = 0.05$ used in Fig. 3.8. Similar to the above results, this figure shows the averages of the results using ten different traffic matrices.

From this figure, even when the loss rate is 0.3, we can reduce the RMSREs dramatically. This is because we can obtain the additional information from the collected link loads even if some of link loads cannot be monitored, though the loss of the information reduces the number of additional information.

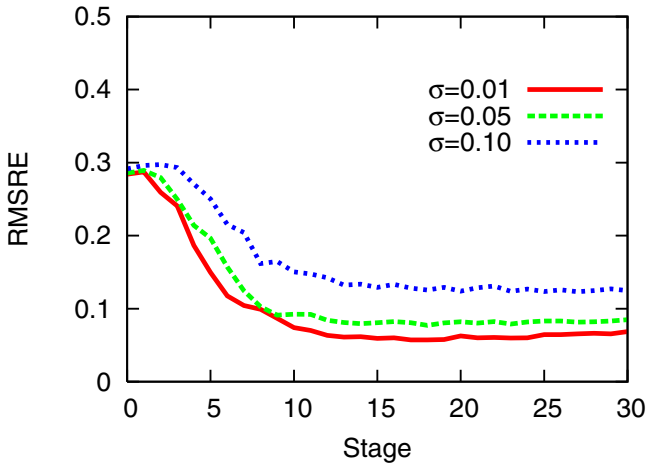


Figure 3.12: RMSREs of the additional equation method when link load information includes some errors

Robustness to the inaccurate link load information Monitored link loads may include monitoring errors. That is, monitored link load matrix $\hat{X}_i$ is described by

$$\hat{X}_i = X_i + e_i \tag{3.20}$$

where X_i denotes the actual link loads and e_i denotes monitoring errors. In this paragraph, we evaluate the gradual reconfiguration method with the additional equation method when the link load information is inaccurate. In this evaluation, we generated e_i based on Gaussian random values whose average is 0 and variance is $(\sigma X_i)^2$. We also used the same actual traffic matrices in the above case of $\lambda = 0.05$ used in Fig. 3.8.

Figure 3.12 shows RMSREs in the cases of $\sigma = 0.01$, $\sigma = 0.05$ and $\sigma = 0.10$. We conducted the simulations using ten different traffic matrices and this figure shows the averages of the results. From this figure, the RMSREs are dramatically reduced even in the case of $\sigma = 0.10$, though the reduction is smaller than the case of smaller σ. This is because the monitoring errors whose corresponding elements of e_i are positive or negative are balanced out, similar to the case of the fluctuations of traffic described above.

5.5 Conclusion

We have proposed a method that reduces estimation errors during VNT reconfiguration. Our method reconfigures the VNT gradually by dividing it into multiple stages. By dividing the VNT reconfiguration into multiple stages, our traffic matrix estimation method calibrates and reduces the estimation errors in each stage by using information monitored in prior stages. We have also investigated the effectiveness of our proposal using simulations. The results show that our method can improve the accuracy of the traffic matrix estimation and achieve an adequate VNT as is the case with the reconfiguration using the actual traffic matrices.

Section 6 Estimation of Current Traffic Matrices from Long-term Traffic Variations

The estimation method proposed in the previous section obtains additional measurements by changing routes via a TE method. By performing TE a sufficient number of times, this approach obtains a sufficient number of measurements and then estimates the traffic matrix by assuming that no or only few elements of the true traffic matrix change significantly throughout the TE method execution. However, when it takes a long time to change routes sufficient times, the current traffic can differ from the initial traffic monitored before the first route change. Therefore, we need a traffic matrix estimation method that considers the time variations of traffic matrices. Ref. [10] proposes a method for modeling traffic variations by using periodic functions and estimates these functions' parameters. When traffic changes unpredictably, however, a TE method cannot configure routes suitable for the current traffic by using the traffic variations estimated by this approach, since it can only estimate the average variations of traffic for a period of a day by monitoring link loads for several days.

Therefore, in this section, we propose a new estimation method, with which we can accurately estimate current traffic matrices even when traffic changes. Unlike in Ref. [10], the purpose of our method is to estimate not the long-term variations of traffic but the current traffic matrix, which consists of both long-term variations and short-term variations. By using the accurate traffic matrix, a TE method can properly work to configure routes suitable for the current traffic.

In our method, we first estimate the long-term variations of traffic by using the link loads monitored the last M times. Then, we adjust the estimated long-term variations so as to fit the current link loads. In addition, when the traffic variation trends change and the estimated long-term variations cannot match the current traffic, our method detects mismatches between the estimated long-term

variations and the current traffic. Then, our method re-estimates the long-term variations after removing information about the end-to-end traffic causing the mismatches, so as to capture the current traffic variations.

We also evaluate our estimation method through simulation. According to the results, our estimation method can estimate current traffic matrices accurately without the root mean squared relative error (RMSRE) larger than 0.1 even when traffic changes significantly.

6.1 Method for Estimating Current Traffic Matrix by Using Changes in Routes

In this section, we propose a new method for estimating current traffic matrices accurately. We assume that a TE method sometimes changes routes in the network. Under this condition, we can obtain additional information, which can be used in estimating the traffic matrices, by monitoring link loads while some routes are changed.

When it takes a long time to change routes enough times to obtain a sufficient amount of additional information, however, the current traffic might be very different from the initially monitored link loads. Therefore, we need to consider long-term variations. By using the link loads monitored the last M times, our method estimates the long-term variations of traffic instead of estimating the current traffic matrices directly. Then, we obtain the current traffic matrices by adjusting the estimated long-term variations so as to fit the current link loads.

In addition, when the traffic variation trends change, the changes may cause significant estimation errors if we also use information obtained before the changes, since this information can be very different from the current traffic. Therefore, in our method, we check whether the estimated long-term variations match the current link loads. Then, if the mismatch is detected, we re-estimate the long-term variations.

Fig. 3.13 shows an overview of the proposed estimation method. Our method estimates the traffic matrix through the following steps.

Step 1 Estimate the long-term variations of the traffic matrices by using the link loads monitored the last M times.

Step 2 Obtain estimation results of the current traffic matrix by adjusting the estimated long-term variations so as to fit the current link loads.

Step 3 Check whether the estimated long-term variations fit the current link loads. If they do not match the current link loads, return to Step 1 after removing the previous information about the end-to-end traffic causing the mismatch. Otherwise, proceed to Step 4.

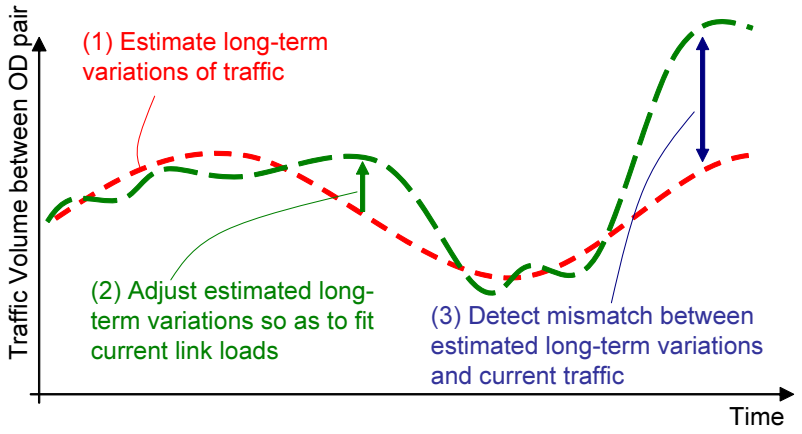


Figure 3.13: Overview of estimation method using long-term traffic variations

Step 4 Designate the estimation results from Step 2 as the final estimation results.

In this subsection, we first describe the method for estimating the long-term traffic variations. Then, we explain how to adjust the estimated long-term variations so as to fit the current link loads. Finally, we describe how to the detect mismatches between the estimated long-term variations and the current traffic, and how to re-estimate the long-term variations and the current traffic matrix after mismatch detection.

Estimating long-term traffic variations

Traffic variation model According to Ref. [10], the amount of traffic between each node pair varies periodically with a certain cycle, such as one day or one week. Therefore, in this section, we model the traffic amount between nodes i and j as

$$t_{i,j}(n) = f_{i,j}(n) + \delta_{i,j}(n), \tag{3.21}$$

where $t_{i,j}(n)$ is the traffic volume between nodes i and j at time n, $f_{i,j}(n)$ is a function modeling the periodic variation, and $\delta_{i,j}(n)$ is the variation not included in $f_{i,j}(n)$. In our method, we estimate the long-term variations by modeling $f_{i,j}(n)$ and estimating its parameters.

We model $f_{i,j}(n)$ by applying the model used in Ref. [10]. This approach models the periodic traffic variation by using sin and cos functions. With this model, the periodic variation is

represented as

$$f_{i,j}(n) = \sum_{h=0}^{N_f} \alpha_{h,i,j} cos\left(\frac{2\pi nh}{N^{\mathrm{cycle}}}\right) + \sum_{h=0}^{N_f} \alpha_{h+N_f,i,j} sin\left(\frac{2\pi nh}{N^{\mathrm{cycle}}}\right). \tag{3.22}$$

where N^{cycle} is the number of times monitoring link loads in each cycle, N_f is a parameter determining the number of terms in Eq. (3.22), and the $\alpha_{h,i,j}$ are the variables to be estimated by our estimation method. With N_f set to a large value, the traffic variation modeled by Eq. (3.22) captures more of the short-term variation, but the number of variables to be estimated also increases. In our method, we only have to roughly model the traffic variations, because we can estimate the current traffic matrix by adjusting the roughly estimated long-term variations. That is, in our method, a small N_f is sufficient.

Method for estimating long-term variations In the model described by Eq. (3.22), the variables $\alpha_{h,i,j}$ determine the long-term variations. Therefore, our method estimates the long-term variations by estimating the $\alpha_{h,i,j}$. We estimate the $\alpha_{h,i,j}$ by using the link loads monitored the last M times. At any time n, the link loads and the traffic matrix have a relation described by

$$X(n) = A(n)T(n) \tag{3.23}$$

where $X(n)$ is a matrix indicating the amount of traffic on each link at time n, $T(n)$ is the traffic matrix at time n, and $A(n)$ is the routing matrix (i.e., a matrix in which an element corresponding to an instance of end-to-end traffic and a link is 1 if the end-to-end traffic passes the link or 0 if it does not). Therefore, we estimate all variables so as to satisfy Eq. (3.23) in any time. In our method, we use a least square algorithm to estimate the variables. That is, when the number of nodes is N, the variables are basically estimated as

$$minimize \sum_{k=n-M+1}^{n} |X(k) - A(k)\hat{T}^{\mathrm{est}}(k)|^2 \tag{3.24}$$

where

$$\hat{T}^{\mathrm{est}}(k) = \begin{bmatrix} f_{0,0}(k) \\ \vdots \\ f_{i,j}(k) \\ \vdots \\ f_{N,N}(k) \end{bmatrix}. \tag{3.25}$$

By using Eq. (3.24), when some routes are changed, we can use additional equations for estimating the variables.

With Eq. (3.24), however, we may not be able to estimate the long-term variations accurately because of the effects of traffic variations that cannot be modeled by Eq. (3.22). Because the actual traffic variations do include variations that cannot be modeled by Eq. (3.22) (i.e., $\delta_{i,j}(n)$ in Eq. (3.21)), long-term variations modeled by Eq. (3.22) cannot completely fit all the monitored link loads. With Eq. (3.24), however, we estimate the long-term variations so as to completely fit all the monitored link loads. As a result, estimation results from Eq. (3.24) can be affected by traffic variations that cannot be modeled by Eq. (3.22), making the results very different from the actual traffic.

To mitigate the impact of $\delta_{i,j}$ on the estimated long-term variations, in our method, by placing constraints on the variables themselves, we avoid estimating the long-term variations so as to completely fit all the monitored link loads. We thus use the following equation instead of Eq. (3.24):

$$minimize \sum_{k=n-M+1}^{n} |X(k) - A(k)\hat{T}^{\mathrm{est}}(k)|^2 + \Phi \sum_{i,j} \left(m_{i,j} \sum_{h=0}^{2N_f} (\alpha_{h,i,j} - \alpha'_{h,i,j})^2 \right), \tag{3.26}$$

where the $\alpha'_{h,i,j}$ are the variables estimated the previous time, $m_{i,j}$ is the amount of information monitored before, and Φ denotes a parameter by which we can set the weight to the constraints on the variables themselves. Using this equation, we estimate all the $\alpha_{h,i,j} (0 \leq h \leq 2N_f)$ of $f_{i,j}(n)$ so as to fit all the monitored link loads while keeping the values close to the values estimated the previous time.

When we estimate the long-term variations the first time, however, we have not obtained the $\alpha'_{h,i,j}$. Thus, in such cases, we set the $\alpha'_{0,i,j}$ to the elements of traffic matrices estimated by other methods [1–3, 6, 89–91], and we set the $\alpha'_{h,i,j} (1 \leq h \leq 2N_f)$ to 0. By using this approach, we can avoid estimating traffic variations as having significantly larger values than the actual variations.

In addition, even if the initial $\alpha'_{h,i,j}$ are not accurate, we can estimate the long-term variations

more accurately by using link loads monitored at multiple times as additional information. Then, when we estimate the long-term variations the next time, we can use more accurate $\alpha'_{h,i,j}$. That is, as we estimate the long-term variations more times, the accuracies of these estimations increase.

Adjustment of estimated long-term variations

By the method proposed in the previous paragraph, we estimate the long-term variations. Because these estimates do not include the $\delta_{i,j}(n)$ in Eq. (3.21), however, they do not fit the current link loads. Therefore, we adjust the long-term variations estimated as given in the previous paragraph so as to fit the current link loads.

The adjustment is performed through the following steps. First, by assigning n to the functions corresponding to the estimated long-term variations, we obtain a roughly estimated traffic matrix $\hat{T}^{\rm est}(n)$. Then, we obtain a traffic matrix $\hat{T}(n)$ that is close to $\hat{T}^{\rm est}(n)$ and fits the link loads monitored at time n. That is, we obtain the estimation results by applying a least square algorithm so as to satisfy the following conditions:

$$minimize|\hat{T}(n) - \hat{T}^{\rm est}(n)|^2 \tag{3.27}$$

where

$$A(n)\hat{T}(n) = X(n). \tag{3.28}$$

A traffic matrix estimated by a least square algorithm, however, can include negative values, which are meaningless in the context of a traffic matrix. Therefore, we eliminate negative values through the following steps. We denote the estimated traffic matrix for the i-th iteration as $\hat{T}^{(i)}(n)$.

Step 1 Let $\hat{T}^{(0)}(n) \leftarrow \hat{T}(n)$.

Step 2 Obtain the matrix $\hat{T}'^{(i)}(n)$, in which we replace all the negative values of $\hat{T}^{(i)}(n)$ with zero.

Step 3 Obtain $D^{(i)}(n)$ satisfying the following condition:

$$minimize|D^{(i)}(n)|^2 \tag{3.29}$$

where

$$A(n)\left(\hat{T}'^{(i)}(n) + D^{(i)}(n)\right) = X(n). \tag{3.30}$$

Step 4 Let $\hat{T}^{(i+1)}(n) \leftarrow \hat{T}'^{(i)}(n) + D^{(i)}(n)$.

Step 5 If all elements of $\hat{T}^{(i+1)}(n)$ are non-negative, proceed to Step 6. Otherwise, return to Step 1.

Step 6 Let $\hat{T}^{(i+1)}(n)$ be the final result for the traffic matrix $\hat{T}(n)$.

Re-estimation of traffic matrix after mismatch of estimated long-term variations

When traffic variation trends change, long-term variations estimated by using all the link loads monitored the last M times can exhibit mismatches with the current traffic. This is because the long-term variations are estimated so as to fit the link loads before the change, which can be very different from the current traffic variations. In such cases of mismatch, we cannot estimate the current traffic matrices accurately even after adjustment, because the adjustment uses only the current link loads, which are insufficient for estimating the traffic matrices accurately.

Therefore, in our method, when the estimated long-term variations exhibit mismatches with the current traffic, we detect the mismatches and re-estimate the long-term variations without using link loads that do not match the current traffic. In this paragraph, we describe how to detect mismatches and identify the end-to-end traffic causing the mismatches, as well as how to re-estimate the long-term variations after mismatch detection.

Detecting mismatches and identifying end-to-end traffic causing mismatches When the estimated long-term variations are very different from the current traffic, the differences between the current link loads and the link loads calculated using the estimated long-term variations are large. In this case, because the results of adjusting $\hat{T}(n)$ must satisfy Eq. (3.28), while $A(n)\hat{T}^{\mathrm{est}}(n)$ is very different from the current link loads $X(n)$, the elements of $\hat{T}^{\mathrm{est}}(n) - \hat{T}(n)$, corresponding to the traffic causing the mismatches, become large. Therefore, we detect mismatches and identify the end-to-end traffic causing the mismatches by evaluating $\hat{T}^{\mathrm{est}}(n) - \hat{T}(n)$.

Because the size of traffic variation that cannot be included in Eq. (3.22) depends on the end-to-end traffic [10], if we set a single threshold for the elements of $\hat{T}^{\mathrm{est}}(n) - \hat{T}(n)$, traffic with large variations that cannot be modeled by Eq. (3.22) will be erroneously detected as traffic causing mismatches.

Therefore, we detect mismatches and identify their sources by comparing $\hat{T}^{\mathrm{est}}(n) - \hat{T}(n)$ with its previous values. Our method performs the comparison by using the Smirnov-Grubbs method [92], which can easily detect outliers in sampled data.

Here, we define the elements of $\hat{T}^{\mathrm{est}}(n)$ and $\hat{T}(n)$ corresponding to the traffic between nodes i and j as $\hat{t}^{\mathrm{est}}_{i,j}(n)$ and $\hat{t}_{i,j}(n)$ respectively. In the Smirnov-Grubbs method, we detect whether

$|\hat{t}^{\mathrm{est}}_{i,j}(n) - \hat{t}_{i,j}(n)|$ is an outlier by calculating

$$d_{i,j} = \frac{|\hat{t}^{\mathrm{est}}_{i,j}(n) - \hat{t}_{i,j}(n)| - \mu_{i,j}}{\sigma_{i,j}}, \tag{3.31}$$

where $\mu_{i,j}$ and $\sigma_{i,j}$ are the average and standard deviation of $|\hat{t}^{\mathrm{est}}_{i,j}(k) - \hat{t}_{i,j}(k)| (n-M+1 \leq k \leq n)$, respectively. Then, $|\hat{t}^{\mathrm{est}}_{i,j}(n) - \hat{t}_{i,j}(n)|$ is detected as an outlier if $d_{i,j}$ is larger than the threshold

$$\tau = (M-1)\sqrt{\frac{\tau^2_{\theta,M+2}}{M(M-2) + M\tau^2_{\theta,M+2}}} \tag{3.32}$$

where M is the number of samples, θ is a parameter specifying the detection sensitivity, and $\tau_{\theta,M}$ is a value corresponding to the top $\theta/M\%$ points of the T distribution with $M-2$ degrees of freedom.

Too small $\sigma_{i,j}$ causes detection of points where $|\hat{t}^{\mathrm{est}}_{i,j}(n) - \hat{t}_{i,j}(n)|$ is small. We do not, however, need to detect such points, because the estimated long-term variations there fit the current traffic, since $|\hat{t}^{\mathrm{est}}_{i,j}(n) - \hat{t}_{i,j}(n)|$ is small. Therefore, to avoid detecting such points, we introduce a parameter s and set $\sigma_{i,j}$ to s if $\sigma_{i,j}$ is smaller than s.

Re-estimation of long-term variations after detection When mismatches between the estimated long-term variations and the current traffic are detected, we need to re-estimate the long-term variations so as to fit the current traffic. Because such mismatches occur when we estimate the long-term variations by using previously monitored link loads that are very different from the current traffic variations, we re-estimate the long-term variations by using link loads and routing matrices in which information about the end-to-end traffic causing the mismatches has been removed.

Our method removes previous information corresponding to the end-to-end traffics causing mismatches at time n through the following steps. We first remove such information from the routing matrices $A(i)(n-M+1 \leq i < n)$ by setting elements corresponding to the identified end-to-end traffic to 0. We denote the routing matrix after such replacement as $A'(i)$.

Then, we create a link load matrix $X'(i)(n-M+1 \leq i < n)$ from which information about the identified end-to-end traffic has been removed. The sum of the elements of traffic matrix T corresponding to the identified end-to-end traffic traversing each link at time i is calculated as $(A(i) - A'(i))\,T$. Therefore, X'_i is given by

$$X'(i) = X(i) - (A(i) - A'(i))\,\hat{T}'^{\mathrm{est}}(i). \tag{3.33}$$

where $\hat{T}'^{\mathrm{est}}(i)$ is the traffic matrix at time i calculated using the estimated long-term variations. In calculating $\hat{T}'^{\mathrm{est}}(i)$, we use the long-term variations estimated at time $n-1$, since the long-term variations estimated at time n can be affected by changing trends.

Next, our method re-estimates the long-term variations by using Eq. (3.34), which is refined from Eq. (3.26) to use $X'(k)$ and $A'(k)$:

$$\begin{aligned} minimize \quad & \sum_{k=n-M+1}^{n-1} |X'(k) - A'(k)\hat{T}^{\mathrm{est}}(k)|^2 \\ & + |X(n) - A(n)\hat{T}^{\mathrm{est}}(n)|^2 \\ & + \Phi \sum_{i,j} \left(m_{i,j} \sum_{h=0}^{2N_f} (\alpha_{h,i,j} - \alpha'_{h,i,j})^2 \right). \end{aligned} \tag{3.34}$$

We only have to re-estimate the long-term variations so as to fit the current traffic, because the purpose of our method is to estimate the current traffic matrix. Moreover, in estimating the traffic amounts of the identified end-to-end traffic by using Eq. (3.34), we do not need to consider the related traffic variations, because the traffic amounts corresponding to the identified traffic are included only in $X(n)$.

Therefore, in re-estimating the long-term variations, we model the amounts of the identified end-to-end traffic by

$$f_{i,j}(n) = \alpha_{0,i,j}, \tag{3.35}$$

instead of using Eq. (3.22). By using Eq. (3.35), we can minimize the number of variables to be estimated.

Re-estimation of traffic matrix after re-estimation of long-term variations After re-estimating the long-term variations, we re-estimate the current traffic matrix so as to satisfy Eq.(3.27) through the same steps described above.

6.2 Evaluation of Estimation Method

Metrics

In this subsection, we describe an evaluation of our method by simulation. In the simulation, we evaluated our method by two general metrics: (1) the accuracy of estimation, and (2) the performance of a TE method using the estimated traffic matrices.

To evaluate the accuracy, we used two specific metrics – the root mean squared error (RMSE), and the root mean squared relative error (RMSRE) – as defined below:

$$\mathrm{RMSE} = \sqrt{\frac{1}{N^2} \sum_{1 \le i,j \le N} (\hat{t}_{i,j}(n) - t_{i,j}(n))^2} \tag{3.36}$$

$$\mathrm{RMSRE} = \sqrt{\frac{1}{N_{\tilde{t}}^2} \sum_{1 \le i,j \le N, t_{i,j} > \tilde{t}} \left(\frac{\hat{t}_{i,j}(n) - t_{i,j}(n)}{t_{i,j}(n)} \right)^2} \tag{3.37}$$

The RMSE gives an overall measure for the errors in estimation, while the RMSRE gives a relative measure. For small matrix elements, however, the relative errors are not really important. Thus, in computing the RMSRE, we consider only matrix elements greater than a threshold $\tilde{t}$. $N_{\tilde{t}}$ is the number of elements greater than $\tilde{t}$ in a traffic matrix. In the following simulation, $\tilde{t}$ was set so that the sum of the end-to-end traffic whose actual rate was greater than $\tilde{t}$ composed 75 % of the total traffic.

To evaluate the performance of a TE method using the estimated traffic matrices, we investigated whether the purpose of the TE method was achieved. The next paragraph describes the purpose of the TE method used in our simulation.

Environment used in evaluation

In our method, we assume that a TE method changes routes sometimes. In this evaluation, we used the optical layer TE as an example of a TE method. The optical layer TE establishes optical layer paths between two IP routers over a physical network consisting of IP routers and optical cross-connects (OXCs). A set of optical layer paths forms a virtual network topology (VNT). Traffic between two routers is carried over the VNT by using IP layer routing. Under these conditions, the optical layer TE accommodates traffic that fluctuates widely by dynamically reconfiguring the VNT.

In our simulation, we used the European Optical Network (EON) (19 nodes, 37 links) shown in Fig. 3.3 as the physical topology and executed the same optical layer TE method used in the previous section once an hour. The purpose of this method is to keep the maximum link utilization under the threshold T_H. In this method, optical layer paths are added or deleted with a limitation on the number of optical layer paths reconfigured at one time. Optical layer paths are added if at least one path whose utilization exceeds the threshold T_H exists. Otherwise, if there is an optical layer

path whose utilization is less than a threshold T_L, the path is deleted. In this simulation, we set the maximum number of optical layer paths reconfigured at one time to 30, T_H to 0.7, and T_L to 0.4.

In the simulation, end-to-end traffic was generated by adding variations to sin functions whose amplitudes and phases were randomly generated.

Case without sudden traffic changes

In this paragraph, we investigate the accuracy of our method for estimating the long-term variations and the effectiveness of adjusting the estimated long-term variations when there are no sudden traffic changes. In the simulation, we set $M = 160$ and $N_f = 2$. Link loads were monitored once per 20 minutes.

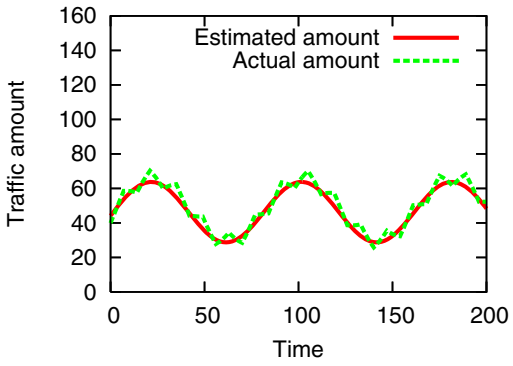
Accuracy of estimation of long-term variations In our method, we estimate long-term traffic variations by using Eq. (3.26) instead of Eq. (3.24) to avoid the impact of traffic variations that cannot be modeled by Eq.(3.22). Therefore, to verify the effectiveness of using Eq. (3.26), we compared the estimation results obtained with Eqs. (3.24) and (3.26).

Note that estimation results obtained with Eq. (3.26) depend on previously estimated variables (i.e., the long-term variations are accurately estimated when the estimation results of the previous time are accurate). Here, however, because the purpose of this comparison was to verify the effectiveness of adding constraints on the variables themselves even in the case of inaccurate $\alpha'_{h,i,j}$, we set $\alpha'_{0,i,j} = \mu$ and $\alpha'_{h,i,j} = 0(1 \leq i \leq 2N_f)$, where μ is the total volume of incoming traffic divided by the amount of end-to-end traffic.

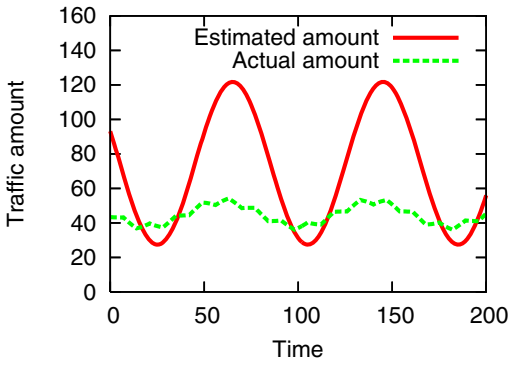
Figure 3.14 shows the comparison results. As seen from this figure, both Eq. (3.24) and Eq. (3.26) can be applied to accurately estimate the traffic between nodes 2 and 14. The traffic between nodes 12 and 16, however, cannot be estimated accurately with Eq. (3.24), and the estimated traffic variation is significantly larger than the actual traffic variation.

This is because Eq. (3.24) estimates the variables so as to completely fit all the link loads monitored the last M times, even though the actual traffic variations include those that cannot be modeled by Eq. (3.22) (i.e., $\delta_{i,j}(n)$ in Eq. (3.21)). As a result, the long-term variations estimated by Eq. (3.24) are affected by $\delta_{i,j}(n)$, and the long-term variations of some end-to-end traffic become very different from the actual variations.

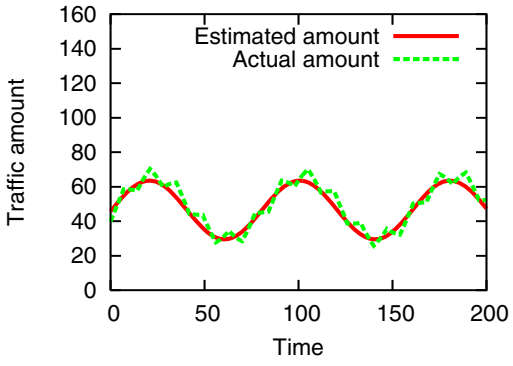
On the other hand, by using Eq. (3.26), we can avoid estimating variations as significantly larger than the actual variations. That is, by adding constraints on the variable themselves, we can mitigate the impact of $\delta_{i,j}(n)$ and increase the accuracy of estimating long-term variations.



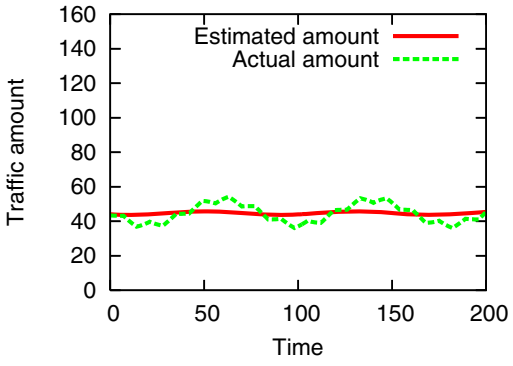
(a) Traffic between nodes 2 and 14 (case of using Eq.(3.24))



(b) Traffic between nodes 12 and 16 (case of using Eq.(3.24))



(c) Traffic between nodes 2 and 14 (case of using Eq.(3.26) with Φ=0.01)



(d) Traffic between nodes 12 and 16 (case of using Eq.(3.26) with Φ=0.01)

Figure 3.14: Results of estimating long-term variations

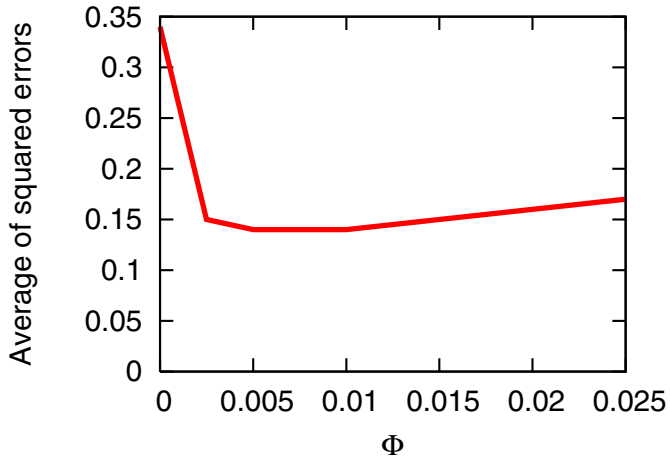


Figure 3.15: RMSRE vs. Φ

Next, to evaluate the effectiveness of the constraints on the variables themselves in detail, we estimated the long-term variations with various values of Φ. Figure 3.15 shows the relation between Φ and the maximum RMSRE for the estimated long-term variations. According to this figure, when Φ is set to a value close to 0, the RMSRE becomes larger. This is because with small values of Φ, the estimation results become more sensitive to $\delta_{i,j}(n)$. As a result, $\delta_{i,j}(n)$ causes estimation errors like that shown in Fig. 3.14(b). On the other hand, with large values of Φ, the variables in the long-term variations are estimated so as to be close to the $\alpha'_{h,i,j}$. As a result, when Φ is too large, the long-term variations cannot be estimated so as to fit the monitored link loads. The optimal Φ depends on the environment (e.g., the amplitudes of traffic variations), and determining optimal value of Φ is a goal for our future works. From Fig. 3.15, however, we can see that the estimation errors are not significant even if Φ is not optimal.

Effectiveness of adjustment In our method, we obtain estimation results by adjusting the estimated long-term variations so as to fit the current link loads. Therefore, we investigated the effectiveness of adjusting the estimated long-term variations, by comparing the accuracy of the estimated traffic matrices after adjustment with the accuracies of the following methods:

- A method using only the current link loads. By comparison with this method, we investigated the effectiveness of using the link loads monitored at previous times. For this method, we used the tomogravity method with the simple gravity model [6]. Although the simple

gravity model does not fit the traffic matrices used in our simulation, because we use randomly generated traffic matrices, this model also is not incorrect in some real networks [68]. The focus of this comparison is the effectiveness of using link loads monitored at previous times when the simple gravity model is not correct.

- A method using the link loads monitored at previous times but not considering the time variations of traffic. By comparison with this method, we investigated the effectiveness of modeling long-term traffic variations. For this method, we used the additional equation method proposed in the previous section.

- A method using the link loads monitored at previous times but only estimating the long-term variations. By comparison with this method, we investigated the effectiveness of adjusting the estimated long-term variations so as to fit the current link loads. In this simulation, for estimating the long-term variations, we set Φ to 0.01.

Figures 3.16 and 3.17 show the RMSE and RMSRE, respectively, at each time. The results show that the errors for the tomogravity method are the largest. This is because the tomogravity method uses only the current link loads, which is an insufficient amount of information.

The errors for the additional equation method are also large. This is because that method does not consider traffic variations but assumes instead that the true traffic matrix does not change during TE execution. Therefore, this method cannot estimate traffic matrices accurately when traffic varies, even while monitoring the link loads a sufficient number of times.

On the other hand, the errors for the method estimating the long-term variations are relatively small. That is, by including the link loads monitored at previous times in considering the time variations of traffic, we can estimate traffic matrices accurately. In addition, by adjusting the estimated long-term variations, we can estimate traffic matrices even more accurately. This is because the adjustment enables the estimation results to also follow traffic variations that cannot be modeled by Eq. (3.22).

Case with sudden traffic changes

In the previous paragraph, we evaluated our method in the case without sudden traffic changes. In real networks, however, traffic can change suddenly, and traffic variation trends can also change. When the trends change and the estimated long-term variations do not match the current traffic, our method detects mismatches and identifies the end-to-end traffic causing them, after which it re-estimates the long-term variations. In this paragraph, we investigate the accuracy of this detection

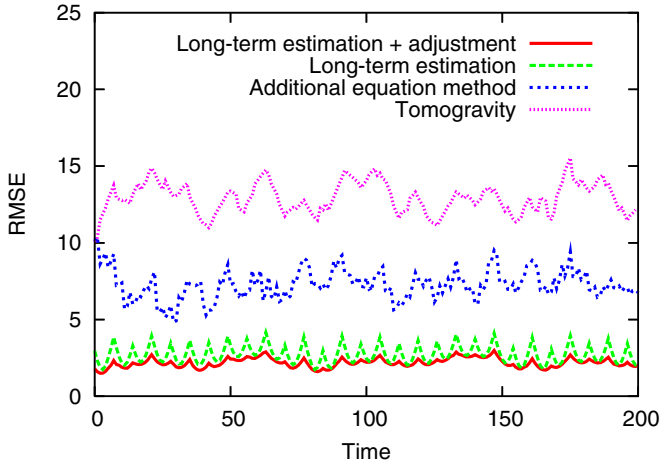


Figure 3.16: Time variation of RMSE (the case without the sudden chages of traffic)

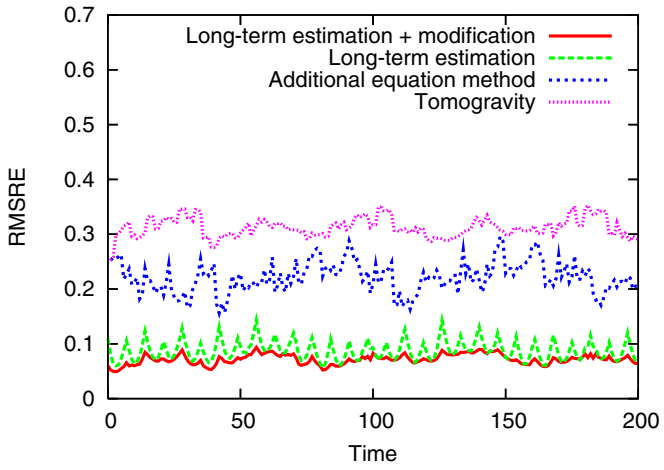


Figure 3.17: Time variation of RMSRE (the case without the sudden chages of traffic)

and the effectiveness of re-estimation after detection. In addition, we investigate how well a TE method works when it uses traffic matrices estimated by our method. In all the simulations described below, we set Φ to 0.01, θ to 0.01, and s to 1.

Accuracy of detection of trend changes To investigate the accuracy of mismatch detection, we calculated $d_{2,14}$ for the traffic generated by adding sudden changes to the traffic between nodes 2 and 14 in the scenario from the previous paragraph. Figure 3.18 shows the results. In Fig. 3.18, the horizontal axis represents the rate of added traffic divided by the maximum rate of traffic before the addition, and the vertical axis represents $d_{2,14}$. In addition, we show the maximum relative error

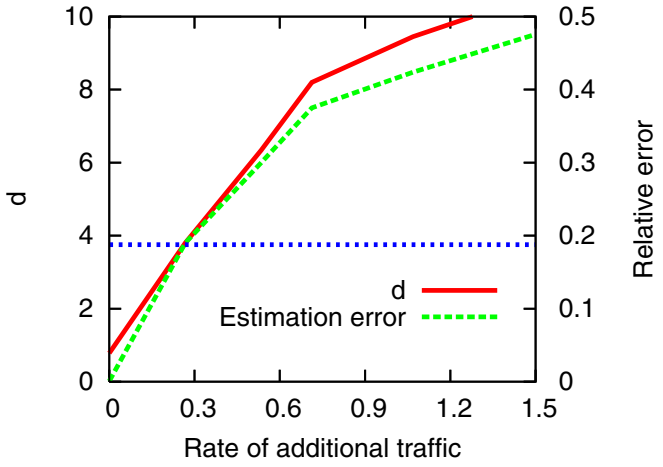


Figure 3.18: $d_{2,14}$ vs. rate of added traffic

corresponding to the traffic between nodes 2 and 14 for the case without re-estimation.

From this figure, we can see that the larger the added traffic is, the larger $d_{2,14}$ becomes. This is because the difference between the estimated long-term variations and the current link loads becomes large when the rate of added traffic is large. As a result, the differences between the traffic matrices before and after adjustment also become large. In this simulation, because we set M to 160 and θ to 0.01, the threshold τ calculated from Eq. (3.32) is 3.75. Therefore, if the added traffic is more than 0.3 times the rate of the traffic before addition (i.e., when the added traffic causes a relative error of more than 0.20 in the case without re-estimation), we can detect mismatches between the estimated traffic variations and the current traffic. In this situation, our method re-estimates the long-term variations so as to fit the current traffic.

In detecting mismatches, false positives (i.e., the cases of mistakenly detecting end-to-end traffic with no changes) can occur. Thus, we investigated the likelihood of such false positives. Figure 3.19 shows the complementary cumulative distribution of $d_{i,j}$ in the case without adding sudden changes. The vertical line in Fig. 3.19 represents 3.75, the threshold τ calculated above. This figure shows that the probability of mistaken detection in the case with no changes is about 0.03 %. We discuss the impacts of these false positives later.

Effectiveness of re-estimation When mismatches between the estimated long-term variations and the current traffic are detected, we re-estimate the long-term variations after removing information about the end-to-end traffic identified as causing the mismatches. We investigate the effectiveness of the re-estimation through simulation. In this simulation, we used traffic generated by

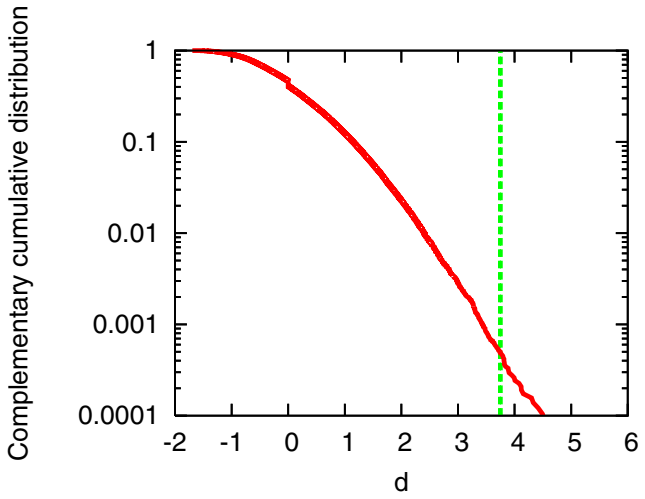


Figure 3.19: Complementary cumulative distribution of $d_{i,j}$ with no changes

adding sudden changes to the traffic used for the simulation described in the previous paragraph. We added sudden changes to the traffic from nodes 2 to 4, 9 to 1, and 0 to 12 at times 70, 110, and 140, respectively. The rates of the sudden traffic changes from nodes 2 to 4, 9 to 1, and 0 to 12 were, respectively, 120 % , 150 %, and 160 % of the maximum rate of traffic before the addition.

Figure 3.20 shows the RMSE when we added these sudden traffic changes, for four different methods: our method with re-estimation, our method without re-estimation, the additional equation method, and the tomogravity method. For our method without re-estimation, we estimated the long-term variations and adjusted them but did not re-estimate them even when the variation trends changed.

From this figure, similarly to the results shown in Figs. 3.16 and 3.17, we can see that the RMSEs for the tomogravity method and the additional equation method are large. The figure also shows that the RMSE for our method without re-estimation is small before time 70 but increases afterward, whereas the RMSE for our method with re-estimation remains small after time 70. This difference is caused by the sudden changes, whose impact we discuss in detail later.

The results shown in Fig. 3.20 also verify that the impact of false positives is small. As described above, about 0.03 % of the end-to-end traffic without changes in the traffic variation trends will be mistakenly identified as causing mismatches between the estimated long-term variations and the current traffic. For example, at time 76 in Fig. 3.20, the end-to-end traffic between nodes 14 and 0 is mistakenly identified as causing a mismatch. From the figure, however, we can see that the RMSE for our method does not become significant even when such false positives occur; it always

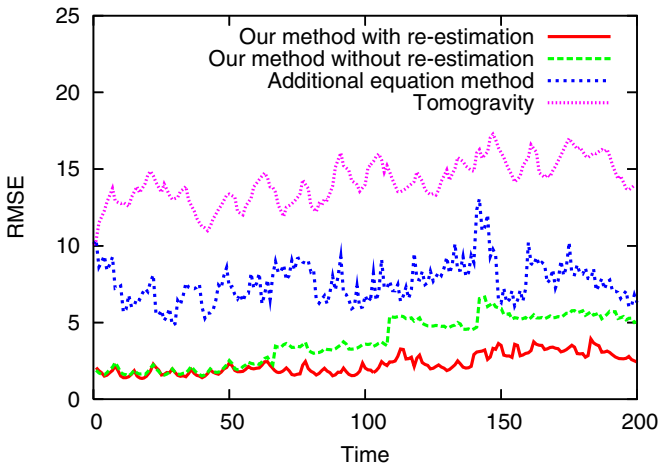


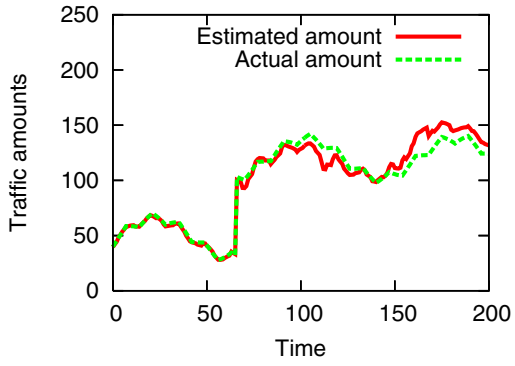
Figure 3.20: Time variation of RMSE (when some traffic variations change)

remains the smallest among the four methods. This is because we have sufficient information to estimate the long-term variations and traffic matrices accurately even when some false positives occur and information about the mistakenly identified end-to-end traffic is removed.

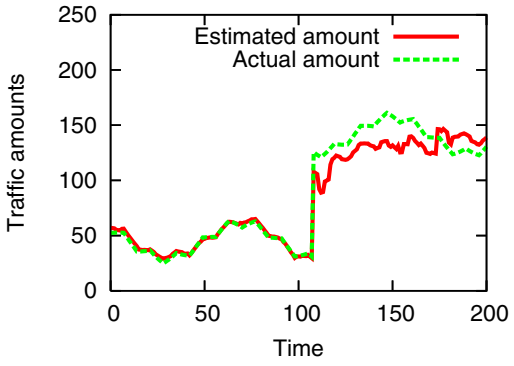
To investigate the impact of sudden changes in detail, we compared the estimation results obtained for traffic with sudden changes added. Figures 3.21 and 3.22 show the estimation results for our method with and without re-estimation, respectively.

These figures show that both methods can accurately estimate all the traffic amounts before adding the sudden changes. After adding the changes, however, the traffic rate estimated by our method without re-estimation cannot capture the changes. This is because that method also uses the link loads monitored before adding the sudden changes, which are very different from the current traffic variations. Therefore, because of this information that does not fit the current traffic variations, the long-term variations cannot be estimated accurately. Even though we adjust the estimated long-term variations so as to fit the current link loads, the adjusted results still do not capture the sudden changes, because the adjustment process can use only the current link loads, which is insufficient information for estimating the traffic matrices accurately.

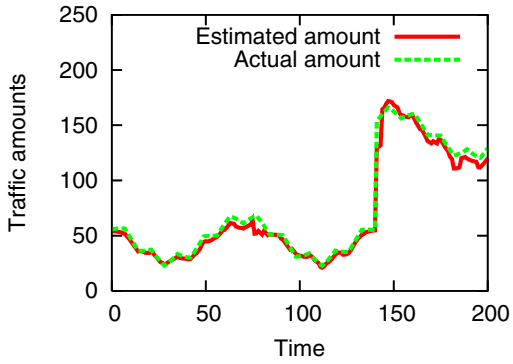
On the other hand, our method with re-estimation can estimate all the traffic amounts accurately even after adding the sudden changes. This is because by re-estimating the long-term variations after removing information about the end-to-end traffic causing the mismatches between the estimated long-term variations and the current traffic, we avoid the impact of information that is very different from the current traffic variations.



(a) Traffic between nodes 2 and 4



(b) Traffic between nodes 9 and 1

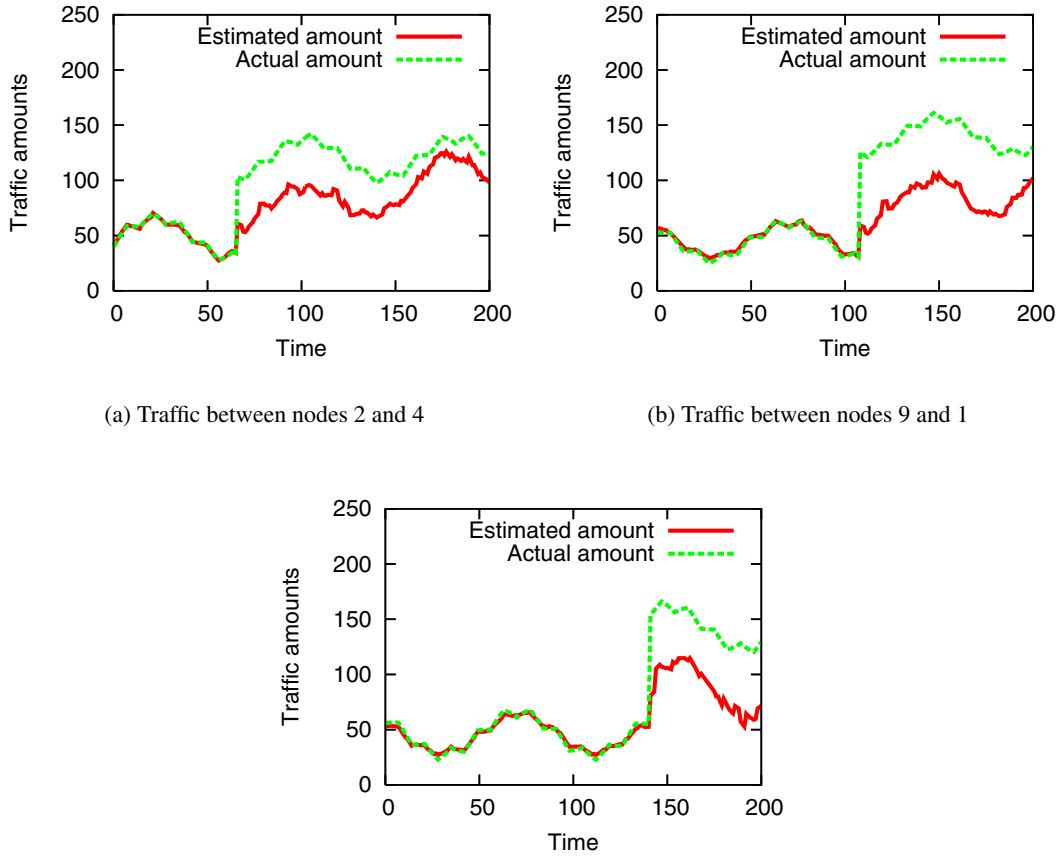


(c) Traffic between nodes 0 and 12

Figure 3.21: Estimation results for our method with re-estimation

Impact on performance of TE methods Finally, we evaluate the performance of TE methods using traffic matrices estimated by our method. The TE method used in our simulations configured the VNT and routes over the VNT so as to keep the maximum link utilization under the threshold T_H. When we use traffic matrices including estimation errors, however, these errors can cause the maximum link utilization to be above T_H. Therefore, in this evaluation, we investigated the maximum link utilization after TE was performed. For this simulation, we used the same traffic with sudden traffic changes described above.

Figure 3.23 shows the results of this simulation. The figure shows that when using the tomogravity method or the additional equation method, the maximum link utilization becomes significantly



(a) Traffic between nodes 2 and 4

(b) Traffic between nodes 9 and 1

(c) Traffic between nodes 0 and 12

Figure 3.22: Estimation results for our method without re-estimation

larger than the threshold T_H. This is because the estimation errors of these methods are large, as described above. When the estimation errors are large, the link utilizations after executing the TE method, as calculated using the estimated traffic matrix, can be very different from the actual link utilizations. As a result, the link utilizations after TE are mistakenly regarded as being lower than T_H, even though the actual link utilizations are still high and the necessary optical layer paths have not been added.

This figure also shows that the maximum link utilizations in the case of using our method without re-estimation sometimes become significantly larger than the threshold, as well. This is caused by significant underestimation of the traffic including the sudden changes. As shown in Fig. 3.22,

our method without re-estimation cannot capture the added sudden changes and significantly underestimates their amounts. Because of these underestimates, when the TE method changes the routes of the underestimated traffic, it does not reserve enough bandwidth. As a result, since the actual traffic rates are much higher than expected, the link utilizations become high.

On the other hand, in the case of using our method with re-estimation, we can reduce the maximum link utilization to around T_H at all times. This is because, with re-estimation, our method can estimate traffic matrices accurately even when the traffic changes suddenly.

Although the maximum link utilization is reduced to around T_H with traffic matrices estimated by our method with re-estimation, however, it is not always smaller than T_H. This is because estimation errors can still be included in the results of our method with re-estimation, even though this method is the most accurate of the four methods considered here.

Especially when multiple instances of end-to-end traffic are identified as causing mismatches between the estimated long-term variations and the current traffic, the estimation errors increase, because removing the previous information about these multiple instances decreases the amount of information used for estimation. In the case of Fig. 3.23, two instances of end-to-end traffic are erroneously identified as causing the mismatch at time 183. These false positives cause a slight increase in the estimation error, leading to a link utilization higher than T_H. Thus, to estimate traffic matrices more accurately, we need to minimize the number of false positives by setting parameters optimally or using a more sophisticated detection method. These considerations remain for our future work.

Minimizing the number of false positives is insufficient, however, because it is possible for multiple instances of end-to-end traffic to actually change suddenly, causing mismatches between the estimated long-term variations and the current traffic. When this happens, increases in estimation errors are difficult to avoid, because we cannot obtain sufficient information about such traffic changing suddenly. Therefore, to avoid the impact of such errors on methods using estimated traffic matrices, TE methods also need to consider estimation errors. This is another topic for our future work.

6.3 Conclusion

In this section, we have proposed a method for estimating current traffic matrices by using the changes in routing matrices introduced via a TE method. In this method, we first estimate the long-term variations of traffic matrices by using the link loads monitored the last M times. Then, we obtain the current traffic matrix by adjusting the estimated long-term variations of traffic so as to

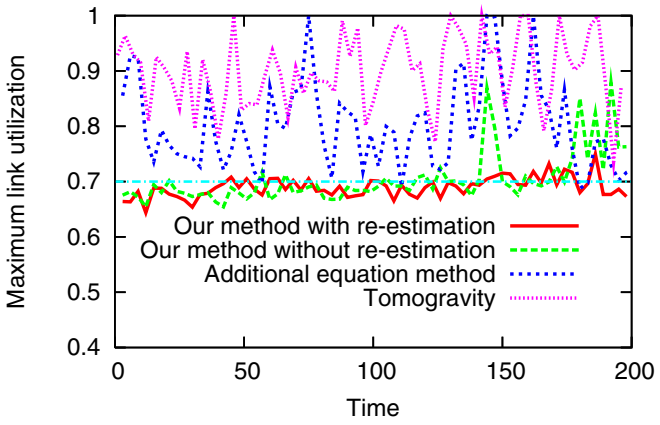


Figure 3.23: Variation in maximum link utilization after TE execution

fit the current link loads. In addition, when the traffic variation trends change and the estimated long-term variations cannot fit the current variations, our method detects mismatches and identifies the end-to-end traffic causing them. Then, our method re-estimates the long-term traffic variations after removing information about the end-to-end traffic causing the mismatches.

In this section, we evaluated our method through simulation. According to the results, our method can obtain accurate traffic matrices by adjusting the estimated long-term variations. In addition, when some end-to-end traffic changes suddenly and the estimated long-term variations do not match the current traffic, our method can detect mismatches accurately. Then, by re-estimating the long-term variations after removing information about the end-to-end traffic causing the mismatches, the method can estimate current traffic matrices accurately even when some end-to-end traffic changes suddenly.

Chapter 4

Conclusion

Network operators design their networks according to the predicted traffic so as to accommodate all traffic efficiently (e.g., without congestion or large delays). However, if the current traffic significantly differ from the predicted one, the previously constructed network becomes no longer suitable to the current traffic; for example, it may happen that utilizations of some links are extremely high and cause congestion or large delays. Similarly, if a server receives unexpectedly many requests, the server cannot respond to the requests.

Thus, when the current traffic becomes significantly different from the predicted one, we need to handle the changes of traffic so as not to degrade the performances of network. Such significant changes of traffic are caused either by the malicious traffic or by the increases of legitimate traffic. In this thesis, we have proposed methods to handle both cases.

To handle the changes caused by malicious traffic, we have proposed the methods to detect attacks, identify attack nodes and protect legitimate traffic as described below.

First, in Section 2 we have proposed a method to detect attacks. In our detection method, we focus on SYN flood attacks which are the most frequent attacks among DDoS attacks. One of the problems in detecting SYN flood traffic is that server nodes or firewalls cannot distinguish the SYN packets of normal TCP connections from those of SYN flood attack. Moreover, since the rate of normal network traffic may vary, we cannot use an explicit threshold of SYN arrival rates to detect SYN flood traffic. Thus, we have investigated the difference of statistics of arrival rates of normal TCP SYN packets and SYN flood attack packets by using the traffic data monitored at the gateway of our university. According to the results, the arrival rate of normal TCP SYN packets can be modeled by a normal distribution while the arrival rate of TCP SYN packets when attack starts is far from a normal distribution. Based on the analytical results, our detection method detects attacks by checking the difference between the sampled SYN rates and the normal distribution. The simulation results show that our method can detect attacks whose rates are even lower than 20 SYNs/sec. In addition, the results also show that our method can detect attacks faster than the existing detection method.

Then, in Section 3 we have proposed a method to identify attack nodes which can work with legacy routers unlike the traditional traceback methods. In our identification method, we identify the egress routers that attack nodes are connecting to by estimating the traffic matrix between arbitral source-destination edge pairs from the traffic volumes of each link which can be monitored by legacy routers. By monitoring the traffic variations obtained by the traffic matrix, we identify the edge routers that are forwarding the attack traffic, which have a sharp traffic increase to the victim. According to the simulation results, even when we can monitor only the link loads, our method can

identify attack sources accurately and limit the total attack rate from unidentified attack sources by setting parameters adequately.

Finally, in Section 4 we have proposed a method to defend legitimate traffic. Our defense method also focuses on SYN flood attacks. In our defense 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 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. According to our simulation results, our method can make the probability of dropping legitimate SYN packets less than 0.1 even when the attack rate exceeds 600,000 SYNs/sec.

On the other hand, if the significant changes of traffic are caused by the increases of legitimate traffic, we should not block any traffic unlike the case of the malicious traffic. Thus, in this case, we reconfigure the network settings so as to accommodate all current traffic efficiently. To reconfigure the network settings, a traffic matrix, which indicates traffic volumes between all pairs of edge nodes, is required as an input. However, the traffic matrices are hard to monitor directly. Though several methods to estimate traffic matrices have been proposed, the estimated traffic matrices may include estimation errors which degrade the performance of the reconfigured network significantly. Therefore, we have proposed methods that reduce estimation errors during the reconfiguration.

First, in Section 5, we have proposed a gradual reconfiguration method in which the reconfiguration of network settings is divided into multiple stages instead of reconfiguring the suitable settings at once. By dividing the reconfiguration into multiple stages and assuming that no or only few elements of the true traffic matrix change significantly throughout the TE method execution, we can calibrate and reduce the estimation errors in each stage by using information monitored in prior stages. We have evaluated the effectiveness of the gradual reconfiguration by simulation. According to the results, the gradual reconfiguration can reduce the root mean squared relative error (RMSRE) to 0.1 and achieve adequate network settings as is the case with the reconfiguration using the actual traffic matrices.

However, when it takes a long time to achieve adequate network settings, the current traffic can differ from the initial traffic monitored before the first route change. This violates the fundamental

assumption of the above method. Therefore, in Section 6, we have also proposed a new estimation method, with which we can accurately estimate current traffic matrices even when traffic changes. In this method, we first estimate the long-term variations of traffic by using the link loads monitored the last M times. Then, we adjust the estimated long-term variations so as to fit the current link loads. In addition, when the traffic variation trends change and the estimated long-term variations cannot match the current traffic, our method detects mismatches between the estimated long-term variations and the current traffic. Then, our method re-estimates the long-term variations after removing information about the end-to-end traffic causing the mismatches, so as to capture the current traffic variations. According to our simulation results, our estimation method can estimate current traffic matrices accurately without RMSRE larger than 0.1 even when traffic changes significantly.

According to the results discussed in each section, even when network traffic changes significantly, we can identify the causes of the changes by measuring and analyzing the network state information and we can avoid the performance degradation of networks by controlling networks based on the results of the analysis.

There are several challenging tasks as future works. One of them is to defend servers and networks from other kinds of attacks than SYN flood 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. Even against these attacks, our methods can identify the attack sources and can also verify that the packets are not spoofed by delegating the connection requests. However, to avoid such attacks, we also need a method which efficiently filters attack traffic after the identification and verification.

Another future work is constructing a TE method that considers estimation errors. Though we can reduce the estimation errors by our methods proposed in Chapter 3, it is very hard to estimate traffic matrices without any estimation errors. Therefore, TE methods also need to consider estimation errors.

Bibliography

[1] C. Tebaldi and M. West, "Bayesian inference on network traffic using link count data," *Journal of the American Statistical Association*, vol. 93, pp. 557–576, June 1998.

[2] J. Cao, D. Davis, S. V. Wiel, and B. Yu, "Time-varying network tomography," *Journal of the American Statistical Association*, vol. 95, pp. 1063–1075, Feb. 2000.

[3] I. Juva, S. Vaton, and J. Virtamo, "Quick traffic matrix estimation based on link count covariances," in *Proceedings of IEEE ICC 2006*, vol. 2, pp. 603–608, June 2006.

[4] L. Tan and X. Wang, "A novel method to estimate IP traffic matrix," *IEEE Communications Letters*, vol. 11, pp. 907–909, Nov. 2007.

[5] L. Tan and X. Wang, "On IP traffic matrix estimation," in *Proceedings of IEEE International Conference on Computer Communications and Networks (ICCCN 2007)*, pp. 617–624, Aug. 2007.

[6] Y. Zhang, M. Roughan, N. Duffield, and A. Greenberg, "Fast accurate computation of large-scale IP traffic matrices from link loads," in *Proceedings of ACM SIGMETRICS 2003*, pp. 206–217, June 2003.

[7] J. Fang, Y. Vardi, and C.-H. Zhang, "An iterative tomogravity algorithm for the estimation of network traffic," *Complex Datasets and Inverse Problems: Tomography, Networks and Beyond*, vol. 54, pp. 12–23, Aug. 2007.

[8] G. Liang, N. Taft, and B. Yu, "A fast lightweight approach to origin-destination IP traffic estimation using partial measurements," *IEEE/ACM Transactions on Networking*, vol. 14, pp. 2634–2648, June 2006.

[9] D. Jiang, J. Chen, and L. He, "An accurate approach of large-scale IP traffic matrix estimation," *IEICE Transactions on Communications*, vol. E90-B, pp. 3673–3676, Dec. 2007.

[10] A. Soule, A. Nucci, R. Cruz, E. Leonardi, and N. Taft, "Estimating dynamic traffic matrices by using viable routing changes," *IEEE/ACM Transactions on Networking*, vol. 13, pp. 485–498, June 2007.

[11] M. Roughan, M. Thorup, and Y. Zhang, "Traffic engineering with estimated traffic matrices," in *Proceedings of Internet Measurement Conference 2003*, pp. 248–258, Oct. 2003.

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

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

[14] "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.

[15] D. Moore, C. Shannon, D. J. Brown, G. M. Voelker, and S. Savage, "Inferring Internet Denial-of-Service activity," *ACM Transactions on Computer Systems*, vol. 23, pp. 115–139, May 2006.

[16] "Symantec internet security threat report." available at `http://www.symantec.com/enterprise/threatreport/index.jsp`, Mar. 2005.

[17] 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.

[18] 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*, vol. 4, pp. 2043–2049, Nov. 2004.

[19] 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. *(in Japanese)*.

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

[21] 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.

[22] 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.

[23] T. Peng, C. Leckie, and K. Ramamohanarao, "Detecting distributed denial of service attacks using source IP address monitoring." available at `http://www.cs.mu.oz.au/~tpeng/mudguard/research/detection.pdf`, Nov. 2002.

[24] Y. Ohsita, S. Ata, and M. Murata, "Identification of attack nodes from traffic matrix estimation," *IEICE Transactions on Communications*, vol. E90-B, pp. 2854–2864, Oct. 2007.

[25] Y. Ohsita, S. Ata, and M. Murata, "Identification of attack nodes from traffic matrix estimation," in *Proceedings of 4th International Trusted Internet Workshop*, Dec. 2005.

[26] Y. Ohsita, S. Ata, and M. Murata, "Traffic matrix estimation for identification of attack sources," *IEICE Society Conference*, Sept. 2005. *(in Japanese)*.

[27] Y. Ohsita, S. Ata, and M. Murata, "Identification of attack nodes from traffic matrix estimation," *Technical Reports of IEICE(NS2005-86, IN2005-86, CS2005-32)*, Sept. 2005. *(in Japanese)*.

[28] B. Wang and H. Schulzrinne, "A denial-of-service-resistant IP traceback approach," in *Proceedings of IEEE Symposium on Computers and Communications*, vol. 1, pp. 351–356, June 2004.

[29] S. Savage, D. Wetherall, A. Karlin, and T. Anderson, "Practical network support for IP traceback," in *Proceedings of ACM SIGCOMM 2000*, pp. 295–306, Aug. 2000.

[30] D. X. Song and A. Perrig, "Advanced and authenticated marking schemes for IP traceback," in *Proceedings of IEEE INFOCOM 2001*, vol. 2, pp. 878–886, Apr. 2001.

[31] K. Law, J. C. Lui, and D. K. Yau, "You can run, but you can't hide: An effective methodology to traceback DDoS attackers," in *Proceedings of International Symposium on Modeling,*

Analysis, and Simulation of Computer and Telecommunications Systems, pp. 433–440, Oct. 2002.

[32] A. C. Snoeren, C. Partridge, L. A. Sanchez, C. E. Jones, F. Tchakountio, B. Schwartz, S. T. Kent, and W. T. Strayer, "Single-packet IP traceback," *IEEE/ACM Transactions on Networking*, vol. 10, pp. 721–734, Dec. 2002.

[33] T.-H. Lee, W.-K. Wu, and T.-Y. W. Huang, "Scalable packet digesting schemes for IP traceback," in *Proceedings of IEEE International Conference on Communications 2004*, pp. 1008–1013, June 2004.

[34] A. Soule, K. Salamatian, and N. Taft, "Combining filtering and statistical methods for anomaly detection," in *Proceedings of Internet Measurement Conference 2005*, pp. 331–344, Oct. 2005.

[35] A. Lakhina, M. Crovella, and C. D. February, "Diagnosing network-wide traffic anomalies," in *Proceedings of ACM SIGCOMM 2004*, pp. 219–230, Aug. 2004.

[36] A. Lakhina, M. Crovella, and C. Diot, "Detecting distributed attacks using network-wide flow traffic," in *Proceedings of FloCon 2005 Analysis Workshop*, Sept. 2005.

[37] B. Krishnamurthy, S. Sen, Y. Zhang, and Y. Chen, "Sketch-based change detection: Methods, evaluation, and applications," in *Proceedings of Internet Measurement Conference 2003*, pp. 234–247, Oct. 2003.

[38] "Cisco NetFlow." available at `http://www.cisco.com/en/US/products/ps6601/products_ios_protocol_group_%home.html`.

[39] Cisco, "NetFlow performance analysis." available at `http://www.cisco.com/en/US/tech/tk812/technologies_white_paper0900aecd8%02a0eb9.shtml`.

[40] L. Huang, X. L. Nguyen, M. Garofalakis, M. Jordan, A. Joseph, and N. Taft, "Distributed PCA and network anomaly detection." Technical Report UCB/EECS-2006-99, Electrical Engineering and Computer Science Department, University of California Berkeley, July 2006.

[41] Y. Ohsita, S. Ata, and M. Murata, "Deployable overlay network for defense against distributed SYN flood attacks," to appear in *IEICE Transactions on Communications*, vol. E91-B, Aug. 2008.

[42] Y. Ohsita, S. Ata, and M. Murata, "Deployable overlay network for defense against distributed SYN flood attacks," in *Proceedings of IEEE International Conference on Computer Communications and Networks (ICCCN 2005)*, pp. 407–412, Oct. 2005.

[43] Y. Ohsita, S. Ata, and M. Murata, "Deployable overlay network for defense against distributed SYN flood attacks," *Technical Reports of IEICE(IN2004-125)*, pp. 13–18, Dec. 2004. *(in Japanese)*.

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

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

[46] 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, July 2001.

[47] 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.

[48] 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.

[49] 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*, vol. 24, pp. 1864–1876, Oct. 2006.

[50] 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.

[51] H. Fuji, E. Y. Chen, K. Okada, and D. Kashiwa, "Movingfirewall: A countermeasure against distributed denial of service attacks," *NTT Technical Review*, vol. 1, pp. 85–88, Aug. 2003.

[52] 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.

[53] H. R. Nagesh and K. C. Sekaran, "Design and deployment of proactive models for mitigating denial-of-service and distributed denial-of-service attacks," *International Journal of Computer Science and Network Security*, vol. 7, pp. 167–175, July 2007.

[54] A. D. Keromytis, V. Misra, and D. Rubenstein, "WebSOS: An overlay-based system for protecting web servers from denial of service attacks," *The International Journal of Computer and Telecommunications Networking*, vol. 48, pp. 781–807, Aug. 2005.

[55] D. G. Andersen, "Mayday: Distributed Filtering for Internet Services," in *Proceedings of 4th USENIX Symposium on Internet Technologies and Systems*, Mar. 2003.

[56] J. Kurian and K. Sarac, "FONet: A federated overlay networks for DoS defense in the Internet," in *Proceedings of Global Internet Symposium*, Apr. 2006.

[57] Y. Ohsita, T. Miyamura, S. Arakawa, S. Ata, E. Oki, K. Shiomoto, and M. Murata, "Gradually reconfiguring virtual network topologies based on estimated traffic matrices," submitted to *IEEE/ACM Transactions on Networking*.

[58] Y. Ohsita, T. Miyamura, S. Arakawa, E. Oki, K. Shiomoto, and M. Murata, "Increasing the accuracy of traffic matrix estimation for gradual reconfiguration of virtual network topologies," *Technical Reports of IEICE(PN2006-90)*, pp. 37–40, Oct. 2007. *(in Japanese)*.

[59] B. Mukherjee, D. Banerjee, S. Ramamurthy, and A. Mukherjee, "Some principles for designing a wide-area WDM optical network," *IEEE/ACM Transactions on Networking*, vol. 4, pp. 684–696, Oct. 1996.

[60] R. Hayashi, T. Miyamura, M. Aoki, and S. Urushidani, "Simulation of a dynamic multi-layer optimization algorithm with SRLG consideration," in *Proceedings of OECC/COIN 2004*, July 2004.

[61] D. Banerjee and B. Mukherjee, "Wavelength-routed optical networks: Linear formulation, resource budgeting tradeoffs, and a reconfiguration study," *IEEE/ACM Transactions on Networking*, vol. 8, pp. 598–607, Oct. 2000.

[62] J. Wei, C.-D. Liu, S.-Y. Park, K. Liu, R. Ramamurthy, H. Kim, and M. Maeda, "Network control and management for the next generation Internet," *IEICE Transactions on Communications*, vol. 83-B, pp. 2191–2209, Oct. 2000.

[63] L. Zhang, K. Lee, and C.-H. Youn, "Adaptive virtual topology reconfiguration policy employing multi-stage traffic prediction in optical Internet," in *Proceedings of Workshop on High Performance Switching and Routing*, pp. 26–29, May 2002.

[64] K. Shiomoto, E. Oki, W. Imajuku, S. Okamoto, and N. Yamanaka, "Distributed virtual network topology control mechanism in GMPLS-Based multiregion networks," *IEEE Journal on Selected Areas in Communications*, vol. 21, pp. 1254–1262, Oct. 2003.

[65] A. Gencata and B. Mukherjee, "Virtual-topology adaptation for WDM mesh networks under dynamic traffic," *IEEE/ACM Transactions on Networking*, vol. 11, pp. 236–247, Oct. 2003.

[66] S. Gieselman, N. Singhal, and B. Mukherjee, "Minimum-cost virtual-topology adaptation for optical WDM mesh networks," in *Proceedings of IEEE ICC*, vol. 3, pp. 1787–1791, June 2005.

[67] A. Soule, A. Lakhina, N. Taft, K. Papagiannaki, K. Salamatian, A. Nucci, M. Crovella, and C. Diot, "Traffic matrices: Balancing measurements, inference and modeling," in *Proceedings of ACM SIGMETRICS 2005*, pp. 362–373, June 2005.

[68] A. Gunnar, M. Johansson, and T. Telkamp, "Traffic matrix estimation on a large IP backbone –a comparison on real data," in *Proceedings of Internet Measurement Conference 2004*, pp. 149–160, Oct. 2004.

[69] Y. Ohsita, T. Miyamura, S. Arakawa, E. Oki, K. Shiomoto, and M. Murata, "Estimation of current traffic matrices from long-term traffic variations," submitted to *IEICE Transactions on Communications*.

[70] Y. Ohsita, T. Miyamura, S. Arakawa, E. Oki, K. Shiomoto, and M. Murata, "Estimating current traffic matrices accurately by using long-term variations information," to be presented at *Broadnets 2008*, Sept. 2008.

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

[72] V. Brazauskas and R. Serfling, "Robust and efficient estimation of the tail index of a one-parameter pareto distribution," *North American Actuarial Journal*, vol. 4, pp. 12–27, Apr. 2000.

[73] D. Watson and C. Labovitz, "Experiences with monitoring OSPF on a regional service provider," in *Proceedings of IEEE International Conference on Distributed Computing Systems (ICDCS)*, pp. 204–213, 2003.

[74] A. Fedmann, A. Greenberg, C. Lund, N. Reingold, and J. Rexford, "NetScope: Traffic engineering for IP networks," *IEEE Network Magazine*, vol. 14, pp. 11–19, Mar. 2000.

[75] "Scilab development team." available at `http://www-rocq.inria.fr/scilab/`.

[76] 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.

[77] 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.

[78] W. R. Stevens and G. R. Wright, *TCP/IP Illustrated*, vol. 2. Addison-Wesley Longman Publishing Co., Inc., 1995.

[79] D. G. Andersen, H. Balakrishnan, M. F. Kaashoek, and R. Morris, "Resilient overlay networks," in *Proceedings of 18th ACM Symposium on Operating Systems Principles*, pp. 131–145, Oct. 2001.

[80] 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.

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

[82] A. Farrel, J. P. Vasseur, and J. Ash, "A path computation element (PCE)-based architecture." RFC 4655, Aug. 2006.

[83] A. Nucci, A. Sridharan, and N. Taft, "The problem of synthetically generating IP traffic matrices: Initial recommendations," *ACM SIGCOMM Computer Communication Review*, vol. 35, pp. 19–32, July 2005.

[84] J. C. Nash, *Compact Numerical Methods for Computers: Linear Algebra and Function Minimisation*. Adam Hilger, 1990.

[85] B. S. Davie and Y. Rekhter, *MPLS: Technology and Applications*. Morgan Kaufmann Publishers, 2000.

[86] "Lapack – linear algebra package." available at `http://www.netlib.org/lapack/`.

[87] "Internet2 network real time atlas." available at `http://atlas.grnoc.iu.edu/I2.html`.

[88] R. Teixeira, N. Duffield, J. Rexford, and M. Roughan, "Traffic matrix reloaded: impact of routing changes," in *Proceedings of PAM 2005*, pp. 251–264, Mar. 2005.

[89] A. Medina, K. Salamatian, N. Taft, I. Matta, and C. Diot, "A two-step statistical approach for inferring network." available at `www.cs.bu.edu/techreports/pdf/2004-011-two-step-tm-inference.pdf`, Mar. 2004.

[90] Y. Zhang, M. Roughan, C. Lund, and D. Donoho, "An information-theoretic approach to traffic matrix estimation," in *Prodceedigs of ACM SIGCOMM 2003*, pp. 301–312, Aug. 2003.

[91] Y. Vardi, "Network tomography: Estimating source-destination traffic intensities from link data," *Journal of the Americal Statistical Association,*, vol. 91, pp. 365–377, Mar. 1996.

[92] S. Burke, "Missing values, outliers, robust statistics & non-parametric methods," *LC-GC Europe Online Supplement, Statistics & Data Analysis*, vol. 2, pp. 19–24, Jan. 2001.