HYBRID TRACEBACK-FILTERING (HTF): AN EFFICIENT DOS/DDOS DEFENSE MECHANISM

WU YUHUI

(B.Sc. Huazhong University of Science and Technology)

A THESIS SUBMITTED

FOR THE DEGREE OF MASTER OF SCIENCE

DEPARTMENT OF COMPUTER SCIENCE

NATIONAL UNIVERSTIY OF SINGAPORE

2004

ACKNOWLEDGMENTS

I wish to extend my sincerest gratitude to my supervisors, Dr Henry C.J. Lee, Dr Sandeep Kumar and Dr Bao Feng, for their encouragement, ideas, and support in bringing this work to completion. I am grateful to all of my friends in the Institute for Infocomm Research (I^2R) and the School of Computing, for their help and ideas throughout the course of my master's study. I would also like to acknowledge the support of the National University of Singapore and I2R through a graduate research scholarship that has made my graduate studies possible. I would like to thank my family for their love and support during my master's study.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

SUMMARY

Denial of Service (DoS) and Distributed Denial of service (DDoS) attacks are characterized by an explicit attempt to prevent legitimate users from access to a service. They pose a grave danger to Internet operation and cause significant financial damage every year, thus making it essential to devise effective techniques to defend against them. This thesis addresses this problem by proposing a new filtering traceback hybrid mechanism, which is shown to be an effective way to defend against DoS/DDoS attacks.

Over the past few years, many countermeasures have been proposed for DoS/DDoS attacks. These approaches are broadly classified into four categories, namely prevention, detection, filtering and traceback. Many studies have shown that prevention methods can effectively prevent the attack from happening, detection and filtering can react to DoS/DDoS attacks in real time and traceback is feasible even after the DoS/DDoS attacks stop. However, few proposals address the issues of minimizing discarding legitimate traffic and link bandwidth consumption when the DoS/DDoS attack is raging on.

In this thesis, I propose a Hybrid Traceback-Filtering-DoS (HTF-DoS) mechanism and its extended version, the Hybrid Traceback-Filtering-DDoS (HTF-DDoS) mechanism to use traceback to enhance packet filtering accuracy that minimizes dropping legitimate traffic. While the HTF-DoS mechanism is effective in simple DoS attack cases, the HTF-DDoS mechanism is feasible in distributed denial of service attack cases.

In the two mechanisms, the packet-dropping function at routers is recursively activated hop-by-hop using traceback. In so doing, the bandwidth consumed by attack traffic in

upstream links is decreased as compared to other proposals where only routers on a hop away from the victim perform filtering. In HTF-DoS and HTF-DDoS, we use controlled flooding to reconstruct the attack path, and then filter packets which come from attack links only at the routers which are on the attack path. This minimizes the false positive ratio caused by imprecise filtering. To minimize the reaction delay against DoS attacks, packet-filtering and route-reconstruction are conducted simultaneously.

Simulation results show that HTF-DoS and HTF-DDoS are able to drop 30% of bad packets at the start of the process and up to 100% when the traceback is completed in simple DoS cases and DDoS cases respectively. Our simulation also shows that false positive ratios range from 15% to 0% at the beginning and end of the traceback-filtering process respectively. As a result, both the HTF-DoS and HTF-DDoS mechanisms can improve the throughput of legitimate packets. Theoretical analysis of some types of graph topologies shows that our mechanism can save link bandwidth consumption. For instance, when there is only one attacker and the generation rate of bad packets is 500 times that of the good packets, the link bandwidth consumption can be saved by more than 56 percent. And when there are thirty attackers and the generation rate of bad packets per attacker is 100 times that of the good packets, the link bandwidth consumption can be saved by more than 400 percent.

CHAPTER 1
INTRODUCTION

The widespread usage of the Internet and networking, while increasing productivity, efficiency and knowledge sharing, has resulted in additional problems in computer security. The increase in the vulnerability of systems connected to the Internet is not only because there are more systems available for attack, but also because there are more systems available from which the attack can be carried out. Also, advancement in technology has provided sophisticated attack tools, which can be used even by novices. DoS/DDoS [1] attacks take advantage of all the three weaknesses mentioned above.

A DoS/DDoS attack is characterized by an explicit attempt to prevent legitimate users of a service from using these services. Examples of DoS/DDoS attacks include attempts to flood a network thereby preventing legitimate traffic from passing through the flooded links, attempts to disrupt connections between two machines thereby preventing access to a service, and attempts to prevent a particular individual from accessing a service.

Distributed denial of service attacks against high-profile web sites such as Yahoo, CNN, Amazon and E* Trade in early 2000 [2] demonstrate how damaging DDoS attacks are, and how defenseless the Internet is under such attacks. The services of these web sites were unavailable for hours or even days as a result of the attacks. Since then, much work has been carried out to develop efficient countermeasures to defend against DoS/DDoS attacks. However, there are few proposals that address the issue of minimizing false positive ratio and link bandwidth consumption. This thesis addresses this issue.

1.1 Accomplishments and Contributions

Our major contribution, which is elaborated in Chapter 3 and Chapter 4, is the development of an effective hybrid traceback-filtering mechanism called HTF-DoS and its extended version, named HTF-DDoS. The two mechanisms enhance packet filtering in conjunction with traceback in DoS and Distributed DoS respectively, and are able to minimize the false positive ratio and bandwidth consumption.

1.2 Structure of the Thesis

Chapter 2 presents a survey of DoS/DDoS attacks and its countermeasures.

In Chapter 3, we present our approach to DoS defense mechanism which addresses the false positive ratio and link bandwidth consumption in DoS attacks. This protocol, named HTF-DoS uses traceback to propagate an increasingly refined filter for DoS. We show through simulations that our mechanism is able to drop a large fraction of bad packets while keeping the false positive to a low value. As a result, our mechanism can improve the throughput of legitimate packets greatly. A theoretical analysis shows that our mechanism can also save link bandwidth consumption.

In Chapter 4, we present HTF-DDoS, an extension of the HTF-DoS mechanism to DDoS. We show by simulation and theoretical analysis that HTF-DDoS is also effective in dropping bad packets, keeping a low false-positive ratio, improving throughput of legitimate packets, and saving link bandwidth consumption in DDoS.

Finally, in Chapter 5, we present a summary of the various accomplishments and contributions of this study. We also outline several possible future research directions.

CHAPTER 2
DOS/DDOS ATTACKS AND COUNTERMEASURES

A Denial of Service (DoS) attack is a kind of attack which explicitly attempts to consume the resources of a host or network, thereby preventing legitimate users from accessing these services. The attackers typically send a huge amount of seemingly legitimate traffic to the victim to request for services, and in so doing, result in the significant consumption of CPU cycles or memory of the target host, or the bandwidth of the target network. Such attacks essentially disable the target host or target network from providing service to legitimate users.

Based on the number of attacking hosts deployed by the attacker to implement the attack, the attack can be classified into two categories---DoS attack and Distributed DoS attack (DDoS). A DDoS attack is a simple variation of a DoS attack. In DoS attacks, a single attacker consumes all the available bandwidth by generating a large number of packets operating from a single host while in the distributed cases multiple attackers coordinate together to produce the same effect from several hosts on the network. Because of the availability of automated tools that made it easy for unsophisticated attacker to conduct DoS/DDoS attacks, DoS/DDoS attack has become widespread [3]. With the emergence of E-commerce, the DoS/DDoS threat is becoming a serious problem which affects E-commerce service's availability. The Computer Emergency Response Team (CERT), the Internet's leading security watchdog, warns that Distributed Denial of Service (DDoS) attacks pose a major threat to e-commerce and e-business in the future [4]. Once under

attack, the E-commerce servers are out of service for some time and no legitimate clients can access the servers. This eventually results in loss of business, time and money for both clients and enterprises.

In DoS/DDoS attacks, the attackers typically put spoofed source address in the IP header of attack packets. This is known as "IP source address Spoofing" [5] or simply "IP-spoofing". With IP spoofing, the attacker can easily prevent itself from being identified, filtered and traced back by the victim. The difficulty in identifying the genuine source of DoS/DDoS attackers makes it much harder to stop DoS/DDoS attacks and to institute accountability. Traceback is another type of countermeasures that attempts to identify the true IP address of the attacker and the path taken by the attack packets.

In section 1, we present several kinds of (Distributed) DoS attacks. Then in section 2 we introduce the countermeasures against DoS attacks and new approaches in this area.

2.1 DoS/DDoS Attacks

DoS/DDoS attacks aim to deny or limit the legitimate users' access to a certain host or network. Based on the different objective of a DoS/DDoS attack, we can classify the attacks into two categories, namely flood attacks and logic or software attacks [6]. In flood attacks, an Internet host is overwhelmed by a continuous flood of traffic designed to consume resources at the targeted server (CPU cycles and memory) and/or in the network (bandwidth and packet buffers). These attacks result in degraded service or a complete site shutdown. The SYN Flood attack [7], Smurf IP attack [8], and the UDP Flood attack [9] are all well-known example of flood DoS/DDoS attack. While in software or logic attacks, a small number of malformed packets are designed to exploit known software bugs on the target system. These attacks are relatively easy to counter either through the installation of software patches that eliminate the vulnerabilities or by adding specialized

firewall rules to filter out malformed packets before they reach the target system. A well-known example is Ping of Death attack [10]. We describe each of these attacks in the following sections.

2.1.1 SYN Flood Attack

The SYN Flood attack is one of the most common attacks used to deny legitimate TCP connection to a server. This attack takes advantage of the vulnerability in the working mechanism of the TCP 3-way handshake protocol [11] used in the connection phase of TCP protocol. The client initiates the connection and sends a TCP SYN packet to the server. In a TCP SYN packet, the source address of the client is included in the packet header. Upon the receipt of the TCP SYN packet, the server sends a SYN-ACK packet to the source address marked on the TCP SYN packet header. At the same time the server allocates a connection buffer record, storing information for the connection which is waiting to be completed in the connection queue. When the client receives the SYN-ACK packets, an ACK packet is sent out to the server and this completes the connection establishment. This message exchange is called a three-way handshake.

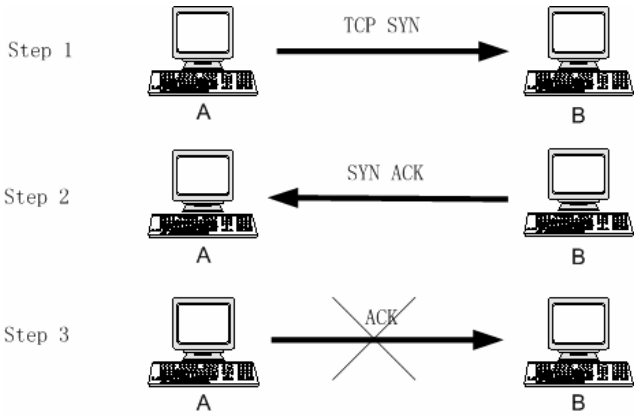

Figure 2.1 TCP 3-way handshake protocol

The potential for abuse lies in the allocation of a data structure describing all pending

connections. This data structure is of finite size, and it can be made to overflow by intentionally creating too many half-open connections (connections whose three way handshake has not completed yet), then the system will be unable to accept any new incoming connections until the table is emptied out. During a TCP SYN Flood attack, the attacker sends a large number of TCP SYN packets with a spoofed source address to the server. Upon receipt of those TCP SYN packets, the server allocates buffer space for each half-open connection until the connection establishment is completed. Because the source address in the TCP SYN packet is spoofed, no ACK packet is sent out and the connection is always in half-open states, so that the buffer space allocated will not be released. Because the attacker generates a multitude of half-open connections, the half-open connections data structure on the victim server system will eventually fill and no more incoming connections can be accepted.

There is no acceptable solution to this problem with the current IP protocol technology. However, one way to avoid this kind of attacks is to install filter in routers which can reduce the number of spoofed IP packets entering and exiting the network. In addition, SYN cookies [12] can also be used as another countermeasure against SYN floods. The three-way handshake requires the sequence number to match between the SYN-ACK and ACK packet to protect against accidentally reopened old connections and unauthorized access. SYN cookies calculate the sequence number as a cryptographic hash value of source address, source port, destination address, destination port, and a destination specific secret key. A server that uses SYN cookies doesn't have to drop connections when its SYN queue fills up. Instead, it sends back the sequence number calculated in the cryptographic way to the requesting client, and the state is not kept in the SYN queue. On a result, the SYN queue is not exhausted and normal TCP communication can continue. When the server receives an ACK, it recalculates the sequence number by using its secret key, the addresses and the ports. If the sequence number recalculated by the receiver

matches the one calculated by the requesting host, the receiver rebuilds the SYN queue entry then. SYN cookies are now a standard part of Linux. However, they are not enabled by default under Linux. To enable them, we need to add a command to the boot scripts [13]. In [14], Schuba develops a software tool synkill that can lessen the impact of SYN flooding attacks by generating RST packets or ACK packets to the victim. The RST packets and ACK packets can release the resources allocated at the victim for the connection establishments. Those actions are in response to observed traffic and the decision making is based on a synkill finite state machine which has four states classified by the source IP addresses of TCP packets-GOOD, NULL, BAD and NEW respectively. NULL means never seen hosts; Good means belonging to correctly behaving hosts; NEW means potentially spoofed addresses; and BAD means most certainly spoofed addresses.

2.1.2 UDP Flood Attack

UDP is a connectionless protocol which does not require any connection setup procedure to transfer data. The UDP flooding attack targets computers providing UDP services. A UDP Flood attack is usually based on UDP ECHO and character generator service (chargen) [15] provided by most computers on the network. Chargen service and echo service are both designed for some performance-testing network programs. Chargen service generates a stream of characters for each packet it receives while Echo service echoes any character it received [16]. When the attacker uses some forged UDP packets to hook up one system's echo service with another system's chargen service, a flood of data passes between the two systems, and the network bandwidth is soon exhausted. And the service provided in the network is denied due to serious congestion.

There is no perfect solution to defeat UDP flood attacks. Recommendations particularly include disabling useless UDP services and blocking UDP ports at firewalls. However, if

one really needs to provide these services, then he cannot protect them. He needs to monitor activity on UDP ports for signs of misuse.

2.1.3 Smurf Attack

ICMP [17] is a protocol used to provide feedback about problems in communication environment. For example, it delivers information about network errors and congestion, it helps to troubleshoot, and it reports IP packet timeouts. ICMP allows checking if a host on a network is responding by sending it an ICMP-ECHO packet. On receipt of an ICMP-ECHO reply packet from the remote host, one can measure the availability and maximum delay time from the remote host. The Smurf attack is a reflector attack that takes advantage of the ICMP-ECHO mechanism. In a Smurf attack, the attacker uses broadcast IP addresses to increase the effect of an attack. Broadcast IP addresses are used to send messages to all the hosts connected to a particular network and they are usually formed by setting all the bits of their host part to 1. By sending a message to a broadcast address, an attacker ensures that the message will be received by many hosts. If these messages require a response, then the attacker will generate a high number of messages as well the hosts will reply. Thus a multitude of machines in this network will receive ICMP_ECHO request and reply to the victim, soon the victim's network is overwhelmed.

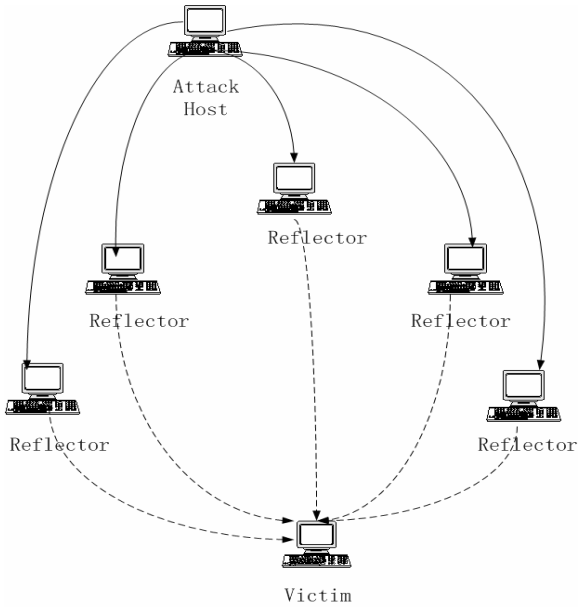

Figure 2.2 Smurf Attack

Unfortunately, there is no easy defensive solution for victims receiving the potentially large number of ICMP echo reply packets. However, we can prevent the smurf attack from occurring to some extent by preventing a site from being used as an intermediary in the attack. The way is to disable IP directed-broadcasts at the routers. By disabling these broadcasts, we can configure the routers to deny IP broadcast traffic onto our network from other networks, and which avoids the attack process to some extent.

2.1.4 PING of Death Attack

Ping of death is a DoS attack caused by an attacker deliberately sending an IP packet larger than the 65,536 bytes allowed by the IP protocol. One of the features of TCP/IP is fragmentation; it allows a single IP packet to be broken down into smaller segments. In 1996, attackers began to take advantage of that feature when they found that a packet broken down into fragments could add up to more than the allowed 65,536 bytes [18]. When an attacker sends an ICMP ECHO request packet that is much larger than the

maximum IP packet size to the victim, since the received ICMP echo request packet is bigger than the normal IP packet size, the victim cannot reassemble the packets and the operating system will freeze, crash or reboot.

For defending against the ping of death attack, the best line of defense it to keep the system patched up. By the end of 1997, operating system vendors had made patches available to avoid the ping of death. In addition, many Web sites block Internet Control Message Protocol (ICMP) ping messages at their firewalls to prevent any future variations of this kind of denial of service attack.

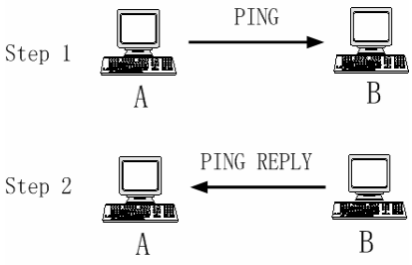

Figure 2.3 Ping of Death Attack

2.2 Taxonomy of DoS/DDoS Countermeasures

From their early days, DDoS attacks have attracted a lot of attention in the research and commercial communities. Many security efforts aim at identifying a certain feature of the attacks to prevent them or to constrain their effect. Generally, the countermeasures can be categorized into four categories: Prevention, Detection, Filtering and Traceback. Figure 2.4 shows the taxonomy of DoS/DDoS countermeasures. We will introduce them in the following sections.

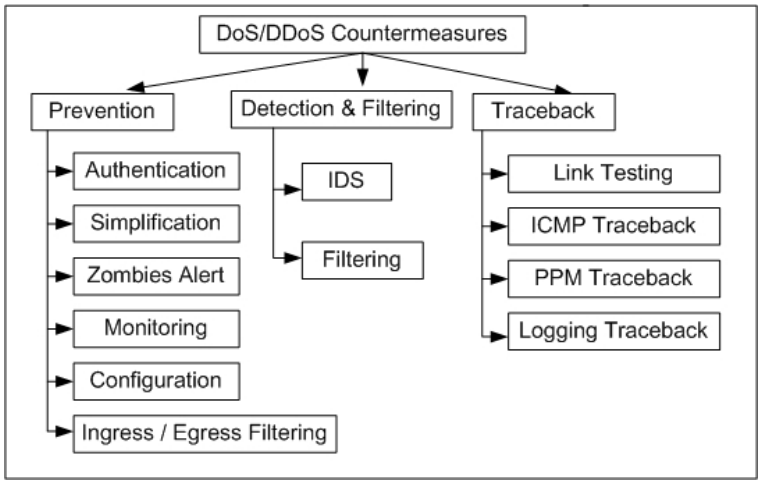

Figure 2.4 Taxonomy of DoS/DDoS Countermeasures

2.2.1 Prevention

Currently, there are many ways that can effectively prevent DoS/DDoS from happening.

- **Authentication**: The inability to authenticate the true source of an originating packet in the TCP/IP protocol suite is an important reason for DoS/DDoS attack occurring. The attacker can easily forge a packet with false source address. A TCP SYN Flood attacker takes advantage of it to send TCP SYN packets with faked source address to the victim. A more effective authentication mechanism may prevent the attack from happening to some extent.

- **Simplification**: Simplification means enabling only essentially services and disabling the rest. Any vulnerability in redundant services may be used by the attackers to conduct DDoS attacks. As we have mentioned before, we may disable the chargen, echo and other unused UDP services to prevent from UDP flooding attack. And to prevent the Smurf attack we can disable IP-directed broadcast at routers.

- **Zombie Alert**: Before conducting a DDoS attack, the attacker needs to select several hosts in the Internet to help perform the attack. The selected hosts are then installed with

a DDoS attack program. We call these hosts "zombies" because once they are corrupted. They become unwitting attackers [19]. The availability of zombies is indispensable for the attacker to launch a DDoS attack. If all hosts on the Internet can be prevented from being compromised, a DDoS attack is much harder to carry out. To guard against DoS/DDoS attacks, hosts must be updated with the latest security patches. Hence we can protect hosts from being used as zombies by the attacker and prevent DDoS attacks to some extent.

- **Regular Monitoring**: We can monitor server CPU load and network bandwidth utilization regularly. Once an abnormal pattern is detected, we should check whether an attack is occurring or whether the host is compromised.

- **System Configuration**: We can decrease the timeout value for the receipt of TCP ACK packet which will help in pruning half-open connections from the SYN queue. This may help to prevent TCP SYN attacks. In addition, significantly increasing the length of the SYN queue may also help the system cope with more simultaneous half-open connections.

- **Ingress Filtering** Preventing IP-spoofing is a key defense against DoS attacks. If we can successfully block specific IP addresses or IP address range, then we can effectively perturb the DoS/DDoS attack. The good way to address the problem of anonymous attacks is ingress filtering which can reduce the ability to forge source address. The concept of Ingress filtering was proposed in RFC 2827 [5]. The idea of Ingress filtering is to configure routers to block packets that arrive with invalid source addresses to reduce or completely eliminate the attacker's ability to use forged source addresses. Ingress filtering requires a router with sufficient power to examine the source address of every packet and sufficient knowledge to distinguish between legitimate and illegitimate addresses. If Ingress filtering can be accomplished successfully, it would

ultimately result in much easier tracing back to the true source of an attack and as such would serve as a significant deterrent for attackers. However, some limitations of the approach still exist. Firstly, it may be the case that an address is forged, but it is still a valid address that corresponds to that of another host on the same network and it cannot be detected. Secondly, the approach when applied to high speed links can become an expensive operation and may reduce network bandwidth. The third drawback is that ingress filtering relies heavily on widespread deployment in the routers in order to be effective. Unfortunately, a significant fraction of ISPs, perhaps the majority, do not implement this service either because they are uninformed or have been discouraged by the administrative burden, potential router overhead and complications with services such as mobile IP [20]. Unless these problems are resolved, there is still a significant need for traceback technology.

2.2.2 Detection

As prevention mechanisms can not provide one hundred percent guarantee, DoS attacks still occur. A critical problem is how to detect the attack quickly and accurately. Usually we use intrusion detection systems (IDS) which can inform us immediately about any suspicious activity, the presence of an intruder, and help to provide accountability for the attacker's actions.

An Intrusion Detection System (IDS) [21] is responsible for detecting inappropriate or other data occurring on a network that may be considered unauthorized. An IDS captures and inspects all traffic, regardless of whether it's permitted or not. Based on the contents, at either the IP or application level, an alert is generated. For instance, Snort [22] is a simple IDS which currently includes the ability to detect more than 1100 potential vulnerabilities. Intrusion Detection Systems can monitor the network for known attack

signatures. An attack signature or packet format is a sequence of events, which is known to occur prior to or during an attack. Therefore, there is no way of knowing the attack signature until at least one site has been attacked.

Besides some feasible tools (snort, etc) several approaches on how to detect Denial of Service more efficiently and accurately have been proposed recently.

In [23], Hussain proposes an attack fingerprinting system to identify instances of repeated attack scenarios by spectral analysis of the arrival stream of attack traffic. To make this identification, firstly, Hussain filters the attack packets and create an attack fingerprint which can be uniquely mapped as a multivariate probability density function corresponding to the attack scenario. The power spectral density is computed by performing the discrete-time Fourier transform on the autocorrelation function (ACF) of the attack segment [23] and is derived from the characteristics of the attack stream which is shaped by many factors: number of attackers, attack tool, operating system, host CPU, network speed, host load, and network cross-traffic. Secondly, Hussain uses the Bayes maximum-likelihood classifier [23] to test if the current attack scenario is similar to a previously registered attack fingerprint in the database.

In [24], Haining Wang and Danlu Zhang propose a simple and robust mechanism for detecting SYN flooding attacks. The mechanism utilizes the inherent TCP SYN-FIN pairs' behavior for SYN flooding detection. According to the specification of TCP/IP protocol, in normal operation, a FIN packet is paired with a SYN packet at the end of data transmission. (The SYN/FIN packets delimit the beginning and end of each TCP connection.) However, under SYN flooding attacks, this SYN-FIN pairs' behavior is violated and the strong positive correlation between SYN and FIN packets offers a clear indication for SYN flooding.

2.2.3 Filtering

Filtering is an effective way to defend against DoS attacks when the attack is in progress. It attempts to mitigate the effect of DoS attack by dropping bad packets. The main components of a filtering approach are the establishment of filtering rules and the installation of a filtering module. The former focuses on setting the most efficient set of filtering rules and its main task is to identify attack packets from good ones precisely. By preferably dropping attack packets, the attack stream can be stopped. A proper set of filtering rules should prevent good streams from collateral damage at best. The installation of filtering module is responsible for setting the filtering function in those routers that can filter most effectively. Some filtering mechanisms [25] [26] [27] have been proposed. In [25], Mahajan proposes an aggregation-based filtering mechanism where the routers preferably drop packets belonging to high-bandwidth aggregates. In [26], Sung proposes a traceback-based filtering mechanism where the routers preferentially drop packets coming from the attack path. (Details of [25] [26] will be given in chapter 3). In [27], Yau proposes a Max-min Fair Server-centric router throttles mechanism. The basic idea of this mechanism is for a server under stress to install a router throttle at selected upstream routers. By asking the selected routers to regulate their own contributing packet rates to more moderate levels, the mechanism can forestall an impending attack. The throttle rates in distributed routing points can be dynamically adjusted. If the current throttle fails to bring down the load effectively, the throttle is reduced, and otherwise increased.

2.2.4 Traceback

Prevention is employed before the occurring of an attack and filtering is used during the attack. Traceback is another kind of countermeasure against the DoS attack. Traceback is usually conducted during or after the attack. Traceback can trace the true location of the

attacker and help in gathering evidence for law enforcement. But the inability to authenticate the true source of an originating packet [28] addresses us a great difficult to trace back to the attacker. Some practical and efficient trace back methods have been proposed. In [29], Bellovin proposes an ICMP traceback technique in which whenever a router forwards an IP packet, it generates an ICMP packet, called ICMP Traceback, with a low probability and sends it to the same destination address as the IP packet with information about the router's backward or forward links or both. Upon reception of a sufficient number of ICMP Traceback packets, the victim can reconstruct the attack path and determine the true IP address of the attacker.

Probabilistic packet marking (PPM) is a traceback method that was first proposed by Savage in [31]. In PPM, routers probabilistically add partial path information in the packets that are forwarded. This information is typically added in the 16-bits Identification field in the IP header. When the victim detects an attack, it can reconstruct the attack path using a sufficient number of marked packets received. In [32], Burch and Cheswick describe a link-testing traceback method. In this method, the victim is assumed to know the network topology. By selectively flooding the various links, the victim can infer the links that carry the attack traffic. This is done by observing how the controlled-flooding perturbs the attack traffic at the victim end. Logging is another way to traceback the real attacker which was proposed in [33]. The basic idea of this approach is that every router stores a record of every forwarded packet. When a victim is attacked, it queries the routers to determine the ones that have forwarded the attack packets so as to determine the attack path. The current solutions can be broadly grouped into four categories: link testing, ICMP traceback, probabilistic marking and logging.

2.2.4.1 Link Testing

Controlled flooding [32] is an ingenious method to trace the source of spoofed attack

traffic. The victim selectively floods links in the network and by observing whether the attack traffic is perturbed at the victim, the attack links can be identified. Usually, controlled- flooding is started from the links nearest to the victim, then recursively applied to upstream links. How to produce an effective and powerful load to make obvious attack perturbation is the key challenge in controlled flooding. Suggested by Burch in [32], the best way is to use the UDP chargen service available on the routers. The Chargen service generates continuous data to anyone who connects to it and the rate of data flow is limited in general by the rate data being acknowledged by the client machine. Though the chargen service is turned off on many Internet hosts and routers, there are still many hosts with the service turned on, and they are easy to find. Before loading a link, cooperative hosts at the right places in the network map in order to produce the required load must be identified. The ideal routers are those which turn on the chargen service and are also on the far end of the link under test as seen from the victim. As illustrated in Figure 2.5, when we want to load the link R7-R8 that is on the attack path, routers R4 and R2 that provide the Chargen services should be identified first. Then we spoof the source addresses of the UDP chargen packets to be the address of router R8. As a result, the chargen response packets will seriously congest the link R7-R8.

Though many traceback approaches have been proposed, most of them require the tedious continued attention and cooperation of each intermediate Internet Service Provider (ISP) for tracing. This is not always easy given the worldwide scope of the Internet. However, controlled flooding can trace spoofed packets to their actual source host without relying on the cooperation of intervening ISPs. This is a great advantage of controlled flooding. However, to effectively load a link, victim must resort to some assistance from the hosts or routers in the Internet, as it is difficult to generate a flow of packets from a single host [8]. Controlled flooding is itself a kind of DoS attack that will influence the legitimate users during traceback.

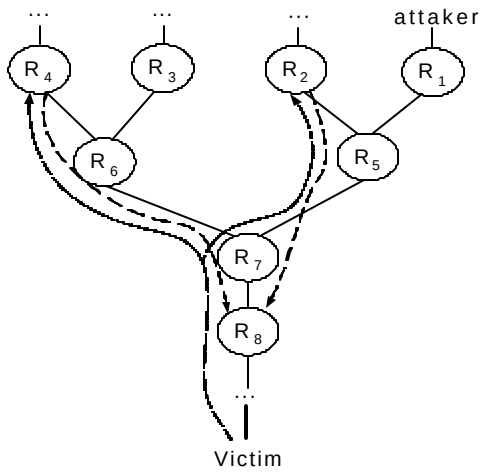

Figure 2.5 Controlled-Flooding Mechanism

2.2.4.2 ICMP Traceback

In the Internet draft [29], Bellovin proposes an ICMP traceback technique where each router picks one packet out of a large amount of IP packets, (say one out of 20,000) and generates a new ICMP packet intended for the same destination address as the selected packet. In those ICMP packets, some information about the back or forward link or both of them is included. With the receipt of a sufficient number of such ICMP packets, the victim can reconstruct the attack path from the information included in them. Another important ICMP traceback scheme called intention-driven ICMP traceback was proposed by Allsion Mankin in [34]. Intension-driven ICMP traceback is a simple enhancement to the IETF draft which introduces an extra bit in the routing and forwarding process to improve the probability that a router sends valuable ICMP packets. Mankin finds that a router cannot always pick up the attack ones to send ICMP packets for instead the router usually tends to pick up some innocent ones due to a potentially large amount of noisy background traffic passing through the routers at the same time. Those ICMP packets corresponding to the innocent packets will be sent to non-victims and become invaluable and usefulness. Usually, the closer the router is to the victim, the higher the probability the router can send the packets to the victim. On the other hand, the routers closer to the

attacker might send ICMP packets to non-victims with a higher probability. So the author proposes an evaluation model to evaluate the value of each ICMP packet before sending them out. Then an intention bit which indicates whether or not the network node is interested in receiving the ICMP messages is introduced. It is added in the BGP routing table and only when the intention bit for a particular route entry is 1, the ICMP packet will be sent. The special intention value can be propagated to routers through Border Gateway Protocol (BGP) updates, thus the probability for a victim to receiving ICMP traceback packets from remote routers is increased. Two modules in this mechanism (Itrace generation and decision module) generate ICMP packets only to those routers that want to receive them and set the intention bit as 1, and try to choose the farther one to the victim out of those willing to receive it. Thus the probability for a router to generate valuable and useful ICMP packets is raised. So we can do traceback more effectively. In [30] Henry Lee proposes an enhanced ICMP Traceback scheme, called ITrace-CP (ICMP Traceback with Cumulative Path) that encodes the cumulative attack path information in the ICMP Traceback message. The ITrace-CP scheme is able to reconstruct the attack path with fewer attack packets.

2.2.4.3 Probabilistic Packet Marking

The key idea of PPM is to probabilistically mark packets at routers (over which the attackers have no control) with partial path information. Once an attack is detected, the entire path could be reconstructed post-mortem at the victim with the receipt of sufficient marked packets. One of the earliest efforts to identify the source of the packet through PPM was done by Savage [31]. The approach allows a victim to identify the network path traversed by the attack packets without requiring interactive operational support from Internet Service Providers. In the marking procedure, the 16 bit identification field in the IP header is used to record a hop count and an edge-id which is the XOR value (32 bits)

of the two IP addresses of adjacent routers. To reduce the per-packet space requirements, Savage suggests subdividing each edge-id into a few smaller non-overlapping fragments. Consequently, the choice for apportioning the identification field is follows: 3 offset bits to present 8 possible fragments, 5 bits to represent the distance, and 8 bits for the edge fragment. In [37], the algorithm uses a 32-bit hash, which doubles the size of each router address to 64 bits. This implies that 8 separate fragments are needed to represent each edge which is indicated by a unique offset value. After the PPM scheme was proposed [31], a number of papers have been published which try to improve the performance of that basic scheme. For example, in [35], Song and Perrig propose two new schemes, the advanced marking scheme and the authenticated marking scheme for tracing back the approximate origin of spoofed IP packets. In the advanced marking scheme, instead of encoding the full IP address of a router into eight fragments as in [31], they simply encode its hash value. The 16-bit IP identification field is divided into a 5-bit distance field and an 11-bit edge field. The advanced marking scheme results in lower computational overhead for the victim to reconstruct the attach paths under large-scale distributed DoS attacks. However, it cannot prevent a compromised router from forging the markings of other uncompromised upstream routers. The authenticated marking scheme solves this problem by authenticating the packet marking. In the process of packet marking authentication, cryptographic MAC (Message Authentication Code) is used. Each router shares a unique secret key with the victim and instead of sending the router address; the router sends a MAC which is a function of its IP address and packet-specific information, and its own secret key to the victim. This presents the compromised router from creating forged packet marking.

Different from the PPM scheme [31] [35] that uses a fixed marking probability, Tao Peng proposes an adjusted PPM scheme in [36]. In this scheme, the marking probability is adjusted in each router according to the distance between the router and the victim, so that

the victim receives packets marked by each router with equal probability. As packets further from the destination victim have a higher probability of being overwritten, the marking probability should increase with increasing distance from the destination. The new scheme successfully reduces the number of packets needed to reconstruct the attack path by 40% compared to uniform PPM.

In approach [37], Franklin reframes the traceback problem as a polynomial reconstruction problem and uses techniques from algebraic coding theory to provide robust methods of transmission and reconstruction. The number of IP authentication headers is reduced from 16 to 13. This scheme does not require an upstream router map to reconstruct the attack path, but it is very vulnerable to fake markings put in the packets by the attackers.

Goodrich presents another approach to do PPM traceback which is called randomized-and-link in [38]. In this scheme, checksum cords are used to "link" message fragments in a way that is highly scalable, for the checksums serve both as associative addresses and data integrity verifiers. The main advantage of these checksum cords is that they spread the addresses of possible router messages across a spectrum that is too large for the attacker to easily create messages that collide with legitimate messages. Therefore the methods scale to attack trees containing hundreds of routers and do not require that a victim know the topology of the routers a priori. In addition, by utilizing authenticated dictionaries in a novel way, the methods do not require routers to sign any setup message individually. Also there are some studies focusing on the tradeoffs for various parameters in conventional PPM. In [39], Abraham Yaar proposes a novel packet marking approach in which a path fingerprint is embedded in each packet, enabling a victim to identify packets traversing the same paths through the Internet on a per packet basis, regardless of source IP address spoofing and which is called Path Identifier (PI). In this per-packet deterministic mechanism each packet traveling along the same path carries the same

identifier. This allows the victim to take a proactive role in defending against a DDoS attack by using the PI mark to filter out packets matching the attackers' identifiers.

2.2.4.4 Logging

The basic idea of logging is that the routers store the actual contents of every packet passing through the router, so that the victim can trace back the origin of the packet by using the history in the router. The drawback with logging approach is that it requires additional storage at the routers. To reduce the low storage requirement, Snoeren proposes a hash-based Source Path Isolation Engine (SPIE) [33] to enable IP traceback by storing packet digests instead of the actual packet contents in the routers. This can also preserve traffic confidentiality as SPIE is prevented from being used as a tool for eavesdropping. Sneoeren's research shows that the first 24 invariant bytes of a packet (20-byte IP header with 4 bytes masked out plus the first 8 bytes of payload) are sufficient to differentiate almost all non-identical packets, thus SPIE computes digests over the 24 bytes. If a packet is determined to be offensive by some intrusion detection system, a query is dispatched to SPIE which in turn queries routers for packet digests of the relevant time periods. The query information includes a packet, victim and the time of attack. The results of this query are used to build an attack graph that indicates the packet's source.

2.3 Summary

Denial of Service is a kind of attack which attempts to deny the users from legitimate service or just disrupt a service. Based on the scale of DoS attack, DoS attacks fall into two categories which are simple DoS and Distributed Denial of Service. While a simple DoS attack only involves one attacker, a DDoS attacker uses numerous coordinated hosts to carry out the attack. In this kind of attacks, the real attacker commands a set of

compromised hosts to make them execute an attack against a single target, thus multiplying the effectiveness of the attack. There are many kinds of DoS attacks and we classify the attacks into two categories: host-based attacks and network-based attacks. In host-based attacks, the attacker aims to crash the server by exhausting the server's resources, such as its CPU utilization, memory or file storage. SYN Flood Attack is a well-known example. While in network-based attacks, the attacker tries to consume all available network bandwidth connecting the target to the rest of the Internet by sending plenty of useless packets to the host. Well-known examples include UDP Flood Attack, Smurf IP Attack, and ICMP Flood Attack. The serious consequence that DoS attack brings has gained more and more attention in both the research and commercial communities. Many effective countermeasures have been developed. The main countermeasures include prevention, detection, filtering and traceback. Prevention method is carried out prior to the occurring of the attack and there are many ways to prevent the attack from happening in advance. Ingress/Egress filtering, authentication, simplification and configuration etc are all effective in attack prevention. Detection and filtering are another ways to defend against DoS attacks and they usually are conducted when DoS attack is raging on. Intrusion Detection System is a widely implemented system in detecting attacks. Traceback is an after-term defense mechanism and many different approaches have been proposed. There are four common kinds of traceback methods which are link testing, logging, PPM and ICMP traceback.

In this chapter, we have introduced several common kinds of (Distributed) DoS attacks - SYN Flood Attack, UDP Flood Attack, Smurf IP Attack, and ICMP Flood Attack respectively. And we have also introduced the countermeasures against DoS attacks from the aspect of prevention, detection, filtering and traceback respectively.

CHAPTER 3
HTF - DOS VERSION

In this chapter we propose the Hybrid Traceback-Filtering (HTF) DoS defense mechanism that is able to improve the throughput of legitimate traffic with minimum collateral damage and bandwidth consumption in upstream links. This mechanism leverages on the attack path constructed by the controlled flooding traceback method [32] to enable the routers to filter intelligently. The controlled flooding scheme selectively floods links based on the network topology, then identifies the attack links by observing whether the attack stream is perturbed by the flooding. With the knowledge of the whole or partial attack path, the scheme recursively activates a filtering function in the routers that are on the attack path. By filtering out the packets coming from attack links in chosen routers, the attack stream can be effectively controlled while protecting the legitimate traffic. Compared to conventional approaches where filtering is only effected at a single level of routers, our mechanism can save bandwidth in upstream links effectively when filtering on routers further away from the victim are activated. To evaluate its effectiveness in defending against DoS attacks, the scheme is simulated in ns2 [40]. Simulation results show that the throughput of the legitimate traffic is increased while the false positive ratio and upstream link bandwidth consumption decrease. This approach can mitigate the effect of DoS attacks in real time, as the filtering function is activated immediately once the first attack link is identified.

Section 3.1 introduces a few related works. Section 3.2 gives an overview of the hybrid mechanism. In section 3.3 we evaluate the performance of our mechanism. Section 3.4

concludes the chapter.

3.1 Related Work

In [25], Mahajan and Bellovin propose an aggregate-based filtering mechanism where the congested aggregates (high-bandwidth aggregates) are in fact responsible for congestion is an assumption. The idea behind this approach is to identify the signature of the high-bandwidth aggregates, and then preferentially drop packets belonging to these aggregates in routers. An aggregate is a collection of packets from one or more flows that have some properties in common and the property could be anything from destination or source address prefixes to a certain application type (streaming video, for instance). The aggregate-based Congestion Control (ACC) module in each router identifies the signatures of the congested aggregates by analyzing the statistics of dropped packets, and then drops the packets with the congest signatures. The authors also extend the simple ACC mechanism to a Pushback form. In pushback, a filtering request is recursively propagated from downstream routers to upstream routers hop-by-hop. The bandwidth of upstream links is saved as bad packets are dropped in upstream routers before arriving at downstream routers; however, the whole procedure is complex. Firstly, in [25], to decide which upstream router should filter, a downstream router needs to contact all its upstream routers to identify which one(s) contributes the major part of the traffic load. This consumes a lot of computational resource at the routers and leads to link bandwidth consumption in the transfer of contact information (dummy packets and status packets) between upstream and downstream routers. Secondly, identifying the signature of high-bandwidth aggregates is resource-consuming as much work needs to be done in information extraction and analysis of dropped packets.

In [26], Sung proposes a technique which leverages on the information concerning

whether a network edge is on the attacking path of an attacker ("infected") or not ("clean"). While an attacker will have all the edges on its path marked as "infected", edges on the path of a legitimate client will mostly be "clean". By preferentially filtering packets that are inscribed with the hash value marks of "infected" edges, the proposed scheme attempts to remove most of the DoS traffic while affecting legitimate traffic only slightly. However, in this method the victim relies on the receipt of a large number of attack packets to identify an attack edge, which brings about a high reactive delay against DoS attacks if the probability for routers to mark packets is not high. Meanwhile, as the malicious packets will traverse a lot of upstream links before reaching the two-hop-away routers, the upstream link bandwidth cannot be effectively saved.

3.2 Overview of HTF-DoS

In this section, we propose a controlled flooding-filtering hybrid traceback-filtering mechanism (HTF). The idea of this mechanism is to preferentially drop packets coming from attack links in the routers which are on the attack path, and in so doing, the attack stream is effectively stopped while the good stream is protected from collateral dropping. Here we use the controlled flooding method to identify the attack links. In HTF, the filtering function is activated recursively from downstream routers to upstream routers hop-by-hop and the activation follows the sequence as controlled flooding. The synchronization of filtering and controlled flooding can effectively decrease the reaction latency against DoS attacks because the filtering function can be activated by the identification of a single attack link instead of whole attack path. As the filtering function is activated in upstream routers that are close to the attacker, upstream link bandwidth consumed by attack packets can be greatly saved. In the rest of this section, we explain why controlled flooding is chosen as the traceback method in our mechanism, and then we introduce the functionality in routers and victims respectively.

3.2.1 Motivations to Select Controlled Flooding

As we introduced in section 2.2.4, there are four different kinds of ways to trace back the original source(s) of an attack, which are logging, PPM, ICMP traceback and controlled flooding respectively. Although each traceback method can be used to provide us the attacking route information and help enhance filtering to some extent, in HTF, we only use controlled flooding as our traceback method. The reason lies in the hop-by-hop way that the controlled flooding method is carried out, which is consistent to our expectation of the HTF mechanism. We aim to propose a hybrid mechanism which is an integration of traceback and filtering mechanism and which can also react to DoS/DDoS attack with a low reaction delay and put a little burden on the routers meanwhile. Controlled flooding starts from the links nearest to the victim and then recursively propagate up to the edges furthest from the victim a hop-by-hop. After each step of controlled flooding, a partial route of attack path (one edge) is identified which enables the scheme to propagate the filtering request to upstream routers recursively hop-by-hop. This feature enables us to create a tight connection between traceback mechanism and filtering mechanism filtering follows the step of traceback. Logging can also provide us the hop-by-hop feature as the victim can contact the upstream routers recursively and enable the filtering function simultaneously. However, the storage requirement for the routers in logging traceback is extremely high and we do not consider it currently. PPM and ICMP traceback can also enhance filtering and some works have been done in this field. As we introduced in section 3.1, [6] proposes a way to use PPM to improve output of legitimate traffic. However, the filtering function is totally non-integrated with the traceback function. Only after the traceback module completes its task, filtering function starts to work, which is not consistent to our idea of introducing a tightly-hybrid mechanism. In addition, it also brings a high reaction delay. In HTF, we evoke filtering function once a single link is identified as a bad one which enables the mechanism reacts to the DoS/DDoS attack

faster compared to the other mechanism using PPM or ICMP. In PPM or ICMP, the filtering function is idle until the full attack route is reconstructed. In addition, PPM and ICMP traceback put a great burden in the routers. The routers are responsible for pending partial path information in a packet on the fly or generating a new packet while in controlled flooding, the routers can be released from such works. However, we are not saying that using PPM or ICMP to enhance filter is not feasible; the reason we don not select them is based on our own concerns. In fact, using PPM or ICMP has its own advantages. PPM and ICMP will not introduce much extra traffic to traceback like controlled flooding does. Generally, how to use traceback method to enhance filtering is worthy research aspect and our future work may include investigating the feasibility of the combination between filtering and other traceback method (PPM or ICMP).

In this paper, we use controlled flooding based on the consideration of low reaction delay, low consumption of routers and tightly-hybrid characterization. The use of PPM or ICMP is an alternative if the extra traffic produced by controlled flooding is not acceptable in some cases. We will discuss the damage brought by controlled flooding in simulation section and by studying the potential damage brought by controlled flooding in a special case, a wise decision on whether or not to use controlled flooding in terms of potential damage can be made.

3.2.2 The Functionality of the Victim

The main functionality of the victim includes controlled flooding functionality, attack link identification functionality, and filtering signal generation functionality.

3.2.2.1 Controlled Flooding Functionality

In our scheme, starting from the router most close to the victim, we apply a brief burst of

load to each link which is attached to the nearest router, and then recursively apply the link loading to all the upstream links. Three functions are required in the victim to conduct successful controlled flooding. Firstly, it requires that the victim be able to start the controlled flooding procedure correctly. The victim should be able to activate trace back as soon as the attack is detected. Secondly, it requires the victim find out voluntary cooperators in the Internet that can assist in flooding specific link effectively. To effectively load a link, the victim must resort to some assistance from the hosts or routers in the Internet, as the victim is unable to flood all the links effectively on its own ability. It must identify cooperative hosts that are willing to perform the task and are at the right place in the network map in order to produce the required load. The ideal routers are those that turn the chargen service on and are also on the far end of the tested link seen from the victim (Details refer to Chapter 2). Thirdly, the victim should be able to specify a link to test at each step. In selecting a tested link, the idea is that if a link is identified to be a good one then there is no need to test all the links behind it from the side of the victim. Otherwise, all the links behind it need to be tested in next steps. Lastly, once the attack is restrained into an acceptable range, the traceback should be stopped timely. If the attack has been effectively controlled by means of filtering, no upstream routers need to filter, thus no further controlled flooding should be conducted. Otherwise, further controlled flooding should be done.

3.2.2.2 Attack Link Identification Functionality

This functionality requires the victim to observe the effect caused by controlled flooding. By observing changes in the attack streams, the attack link identification functionality can identify which links are bad. The assumption here is that if the loaded link is a component of the path of the attacking stream, the induced load will perturb the attacking stream. Therefore by observing the statistics of the attack stream before and after link loading, it

can deduce whether the tested-link is used by the attacker to send malicious packets. If the intensity of the stream is unperturbed by the load, it is unlikely to be the attack link. However, if the stream is altered when we load a link, then this link is likely to be on the path from the attacker to the victim.

3.2.2.3 Filtering Signal Generation Functionality

The function of filtering signal generation is to control filtering in different routers. It enables the victim to generate two different kinds of signaling packets. We define two new types of ICMP packets: ICMP filter-on and ICMP filter-off. The ICMP filter-on packet activates the filtering function in the destination router while the ICMP filter-off packet deactivates it. After a link is identified as being on the attack path, the victim generates an ICMP filter-off packet for the downstream router which was activated in the previous step and then generates an ICMP filter-on packet for the upstream router which is chosen to filter in the current step. An ICMP filter-on packet includes two items of information: the IP addresses of the routers at the ends of an attack link, and the Base Value (BV) as a tuning parameter based on which routers adjust their filtering probability. BV depends on the hop distance from the router to the victim. The direct upstream router of the victim is specified as level-1 node while routers with a two-hop distance from the victim are level-2 nodes, so on and so forth. Generally, a higher BV is assigned to a higher-level router, while a smaller BV is assigned to a lower-level router. Assume BV_i refers o the base value of the level-i routers, and n is the highest level, then

$$BV_{i+1} = BV_i + \Delta_{i+1} \text{ with } 0 < BV_i < 1 \text{ and } 0 < \Delta_i < 1 (i = 1,2,\Lambda\ n)$$

As higher-level routers are closer to the attacker than lower-level routers, the precision in differentiating bad packets of high-level router is higher; they should be assigned with

relatively higher BVs. On the contrary, lower-level router should be assigned with relatively lower BV to minimize the probability of dropping good packets.

3.2.3 Functionality of the Routers

The main functionalities of the routers include Decision Making Functionality and Packet Dropping Functionality.

3.2.3.1 Decision Making Functionality

The decision making functionality enables the victim to decide the type of packets that should be dropped and in what percentage. Generally, it performs two tasks. The first task is to classify the packets passing through the router. Consider that the router knows the attack edge. We represent the destination address of a packet and the incoming link of the packet as **dest** and **link** respectively. The address of the victim is denoted by **Avic**, and the combination of all the attack links is denoted by **Latt**. The various parameters are illustrated in the following table.

Table 3.1 Parameters used in Decision Making Functionality

Parameters	**Meaning of the parameters**
dest	the destination address of a packet
link	the incoming link of a packet
Avic	The address of the victim
Latt	the combination of all the attack links

In our scheme, we classify the packets into four categories.

Type A: $link \in L_{att} \& dest = A_{vic}$

Type B: $link \in L_{att} \& dest \neq A_{vic}$

Type C: $link \notin L_{att} \& dest = A_{vic}$

Type D: $link \notin L_{att} \& dest \neq A_{vic}$

Corresponding to the four types of packets, we calculate four different probability values.

P_a: probability of passing packets of type A

P_b: probability of passing packets of type B

P_c: probability of passing packets of type C

P_d: probability of passing packets of type D

The values of the four probability parameters are determined by the following:

1. We define four global probability parameters: P_a', P_b', P_c' and P_d'. We assume that all the routers in our mechanism have the same set of values of these parameters and that the values are all in the range of 0 to 1.0.

2. According to the conditional probability of certain types of packets to be attack packets, we can set different values of the four parameters: P_a', P_b', P_c' and P_d' respectively. Clearly, type A packets are most likely to be attack packets, therefore, the value for P_a' should be highest. Similarly, type D packets are most likely to be good packets, hence the value of P_d' should be the smallest. And the values of P_b' and P_c' should be set between P_a' and P_d'.

3. Initially, P_a' is set to a large percentage value (1.0 in the simulation) to effectively stop type A streams. P_b' and P_d' are set to a very small value (0.0 in the simulation) to pass type B or D packets as most as those packets are not destined to the victim. P_c' can also be set to a small value (0.0 in the simulation) as these packets are identified as legitimate ones in our scheme. However, as the precision in identifying an attack link cannot be guaranteed which may be affected by many factor, we should increase the value for P_c' slightly.

4. All the values should be decided based on the base value passed by the victim. Suppose that the base value set for a router is BV, then all the probabilities for this router to drop packets are calculated as:

$$P_a = BV \times P_a'$$

$$P_b = BV \times P_b'$$

$$P_c = BV \times Pc'$$

$$P_d = BV \times Pd'$$

3.2.3.2 Packet Filtering Functionality

Based on the information generated from the filtering decision functionality, the preferential packet filtering is carried out. Figure 3.1 presents a pseudo code of the filtering algorithm.

3.3 Performance Evaluation

In this section, we evaluate the performance of our hybrid mechanism under a DoS attack using the ns2 simulation software. First we explain the common configuration settings of routers and hosts in our simulation, and the attack model used in our simulation. Then we present some performance metrics used in the simulation. Finally, we present the results of our simulation.

```
For each incoming packet
  If link ∈ Latt
     If dest = Avic
            Pass the packet with probability Pa
     Else
             Pass the packet with probability Pb
Else   If dest = Avic
            Pass the packet with probability Pc
          Else
            Pass the packet with probability Pd
```

Figure 3.1 Packet Filtering Functionality

3.3.1 Common Configuration

For our simulation, we use a binary tree topology construction as our network model. In this topology, there are 256 hosts, 255 routers and 1 victim as shown in Figure 3.2. We choose our attacker from 256 hosts randomly, and the other hosts are assumed to be legitimate ones.

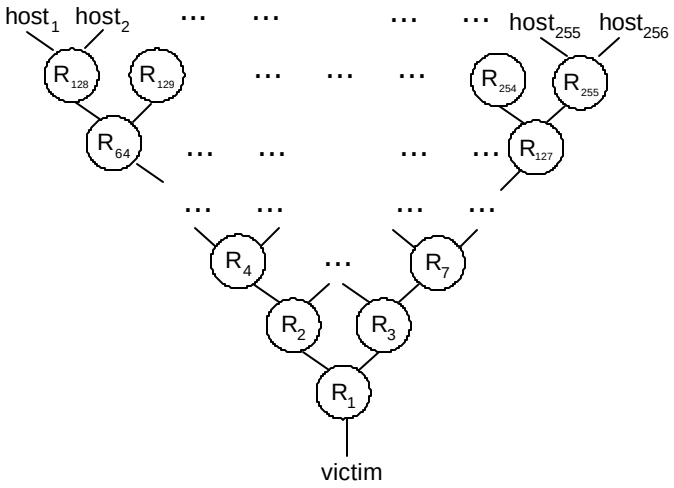

Figure 3.2 Network Topology in NS-2 Simulation

The basic configuration parameters of all the legitimate hosts are the same. They are assumed to have the same packet generation rate in each run and the same maximum packet size. Specifically, to create a successful attack that can generate congestion effectively, we set the packet generation rate of the attacker to a much larger value than the legitimate ones. In section 3.3 we show the effect of different packet generation rates on the performance metrics which are introduced in section 3.3.2. In our mechanism the routers execute the same tasks, and we set the same queue length. In this simulation, we have 511 links with the same link bandwidth and link delay. Detailed configuration values are given in Table 3.2.

3.3.2 Performance Metrics

To evaluate the performance of our mechanism, some performance metrics are introduced. Firstly, to show the advantage in improving the throughput of legitimate packets, we introduce two metrics: the Good Packets Percentage (GPP) and the Bad Packets Percentage (BPP), which are the percentage of good and bad packets respectively. These two metrics are measured at the victim and we have

Table 3.2 Common Configuration Values

Bad Packets Generation Rate	R_b	50,100,200,500,1000 packets/second
Good Packets Generation Rate	R_g	1 packet/second
Maximum Packet Size	S_{pkt}	500 bytes
Maximum Length of Packet Queue in routers	L_{queue}	64
Maximum Delay time of a router	T_{delay}	30ms
Link Bandwidth	B_{link}	1Gb/s
Link Delay	T_{link}	100ms
Base value to 1-hop away router	BV_1	0.3
Increment	?	0.1

BPP+GPP=1 by definition. As our mechanism strives to release dropping legitimate packets, we evaluate our mechanism on the effectiveness of dropping bad packets and protecting good ones. We define Good Packets Dropping Percentage (GPDP), which is actually the False Positive Ratio, and Bad Packets Dropping Percentage (BPDP). In HTF, we recursively propagate the filtering request to upstream routers to save the bandwidth consumption of upstream links, so we also evaluate the performance of bandwidth consumption theoretically. In the rest of the section we present our results for each metric respectively.

3.3.3 Simulation Results and Analysis

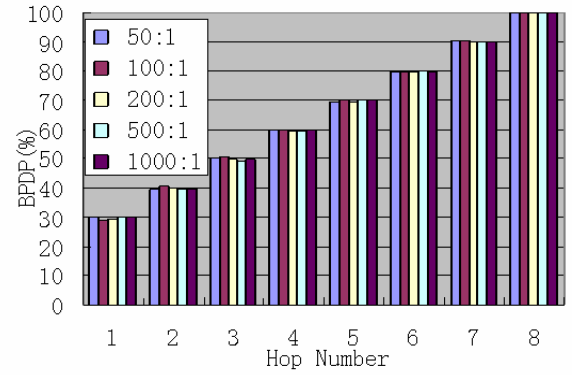

Figure 3.3 Bad Packet Drop Percentage in DoS cases

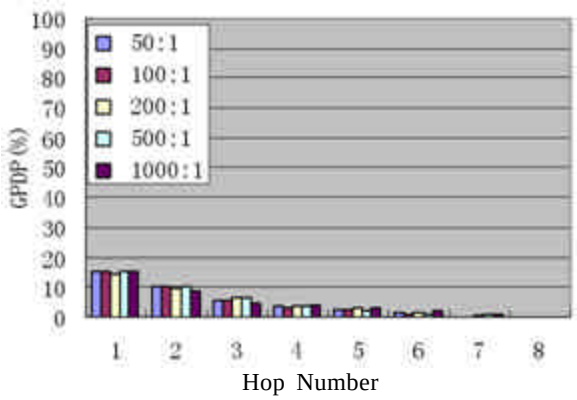

Figure 3.4 Good Packet Drop Percentage (False Positive Ratio) in DoS cases

Figure 3.3 and Figure 3.4 show the BPDP and the GPDP as a function of the hop number respectively. As we can see from Figures 3.3 and 3.4, BPDP and GPDP at a given hop are almost the same when the attackers generate bad packets at different rates. For instance, when the first-level routers filter, the BPDPs and GPDPs are about 30% and 15% respectively (Detailed values are shown in Table 3.3 and Table 3.4). That is because routers drop packets with the same probability despite the packets generation rate. As shown in Figure 3.3, when the hop number increases, BPDP increases. For instance, BPDP is about 30% when first-level routers filter, and then gradually increases to 100% when last hop is reached. That is consistent with the dropping probability used in our simulation which ranges from 0.3 to 1. Similarly, as we can see from Figure 3.4, while

hop number increases, GPDP decreases instead. For instance, when first-level routers filter, GPDP is about 15%; at the last hop, the value decreases to about 0%. The reason for GPDPs decreasing with increasing dropping probability of a router is that higher precision for higher-level routers helps identify bad packets which makes up for collateral damage. Higher-level routers are more accurate in differentiating good packets from bad packets by their incoming links as the bad and good streams are not highly integrated at this point. Even though the higher-level routers drop packets heavily, good packets are discarded less.

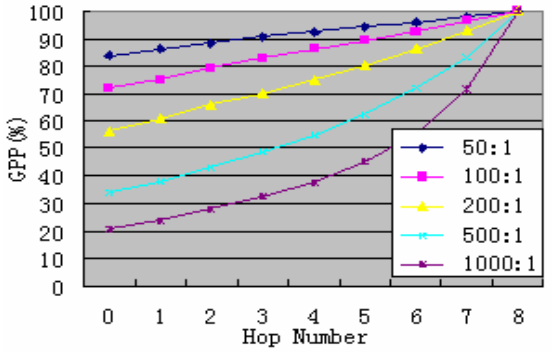

Figure 3.5 Good Packets Percentage (GPP) in DoS cases

Table 3.3 Bad Packet Drop Percentage (BPDP) in DoS cases

Ratio	N_{hop}=1	N_{hop}=2	N_{hop}=3	N_{hop}=4	N_{hop}=5	N_{hop}=6	N_{hop}=7	N_{hop}=8
50:1	30.20	39.48	50.50	60.03	69.29	79.67	90.16	100.00
100:1	29.11	40.59	50.57	60.01	70.05	79.64	90.14	100.00
200:1	29.67	40.04	49.57	59.42	69.72	79.78	89.99	100.00
500:1	30.11	39.46	49.50	59.56	69.85	80.21	89.91	100.00
1000:1	30.04	39.62	49.97	59.81	70.25	79.76	90.02	100.00

Table 3.4 Good Packet Drop Percentage (GPDP) in DoS cases

Ratio	N_{hop}=1	N_{hop}=2	N_{hop}=3	N_{hop}=4	N_{hop}=5	N_{hop}=6	N_{hop}=7	N_{hop}=8
50:1	15.07	10.04	5.57	3.47	2.52	1.33	0.11	0.0039
100:1	15.31	9.99	5.82	3.13	2.62	0.95	0.23	0.0078
200:1	14.15	9.77	6.40	3.45	2.79	1.52	0.59	0.0039
500:1	15.32	10.16	6.39	3.76	2.24	0.97	0.82	0.0270
1000:1	15.22	8.73	4.51	4.23	3.06	2.04	0.84	0.0039

Table 3.5 Good Packets Percentage (GPP) in DoS cases

Ratio	N_{hop}=0	N_{hop}=1	N_{hop}=2	N_{hop}=3	N_{hop}=4	N_{hop}=5	N_{hop}=6	N_{hop}=7	N_{hop}=8
50:1	83.67	86.12	88.35	90.68	92.49	94.18	96.12	98.11	100.00
100:1	71.87	75.29	79.44	82.93	86.07	89.24	92.54	96.27	100.00
200:1	56.25	60.88	65.74	70.30	75.21	80.361	86.13	92.68	100.00
500:1	33.87	38.19	43.08	48.57	54.83	62.32	71.85	83.37	100.00
1000:1	20.92	23.61	27.82	32.73	37.80	45.38	55.24	71.71	100.00

Figure 3.5 shows GPP as a function of the hop number. The five curves in the figure correspond to 50:1, 100:1, 200:1, 500:1 and 1000:1 respectively, which are the ratio of the rate that bad packets are generated over the rate that good packets are generated. As shown in Figure 3.5, when the hop number increases, GPP increases gradually (Detailed values are shown in Table 3.5). For instance, when the ratio is 1000:1, GPP increases from 21% to 100%, which represents an improvement of 378%. The higher GPP reflects the increasing effectiveness of HTF. Particularly, when the highest-level routers drop all the bad packets (BVis set to 1), GPP finally reaches 100%.

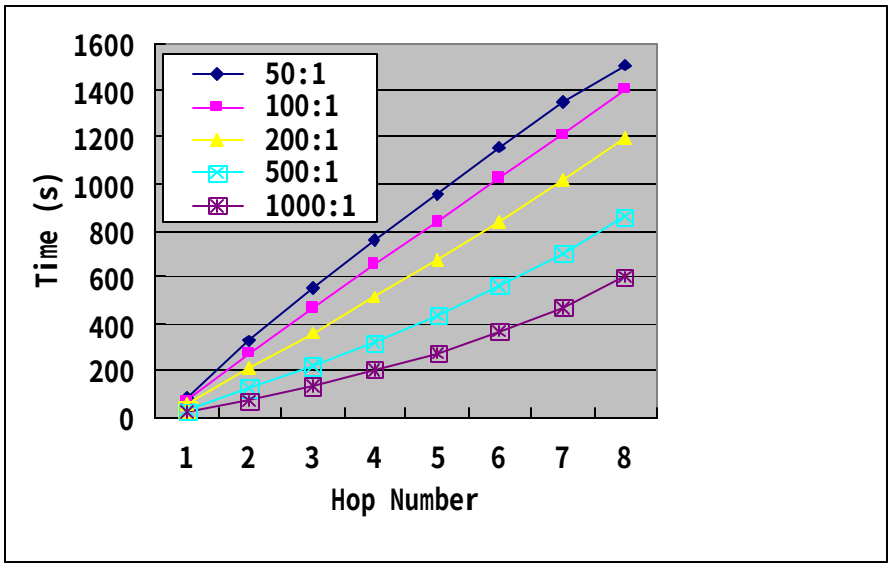

Figure 3.6 Simulation Time

Figure 3.6 shows the time consumption in the simulation of the HTF-DoS mechanism as a function of the hop number. As we can see from the above figure, there are five curves which correspond to 50:1, 100:1, 200:1, 500:1 and 1000:1 respectively. Each ratio represents the ratio of the rate that bad packets are generated over the rate that good packets are generated, which is the same as that in figure 3.5.

As shown in Figure 3.6, when the hop number increases, time increases gradually due to the progress of the simulation. In addition, the curve corresponding to a smaller ratio value is always above the one corresponding to a greater ratio value during the whole simulation progress. The curve corresponding to 50:1 is the highest curve and the curve corresponding to 1000:1 is the lowest one among the five curves in Figure 3.6. This is due to the method we use to simulate the functionality of the victim, specifically, the Attack Link Identification Functionality. As we introduced in section 3.2.2, the attack link identification functionality of the victim is to identify the bad links by observing changes in the attack streams. For simplicity, in this simulation, we identify the bad links by counting the percentage of the number of the bad packets over the number of all packets. The change of the percentage of the bad packets is a sign of perturbation of the bad streams, and also a sign of bad link identification consequently. To identify the bad links more precisely, we count the percentage of the number of the bad packets only when the number of all the packets reaches a certain threshold instead of counting it at a random time point. According to the rational of the probability theory and mathematical statistics, the result is more accurate when the sample space is bigger. By setting a threshold when we need to count the bad packets percentage, we can guarantee the sample space is large enough to produce an accurate enough result. That is to say, a proper setting of threshold value for the total number of the packets can guarantee the bad packets percentage represents the change of an attacking stream accurately. We use the same counting threshold whatever the rate that bad packets are generated, which results in the different process time of the HTF-DoS mechanism with different bad packets generation rate. When the rate of bad packets generation is higher, it is faster to meet the requirement of counting threshold and the decision can be made in a shorter time, which results in a shorter process time. On the contrary, when the attacker generates the bad packets in a

lower rate, the process time is increased. That is why the curve corresponding to a smaller ratio value is always above the one corresponding to a higher ratio value.

From figure 3.6, we can see that when the ratio is 1000:1, our mechanism processes to the first-level routers when the simulation time is around 21 and processes to the last level when the simulation time is around 778. As we said in section 2.2.4, link testing itself is a kind of DoS attacks. It brings temporarily congestion to the internet links when it is carried on, which results in serious packets lost or even link inaccessibility. That is to say, during the period of 778 simulation time, there are always some links suffering from link testing. And the period lasts even longer when the bad packets generation rate is lower. The potential damage brought by link testing is the main downside of our mechanism . However, it doesn't deny the feasibility of the HTF-DoS mechanism. The reason is as follows, from the figure 3.6 we can see that the HTF-DoS mechanism takes effect at simulation time 21, which shows that our mechanism can react to a DoS attack timely. On simulation time 21, the DoS attack is under control already. During the period from time 21 to time 778, there are some links suffering from link testing; however there are still some links released from DoS attack by the filtering functionality at routers. The damage brought by the link testing, while at the same time and the benefits some links can get from the link testing interact together during the whole process time. Therefore, we can not take the period of 708 simulation time as a totally disaster time. Moreover, we can understand that after the whole process completes all the effect that the HTF-DoS mechanism brings is in the bright side. As the damage brought by link testing is always a big concern, we have some suggestions in implementing the HTF-DoS mechanism. If the current DoS attack tends to last a long time, say, much longer than 778s, we can definitely use the HTF-DoS mechanism, as the link testing time is neglectable compared to the whole attack period. In a long run, it is worthwhile to temporarily congest some

suspicious links in order to release the network from the attack which may last for a long period of time. When it is not sure of the period that a DoS attack may last, or it tends to last shorter than 778s, the decision is not that straight forward and we need to consider it carefully. However, the benefits that the HTF-DoS mechanism brings and unique characters that the HTF-DoS mechanism has still qualify the implementation of the HTF-DoS for our consideration. Especially, when there is a requirement of short reaction delay and few burdens on the routers, and even when other mechanisms are not applicably at current time, the use of the HTF-DoS mechanism is still a good choice. Moreover, regarding the process time, we need to mention that it represents only the simulation time, which may be further reduced in real implementation. In this simulation, we use a very dump method to imitate the link identification functionality, which costs a lot of unnecessary time to wait the total number of the packets exceeding the threshold. In realistic implementation, we can use the real monitoring tools which can detect the change of the bad stream timely, and in so doing, we can save the process time greatly. The bad link identification functionality is the key functionality of the HTF-DoS mechanism, which is called in each step. If we can reduce the process time of bad link identification functionality even slightly by introduction of real monitoring tools, the whole process time is expected to be reduced greatly and the potential damage brought by link testing can be reduced consequently.

3.3.4 Theoretical Analysis

Our mechanism also saves upstream bandwidth consumption compared to those schemes where only the low level routers filter. We assume the router i hops away from the victim drops the type A packets with the probability Pai, drops B, C and D types of packets with zero probability respectively. Suppose the bad packet generation rate is Rb and the good

packet generation rate is Rg. In an i-level binary tree topology where the number of attackers is Na and the number of non-attackers is Nn, the upstream link consumption C in our mechanism is as follows:

$$C = (N_a \cdot R_b + N_n \cdot R_g) + (Na \cdot R_b \cdot (1 - Pa_{I-1}) + N_n \cdot R_g) \cdot (I-1)$$

Before arriving at the highest-level routers, all streams (good and bad) traverse without packet being dropped. Therefore the bandwidth consumption at highest-level link is (Na • Rb + Nn • Rg). After passing through the highest level, PaI-1 of bad packets are dropped. So the bandwidth consumption of each following level is Na • Rb• (1-PaI-1) + Nn • Rg. Altogether, the total link bandwidth consumption caused by HTF is given in above formula.

Let's consider the case that only the router one-hop away from the victim filters and drops all A type of packets with probability P of 1, then the upstream link consumption C' is as the following:

$$C' = (Na \cdot Rb + Nn \cdot Rg) \cdot (I-1) + (Na \cdot Rb \cdot (1-P) + Nn \cdot Rg)$$

Because no packet is dropped until reaching the level-1 routers, the total bandwidth consumption from level-I to level-1 is (Na • Rb + Nn • Rg) • (I-1). After passing through level-1 routers, P of bad packets are dropped, so the bandwidth consumption of the link between victim and level-1 routers is Na • Rb• (1-P) + Nn • Rg.

The values of C and C' according to different values of Rb are shown in Table 3.6.

Table 3.6 Values of C and C'

	R_b=50	R_b=100	R_b=200	R_b=500	R_b=1000
Consumption	2345	2395	2495	2795	3295
Consumption'	2695	3095	3895	6295	10295

As we can see from Table 3.6, the upstream bandwidth consumption is greatly saved in our mechanism by 12.99%, 22.62%, 35.94%, 55.60% and 67.99% when Rb is 50, 100, 200, 500 and 1000 respectively. And the higher rate for the attacker to generate bad packets, larger percentage of link bandwidth is saved. Hence we can say our scheme works even better when the attacker is stronger on the metric of bandwidth consumption.

3.3.5 Feasibility in Real Network

In this simulation, we use a binary-tree topology as our network construction. When we implement the HTF-DoS mechanism in real network, we need no changes in the design of HTF-DoS itself. The illustration is as follows.

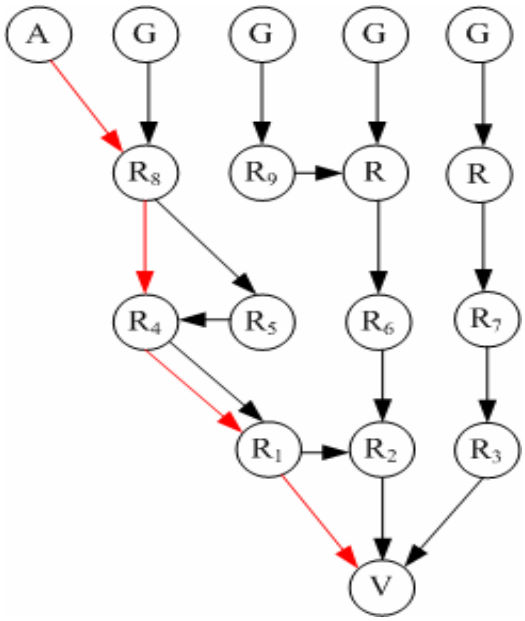

Figure 3.7 Random Network Topology in DoS Cases

In a random network topology as shown in figure 3.7, there is one attacker A and some good hosts marked as G. The route that the attacking traffic takes is marked red while the routes that are not affected by the attacking traffic are marked black. When implementing the HTF-DoS mechanism in a random network topology, for instance figure 3.7, we need

not change any functionality in the routers or the victim. Specially, as for the filtering signal generation functionally of the victim, victim is responsible to generate two kinds of different signaling packets to control filtering and link testing in different routers. Once a link is identified as a bad one in a certain level, the according ICMP packets are sent out by the victim immediately. The victim will not test other links in the same level anymore and propagate the mechanism to upstream levels directly. Whatever how many links are connected to a router (in a random graph, the number of the links connecting to a router is uncertain), the method works well which will not let any suspicious links untested. The reason lays in a DoS attack the attack traffic has only one source and comes from only one direction. Once an attack link is identified, there is no possibility of other attack links existing besides this one. Hence whatever the topology is, ultimately, we can traceback the attacking source A and evoke filtering module in router R8. The red lines in figure 3.7 show the way that filtering function is evoked consequently.

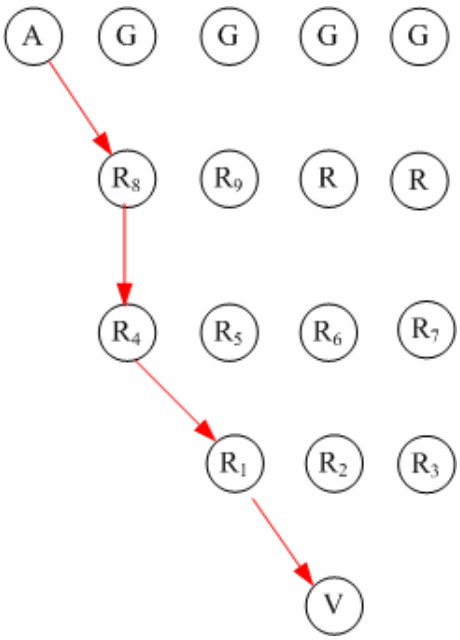

Figure 3.8 Random Network Topology in DoS cases (good links omitted)

3.4 Summary

In this chapter, we proposed a hybrid traceback-filtering (HTF) mechanism that leverages on the controlled flooding method to identify the attack links. Then by preferentially dropping packets coming from attack links in the routers that are on the attack path, HTF effectively mitigates the attack while protecting the legitimate traffic from collateral damage. In the HTF mechanism, the packet filtering function at the routers is recursively activated hop-by-hop to progressively save the bandwidth consumption in upstream links. We performed simulations using ns2 to evaluate the performance of our HTF mechanism under DoS attack. Simulation results show that our mechanism is able to drop 30% of bad packets at the start of the process and up to 100% when the traceback is completed. Our simulation also shows that the false positive ratio ranges from 15% to 0% at the beginning and end of the traceback-filtering process respectively. As a result, our mechanism can improve the throughput of legitimate packets. Theoretical analysis shows that our mechanism can save link bandwidth consumption. For instances, when the generation rate of bad packets is 500 times that of the good packets, the link bandwidth consumption can be saved by more than 56 percent.

CHAPTER 4
HTF-DDOS VERSION

DDoS attacks aim to deny legitimate users from their access to some services as well as DoS attacks do. However, the scale of a DDoS attack is much larger than a DoS attack. In a DoS attack, only one attacker is involved in while in a DDoS attack, numerous hosts are coordinated by the real attacker to carry out an attack. Thanks to its distributed characteristic, DDoS attacks multiply the effectiveness of an attack. Also, DDoS attacks are harder to be traced as DDoS attacks enable a diversity of sources. All of these accelerate DDoS attacks substitute DoS attacks to be the main trend in DoS attacker community. And it is necessary for us to extend the HTF mechanism to DDoS cases.

To extend the HTF mechanism to DDoS cases, we entitle the victim with more responsibility. Accordingly, the three functionality of the victim (controlled flooding functionality, attack identification functionality and filtering signal generation functionality) become more complicated. We will introduce them respectively in section 4.2 and 4.3. Meanwhile, as we keep the functionality of the routers unchanged, we will not repeat them in this chapter (please refer to chapter 2 for the details of the functionality in the routers).

This chapter is organized as follows: Section 4.1 briefly introduces the improvement of the functionality of the victim in the HTF-DDoS mechanism. Section 4.2 introduces the functionality of the victim respectively. In section 4.3 both performance evaluations

through extensive simulations in ns-2 and theoretically analysis are described. Finally in section 4.4 we summarize the results and conclude this chapter.

4.1 A Brief Introduction of the Functionality of the Victim

- **Controlled Flooding Functionality.** The controlled flooding functionality requires the victim to have three basic abilities which are similar to those in the HTF-DoS mechanism. However, some difference exists. Firstly, in the HTF-DoS mechanism, the termination of the whole process depends on the how the attack stream is restrained. Only when the attack stream is restrained into an acceptable range, the HTF-DoS mechanism is stopped. However, in the HTF-DDoS mechanism, the termination is also determined by a new tuning parameter T which affects the highest level of the Internet that the HTF-DDoS mechanism can reach. Details of the tuning parameter T is given in the section 4.2. Secondly, a bad link marking algorithm is added to the HTF-DDoS mechanism which helps to specify a link to test in each step.

- **Attack Link Identification Functionality.** The attack link identification functionality means the ability for a victim to identify attack links from good ones. The difference between the HTF-DoS mechanism and the HTF-DDoS mechanism lies on the introduction of a tuning parameter T which adjusts the sensitivity of the victim in identifying the bad links.

- **Filtering Signal Generation Functionality.** The filtering signal generation functionality is the ability for a victim to control the filtering function and link testing function in routers. Different from the HTF-DoS mechanism, four kinds of signal packets are added to the HTF-DDoS mechanism which is ICMP *filter-on*, ICMP *filter-off*, ICMP *linktesting-on* and ICMP *linktesting-off*.

4.2 The Functionality of the Victim

The main functionality of victim includes controlled flooding functionality, attack link identification functionality and filtering signal generation functionality.

- **Controlled Flooding Functionality.** Similar to the HTF-DoS mechanism, in the HTF-DDoS mechanism, starting from router closet to the victim, we apply a brief burst of load to each link which is attached to the nearest router, and then recursively apply the link loading to all the upstream links. Three functions are required in the victim to conduct successful controlled flooding. Firstly, it requires that the victim be able to start the controlled flooding procedure correctly which is the same to the HTF-DoS mechanism. Secondly, it requires the victim find out voluntary cooperators in the Internet that can assist in flooding specific link effectively which is mostly similar to the HTF-DoS mechanism except that in DDoS attacks, much more voluntary cooperators are needed. It is not easy to get so many cooperators in reality, however, for the sake of research, we assume that the victim can always get the cooperate hosts on its needs. Thirdly, the victim should be able to specify a link to test in each step which is similar to that in the HTF-DoS mechanism. Additionally, we propose a bad link marking algorithm (illustrated in the following figure) which identifies the unsuspicious links from the suspicious ones and marks them respectively.

In the marking algorithm, we assume the maximum identification value of all the links in the Internet is M, the highest level of the Internet is H and the union of all the unsuspicious links is U. Suppose the identification number of a link is id, the level of a link in the Internet is L_{id}, and the near end of the link is N_{id} and the far end of the link is F_{id}. All the parameters are listed in Table 4.1 and Figure 4.1 illustrates the marking algorithm briefly.

Table 4.1 Parameters in Marking Algorithm

Parameters	Meaning of the parameters
M	the maximum identification value of all the links on the Internet
H	the highest level of the Internet
U	the union of all unsuspicious links
id	the identification number of a link
L_{id}	the level of a link
N_{id}	the near end of a link
F_{id}	the far end of a link

In the DDoS-HTF mechanism, we adopt a Breadth-First Traversal algorithm [41] for link searching. The process of selecting a suspicious link to be tested in each step is actually a tree traversal process. By using the breadth-first traversal algorithm, we prefer to choose the links which are in the same level as the previously tested link I, unless all the suspicious links in this level are already tested, we select the link to be tested next from the suspicious links in the upstream levels.

Lastly, the victim should stop the whole process timely. There are two cases: firstly, when the attack is restrained into an acceptable range which is the same as that in the HTF-DoS mechanism. Secondly, the victim terminates the process when the HTF-DDoS mechanism cannot be propagated up any more, which is a great difference compared to DoS cases. A tuning parameter T is introduced to the HTF-DDoS mechanism which affects the highest level that the HTF-DDoS mechanism can reach in the Internet, therefore the HTF-DDoS mechanism may become ineffective before the real attacker is traced or the attack stream is constrained into an acceptable range. Besides the two cases, further controlled flooding should be done.

```
For each link id in the Internet
 If link id is a good one
    Put id into U
     While (id < M&& L_id<H)
          For each link id' in level L_id+1
            If N_id' = F_id
                  Put id' into U
                  id =id'
                  L_id=L_id+1
```

Figure 4.1 Marking Algorithm

- **Attack Link Identification Functionality.** This functionality requires the victim to observe the effect caused by controlled flooding. By observing changes in the attack streams, the victim tells which links are bad ones. However, there is no quantitative specification on how much change could be a sign of bad stream perturbing so far. Hence we introduce a tuning parameter T to solve this problem. We state that only if the intensity of the attack stream after link testing is less than the value of T out of the intensity before link testing, the link under test is identified as an attack one. Otherwise, the link under test is identified as a good one. In one word, the victim takes the links which contribute more than 1/T of the whole attack traffic as bad links.

The introduction of the tuning parameter T brings two particulars to the HTF-DDoS mechanism. Firstly, the value of the tuning parameter T decides the sensitivity of the victim in attack link identification. Specifically, a higher value of the tuning parameter T enables a higher sensitivity while a smaller value of the tuning parameter T brings a lower sensitivity. When T is given a high value, according to the definition of the tuning parameter T, the victim can detect links contributing more that 1/T percent of the attack traffic and which is a small value accordingly. That means the victim can identify the attack links sensitively. On the contrary, when T is given a low value, the sensitivity of the victim is low.

Secondly, the value of the tuning parameter T indirectly affects the highest level that the HTF-DDoS mechanism can reach in Internet. The reason is as follows: the attack stream is gradually integrated while flowing to the victim, and seen from the victim, a higher-level attack link contribute a smaller portion to the whole attack stream. Therefore, when T is given a smaller value, the victim is less sensitive in the attack traffic change and it is less possible to identify the higher-level bad links. Inversely, when we have a higher value of the tuning parameter T, the victim is more capable to detect the higher-level bad links and the HTF-DDoS mechanism can be propagated up to a higher-level of the Internet. We prove the two statements by simulations which are given in Section 4.3.

- **Filtering Signal Generation Functionality.** The function of filtering signal generation is to control filtering and link testing in different routers. In the HTF-DoS mechanism, the correspondent functionality introduces two kinds of signaling packets: ICMP filter-on and ICMP filter-off. In the HTF-DoS mechanism, the victim generates two more kinds of signaling packets: ICMP linktesting-on and ICMP linktesting-off, at the same time we also improve the function of ICMP filter-on and ICMP filter-off.

The ICMP filter-on packet activates the filtering function in the destination router while the ICMP filter-off packet deactivates it. Different from the ICMP filter-off in the HTF-DoS mechanism which turns off the filtering function entirely, the ICMP filter-off packets in the HTF-DDoS mechanism turns off it partially. It only deactivates the router in dropping packets from specific links. In the HTF-DoS mechanism, on the receipt of ICMP filter-off, the router passes all the packets through whatever their incoming links. However, in the HTF-DDoS mechanism, the routers only lets those from the link specified in the ICMP filter-off packet go and keeps dropping the rest. The ICMP linktesting-on packet activates the link testing function in the destination router while the ICMP linktesting-off packet deactivates it.

After a link R1-R2 (R1 and R2 are the end routers of R1-R2 link) is identified as being on the attack path, the victim generates an ICMP filter-on packet for the nearer end of R1-R2 link seen from the victim, say R2. If the R1-R2 link is a right branch in Internet, the victim also generates an ICMP filter-off packet for the father node of R2 which is R3. The ICMP filter-off packet stops R3 router from dropping packets coming from R2-R3. If the attack link is identified as not being on the attack path, two cases come up. If the R1-R2 is a right branch, the victim generates an ICMP linktesting-off packet for the R2 and generates an ICMP filter-on packet to R3. Otherwise the victim generates an ICMP linktesting-on packet to the R2 and the right sibling branch of R1-R2 is to be tested next.

An ICMP filter-on packet includes two items of information: the IP addresses of the routers at the ends of an attack link, and the Base Value (BV) as a tuning parameter based on which routers adjust their filtering probability which is the same to that in the HTF-DoS mechanism.

An ICMP filter-off packet tells the destination which incoming link should not be rate limited by then. Hence an ICMP filter-off packet includes the IP addresses of the end routers of this link.

An ICMP linktesting-on packet includes the IP addresses of the end routers of the link which is chosen to test next and an ICMP linktesting-off packet includes the IP addresses of the end routers of the link which is currently tested.

Suppose the near end router of a link Rn-Rf is Rn and the far end is Rf. Then the algorithm is as the following:

```
 For the link tested in the current step
If the link is a bad one
     If the link is a right one
        Send an ICMP filter-on packet for Rn aimed at Rf
         Send an ICMP filter-off packet for Rn/2 aimed at Rn
    Else
      Send an ICMP filter-on packet for Rn aimed at Rf
Else if the link is a right one
   Send an ICMP linktesting-off packet for Rn aimed at Rf
       Send an ICMP filter-off packet for Rn/2 aimed at Rn
     Else
      Send an ICMP filter-on packet for Rn aimed at Rf+1
```

Figure 4.2 Filtering Signal Generation

4.3 Performance Evaluation

In this section, we evaluate the performance of the HTF-DDoS mechanism under DDoS attack using the ns2 simulation software. First we explain the common configuration settings and attack model used in our simulation. Then we present the results of our simulation by some performance metrics which is the same as that in the HTF-DoS mechanism.

4.3.1 Common Configuration

In the HTF-DDoS simulation, we use the same network model as the one used in HTF-DoS simulation. We simulate a larger-scale attack by increasing the number of the attackers which are selected from 256 hosts randomly and are also location-integrated. The rest hosts are assumed to be legitimate ones. The basic configuration parameters of all the legitimate hosts, attacks, routers and links are the same as that used in DoS simulations.

4.3.2 Performance Metrics

To evaluate the performance of the HTF-DDoS mechanism, we use the same performance metrics introduced in Chapter 3: the Good Packets Percentage (GPP), the Bad Packets Percentage (BPP), the Good Packets Dropping Percentage (GPDP) and the Bad Packets Dropping Percentage (BPDP). We also evaluate the performance of bandwidth consumption theoretically. In the rest of the section we present our results for each metric respectively.

4.3.3 Simulation Results and Analysis

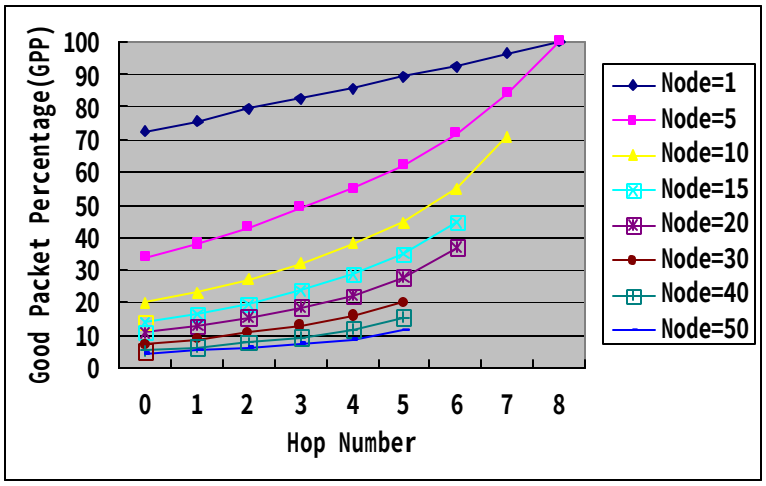

Figure 4.3 GPP with T equal to 0.85

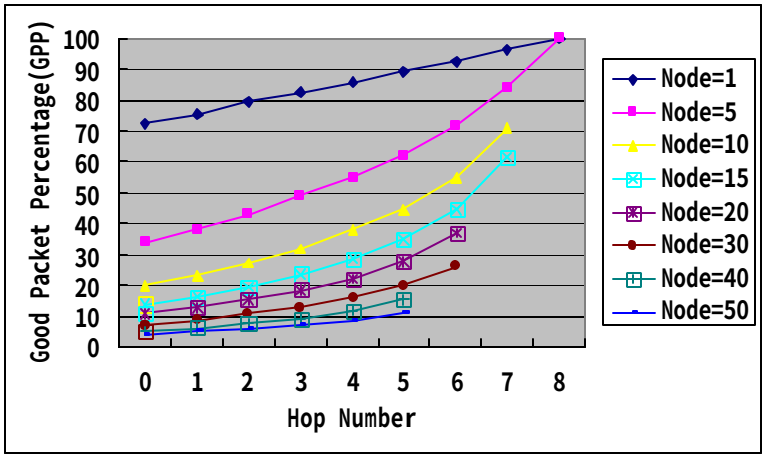

Figure 4.4 GPP with T equal to 0.9

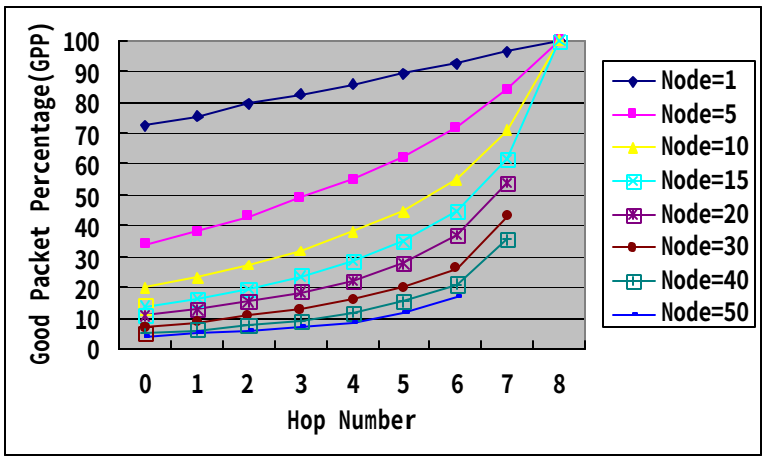

Figure 4.5 GPP with T equal to 0.95

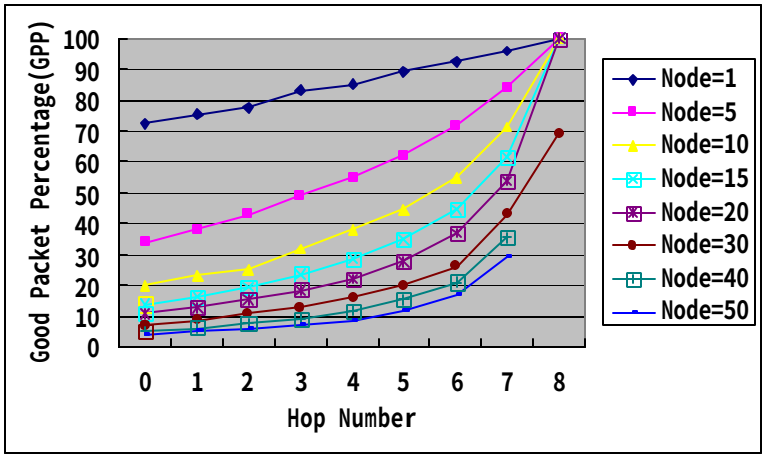

Figure 4.6 GPP with T equal to 0.98

Figure 4.3 to Figure 4.6 show the GPP as a function of the hop number respectively. The eight curves in each figure above reflect the different scale of DDoS attacks and correspond to one-attacker case, five-attacker case and ten-attacker case etc respectively. As shown in each figure, when the hop number increases, GPP increases gradually. For instance, as we can see from Figure 4.5, when the number of the attackers is 10 and the value of the tuning parameter is 0.95, GPP increases from 20% to 100%, which represents an improvement of 400%. The higher GPP reflects the higher effectiveness of HTF. In addition, we can see from each figure that when the number of the attackers increases, the highest-level that HTF-DDoS can reach increases. For instance, in figure 4.5 when there are less than 15 attackers, the HTF-DDoS mechanism can be propagated up to the top level of the internet topology, while when the number of attackers is between 20 and 40, the HTF-DDoS mechanism can only reach the second-top level of the internet topology. The observation is consistent to the value of T used in the simulation. In figure 4.5, the value of the tuning parameter T is 0.95 which means we can only detect the attack links which contribute at least twenty percent of the whole attack volume and those contributing less than twenty percent of the whole attack volume are neglected. When the number of the attackers is less than 15, all the top-level links being on the way of the attack stream contribute at least 15 percent of the attack stream; therefore, each of them can be detected by the HTF-DDoS mechanism. While when the number of the attackers is between 20 and 40, all the top-level bad links contribute less than the 20 percent of the whole attack volume, and the HTF-DDoS process can not reach the highest level. However, the process can be propagated up to the second-top level of the internet topology as each second-level attack links contribute more than twenty percent of the whole attack volume.

From the four figures-Figures 4.3 to Figure 4.6, we can see that when the attack scale is fixed, the HTF-DDoS mechanism is more effective when the value of T is higher. When the value of T increases from 0.85 to 0.95, the HTF-DDoS process can be propagated up to a higher and higher level. For instance, when the value is 0.85, we can implement the HTF-DDoS mechanism to the whole internet only when the number of the attackers ranges from 1 to 5, however, when the value of the tuning parameter is 0.98, the HTF-DDoS mechanism can reach top level of internet topology when number of attacker ranges from 1 to 20. In addition, when the number of the attackers is 30, the highest level that the HTF-DDoS mechanism can be propagated up to is fourth-top level, third-top level, second-top level and top-level when the value of T is 0.85, 0.9, 0.95 and 0.98 respectively. It can be explained by the mechanism itself. The bigger value of the tuning parameter T enables the HTF-DDoS mechanism to detect those less-contribution links which may not be detected given a smaller T value, as a result, the highest level can be reached increases.

Suppose the value of T is Vt and there are Ni links passed by the attack packets in ith-level of the internet topology. When the top level of internet topology is I, we can use the following formula to estimate the highest level H that HTF-DDoS can reach theoretically.

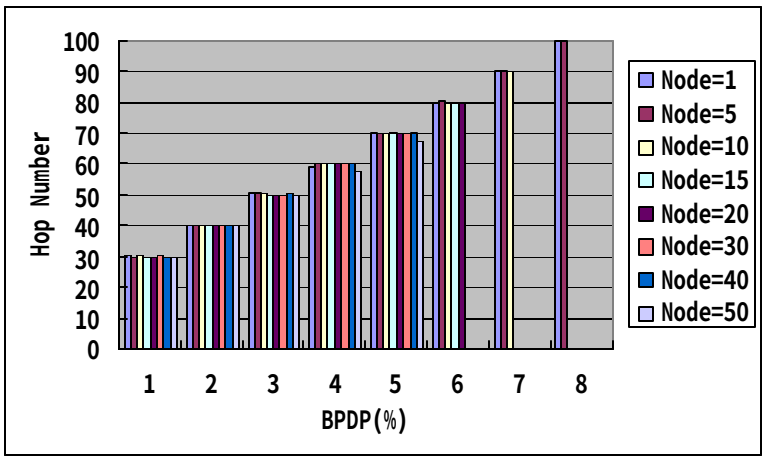

Figure 4.7 BPDP with T equal to 0.85

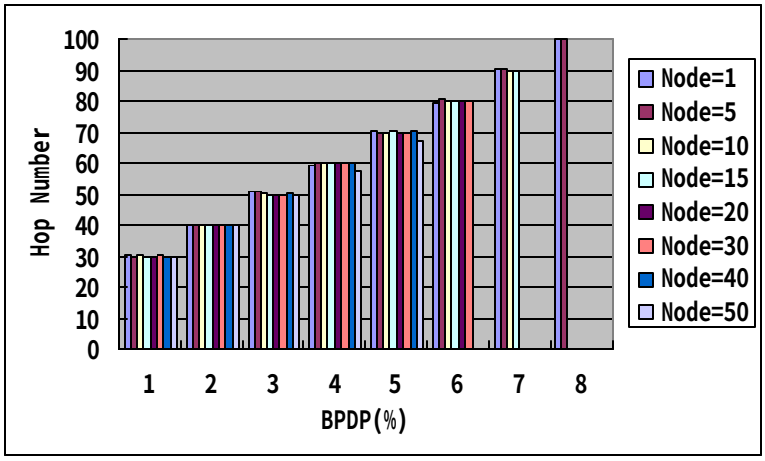

Figure 4.8 BPDP with T equal to 0.9

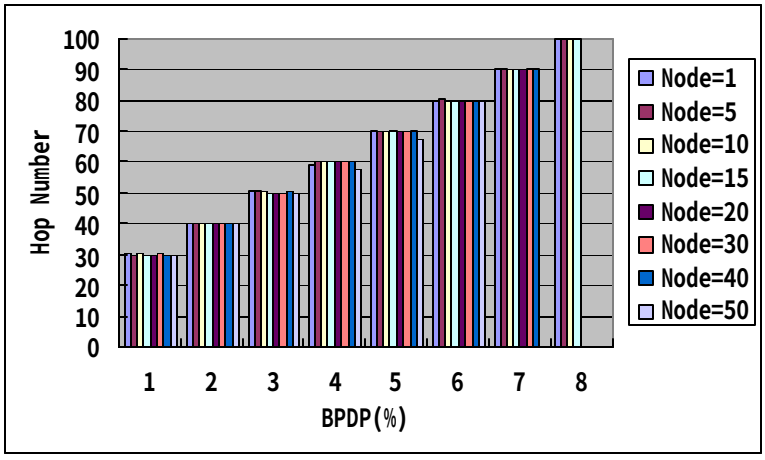

Figure 4.9 BPDP with T equal to 0.95

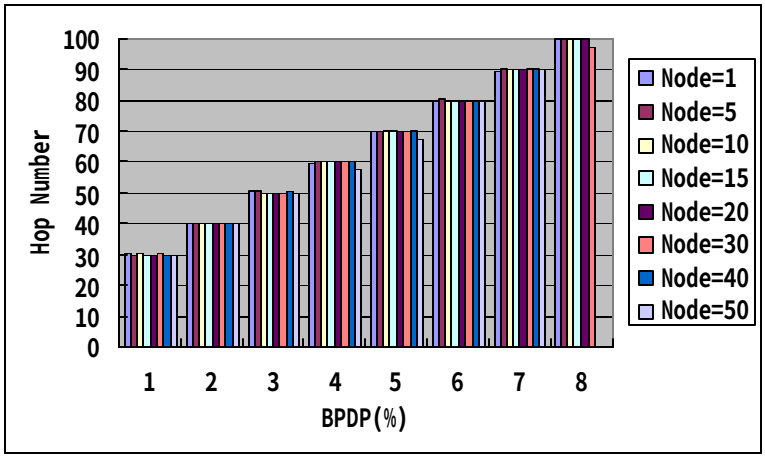

Figure 4.10 BPDP with T equal to 0.98

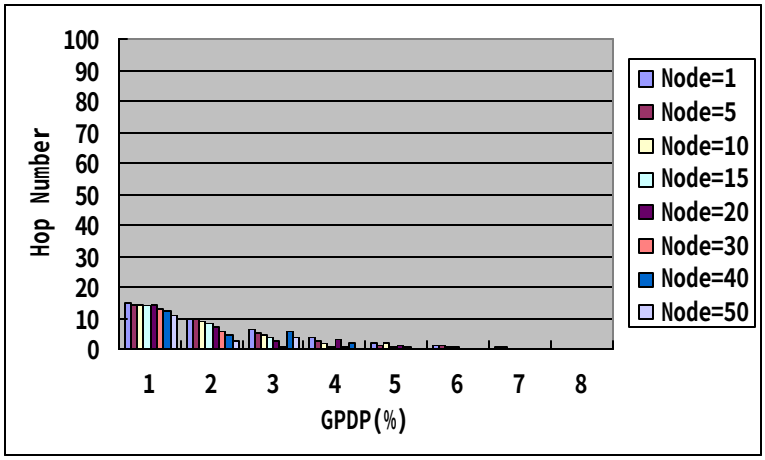

Figure 4.11 GPDP with T equal to 0.85

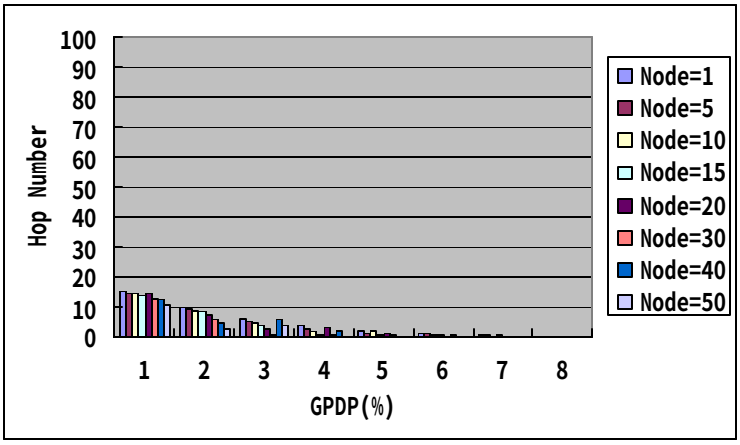

Figure 4.12 GPDP with T equal to 0.9

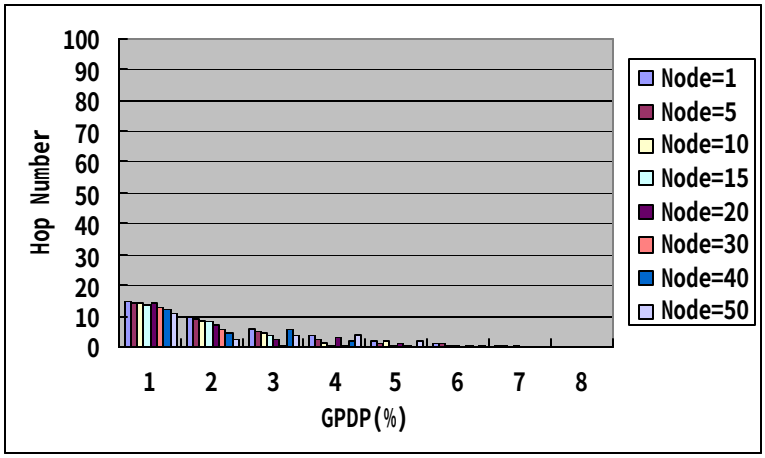

Figure 4.13 GPDP with T equal to 0.95

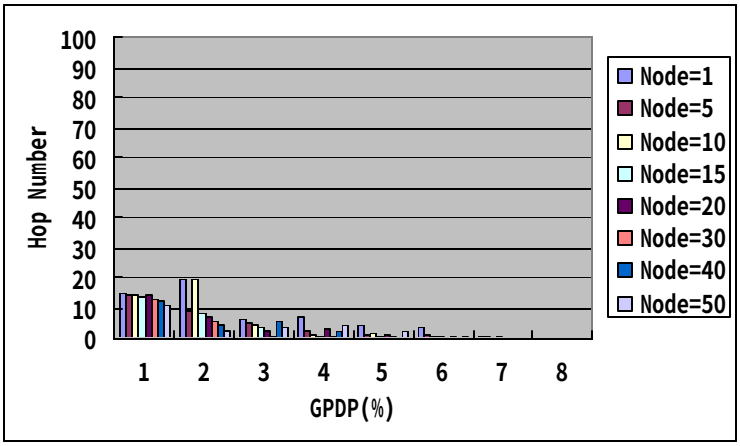

Figure 4.14 GPDP with T equal to 0.98

Figure 4.5 to Figure 4.12 show the BPDP and the GPDP as a function of the hop number respectively. As we can see from the above figures, whatever the value of the tuning parameter T is, BPDP and GPDP at a given hop are almost the same despite the number of attackers varies. For instance, when the first-level routers filter, the BPDPs and GPDPs are all about 30% and 15% respectively. That is because that the routers drop the bad packets with the same probability despite the number of attackers. In addition, we can see from Figure 4.5 to Figure 4.8 that when the hop number increases, BPDP increases. For

instance, BPDP is about 30% when first-level routers filter, and then gradually increases to 100% at the last hop. That is consistent to the dropping probability used in our simulations which ranges from 0.3 to 1. In addition, as we can see from Figure 4.9 to Figure 4.12, while the hop number increases, GPDP decreases instead. For instance, when the first-level routers filter, GPDP is about 15%; at the last step, the value decreases to about 0%. The reason for GPDPs decreasing with increasing dropping probability of a router is that the higher precision for a higher-level router to identify bad packets makes up the collateral damage greatly. Higher-level routers are more accurate in differentiating good packets from bad packets by their incoming links as the bad and good streams are not highly integrated. Even though the higher-level routers drop packets heavily, good packets are discarded less. Due to the imperfect precision of the simulation, the decreasing trend of GPDP is not absolute and it may be disturbed in a few steps, however, seen as a whole GPDP decreases as expected.

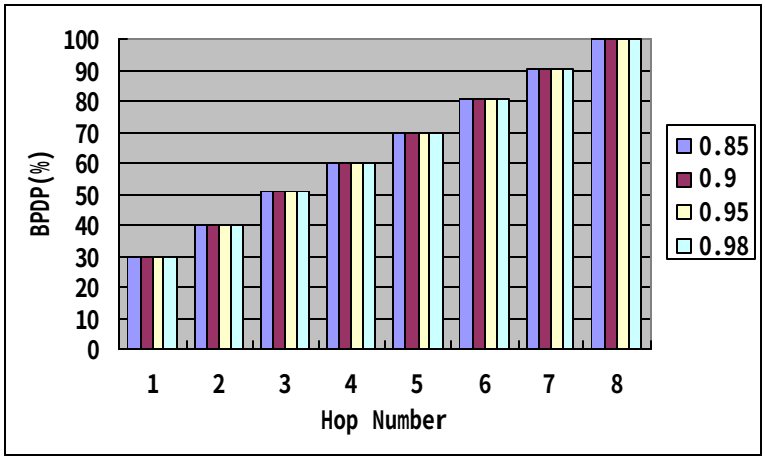

Figure 4.15 BPDP with five attackers

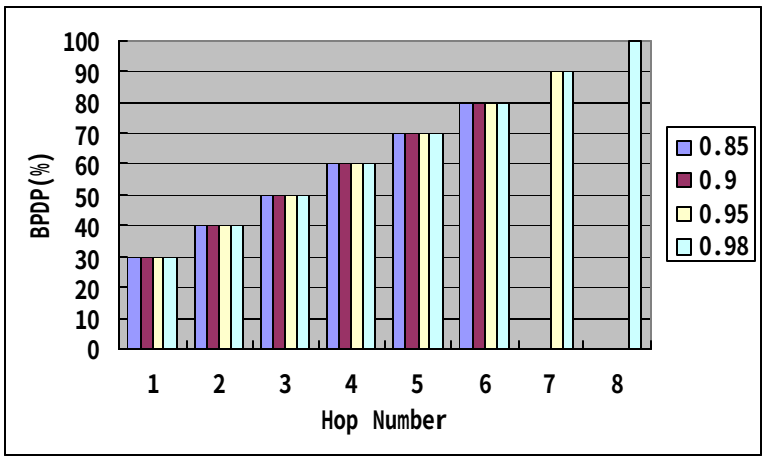

Figure 4.16 BPDP with twenty attackers

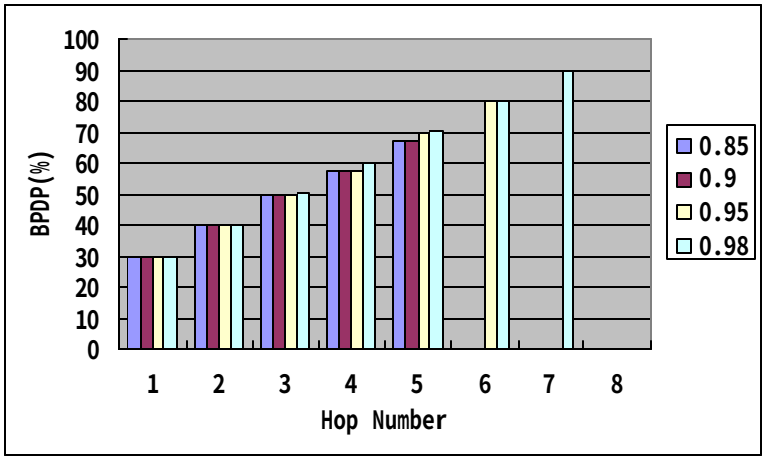

Figure 4.17 BPDP with fifty attackers

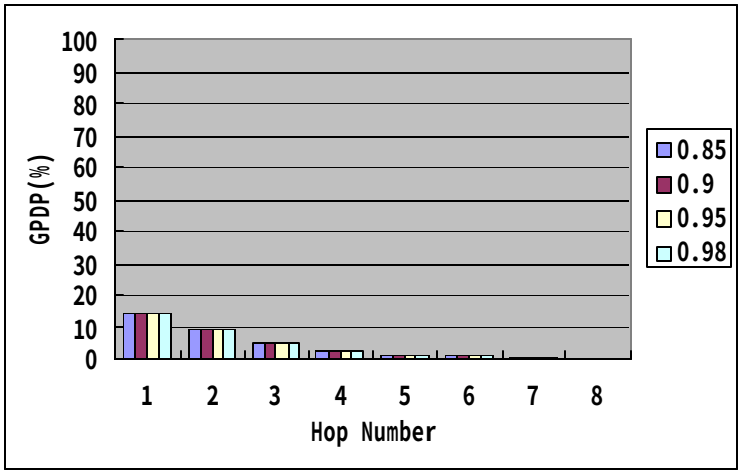

Figure 4.18 GPDP with five attackers

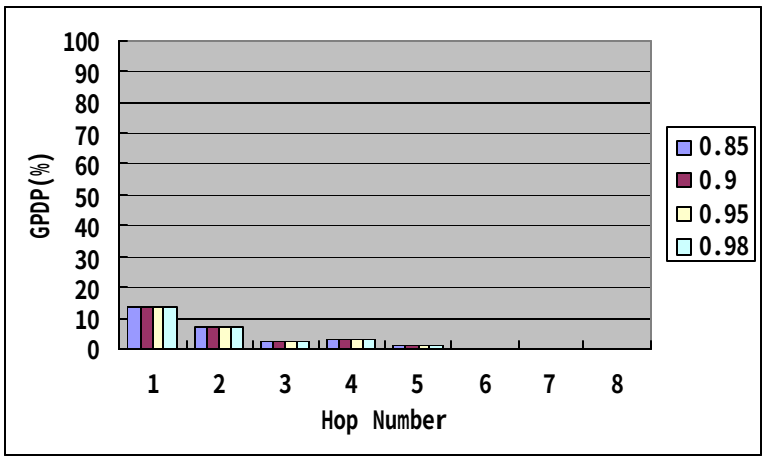

Figure 4.19 GPDP with twenty attackers

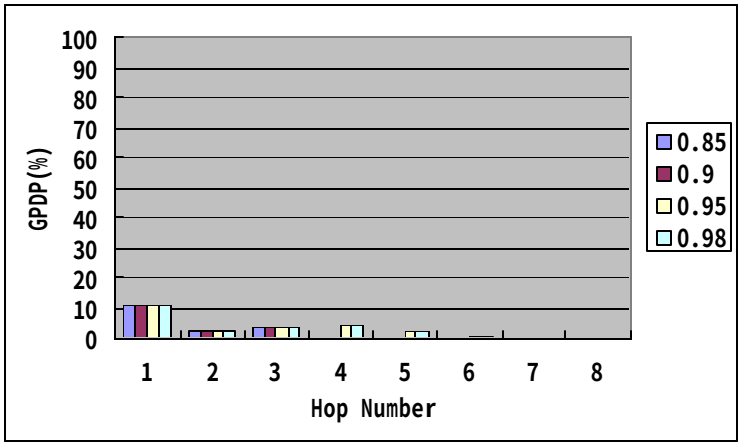

Figure 4.20 GPDP with fifty attackers

Figure 4.15 to 4.20 show the BPDP and the GPDP as a function of the hop number respectively. As we can see from the above figures, the BPDPs and GPDPs do not change with the value of T, nevertheless the highest level can be reached by the HTF-DDoS process changes. For instance, in figure 4.16 and figure 4.17 when the first-level routers filter, the BPDPs and GPDPs are all about 30% and 15% whatever the value of the tuning parameter T is 0.85 or 0.9 or 0.95 and 0.98. That is because routers drop packets with the same probability despite of the value of T. However, the highest level that the HTF-DDoS

mechanism changes when the value of T, which is theoretically analyzed in section 4.2 and we don not repeat again.

4.3.4 Theoretical Analysis

Our mechanism also saves upstream bandwidth consumption compared to those schemes where only the low level routers filter. We assume the router i hops away from the victim drops type A packets with the probability Pai, drops types B,C and D packets with zero probability respectively. Suppose the bad packet generation rate is Rb and the good packets generation rate is Rg. In an i-level binary tree topology where the number of attackers is Na and the number of non-attackers is Nn, the upstream link consumption C in our mechanism is as follows:

$$C = (N_a \bullet R_b + N_n \bullet R_g) + (Na \bullet R_b \bullet (1 - Pa_{I-1}) + N_n \bullet R_g) \bullet (I-1)$$

Let's consider the case that only the router one-hop away from the victim filters and drop all type A packets with probability P of 1, then the upstream link consumption C' is as the following:

$$C' = (Na \bullet Rb + Nn \bullet Rg) \bullet (I-1) + (Na \bullet Rb \bullet (1-P) + Nn \bullet Rg)$$

The values of C and C' according to different values of Rb are shown in Table 4.2.

Table 4.2 Values of C and C' in DDoS cases

	N_a=1	N_a=10	N_a=20	N_a=30	N_a=50
Consumption	2395	3214	4124	5034	6854
Consumption'	3095	10214	18124	26034	41854

As we can see from Table 4.2, the upstream bandwidth consumption is greatly saved in our mechanism by 29.23%, 217.8%, 339.48%, 417.16% and 510.65% when Na is 1, 10, 20, 30 and 50 respectively. And the bigger scale of the distributed attack, larger percentage of link bandwidth is saved. Hence we can say our scheme works even better when the distributed denial of service is more intense on the metric of bandwidth consumption.

4.3.5 Value Setting of the Tuning Parameter

As we can see from the figures and the tables listed in section 4.3.4, given a higher value of the tuning parameter T, the HTF-DDoS mechanism can be propagated to a higher level in internet topology and the output of good packets are better. However, it doesn't mean that the rule for us to set the value of T is the higher, the better. Sometimes, a smaller value is better than a bigger value. We should take into account the following two factors in setting a proper value: Firstly, the good packets are probably collateral dropped when the value of T is quite high. The big value of T makes the victim very sensitive in volume change of the attack streams, therefore a tiny change caused by the unstable status of internet or unstable bad packets transition rate can be detected and is wrongly looked as a sign of bad stream perturbing. Consequently, the link under test is identified as a bad link even it is a good one in fact and the good link will be rate-limited next. The bad consequence is that good packets coming from this link are dropped which increases false positive ratio of the HTF-DDoS mechanism. The performance of our mechanism is influenced. However, when the value of the tuning parameter T is not extremely high, the tiny change brought by the environment is neglected and the false positive ratio is kept low.

Secondly, the cost effectiveness is another key consideration when setting the value of T. We should take into account the cost of the implementation of link testing as well as the benefit we can get. As we have mentioned above, link testing itself is a kind of DDoS attack. It brings a short period of service-denial to the link under test, which affects internet availability. Therefore we try not to conduct it unless it is avoidable. Our aim is to release the attack effect, so once the attack stream is constrained into an acceptable range, we can terminate the process. It is not necessary to trace all the way to the attacker as the attack may be effectively controlled already before we reaching the highest level of the Internet. We should consider the balance of pains and gains carefully and strive to get better result with lower cost. It is always unnecessary to identify the links contributing a little and trace them to the top level. Instead, we should put the effort in stopping the main branches of an attack stream and terminate the HTF-DDoS process as soon as the attack is under control.

When we set a value to the tuning parameter T, many factors should be considered and try to find the one fitting our current need best.

4.3.6 Feasibility in Real Network

In this simulation, we use a binary-tree as our network topology as we did in HTF-DoS simulation. When we implement the HTF-DDoS mechanism in the real network, the basic idea of the HTF-DDoS mechanism need not be changed. However, some details of the design need to be changed. We will explain them respectively. Firstly, let us discuss the feasibility of our HTF-DDoS mechanism in the implementation in a random network topology. We assume figure 4.21 is a random network topology which is also a non-tree construction.

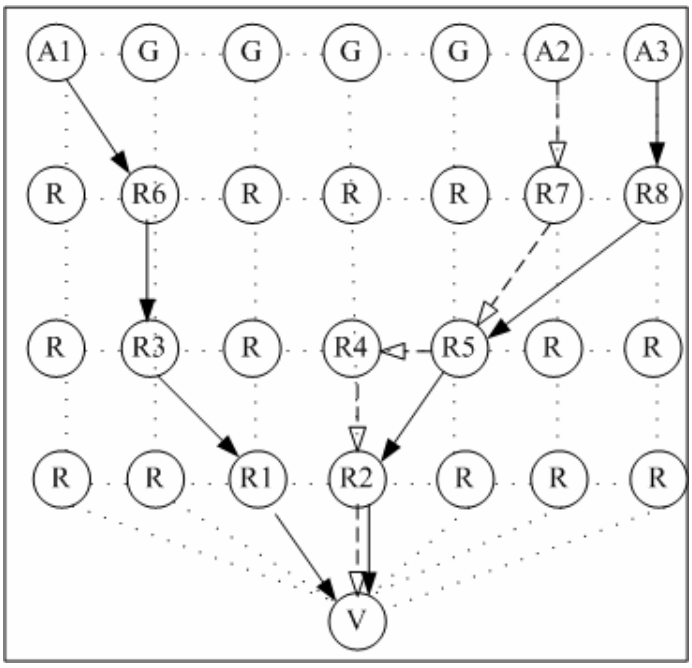

Figure 4.21 Random Network Topology in DDoS Cases

In this topology, we have three attack sources which are A1, A2 and A3 respectively. And the rest of the components are non-attackers and routers which are marked as G and R respectively. The arrow line represents the attacking path.

As we did in binary-tree situation, we will test the links hop-by-hop. Firstly, we test the links in level 1 and identify R1-V and R2-V as attack links, then we test links in level 2 and identify that R3-R1, R4-R2 and R5-R2 are bad links. Consequently, we test the links in level 3 and we find out that R5-R4 is an attack link and R5 is already been identified as the end of other attack link. The situation will not happen in a binary-tree or other tree topology while it tends to happen in non-tree topology. In this situation, we will not filter the packets coming from R4-R5 immediately as we can get the same effect by filtering the packets coming from R7-R5 and R8-R5 in next step. Hence, the route that we propagate the filtering function is shown as figure 4.22 which is essentially a tree-

topology. According to the explanation above, we can see it clearly that our HTF-DDoS mechanism which essentially propagate the mechanism in a tree-construction way works well in a random non-tree

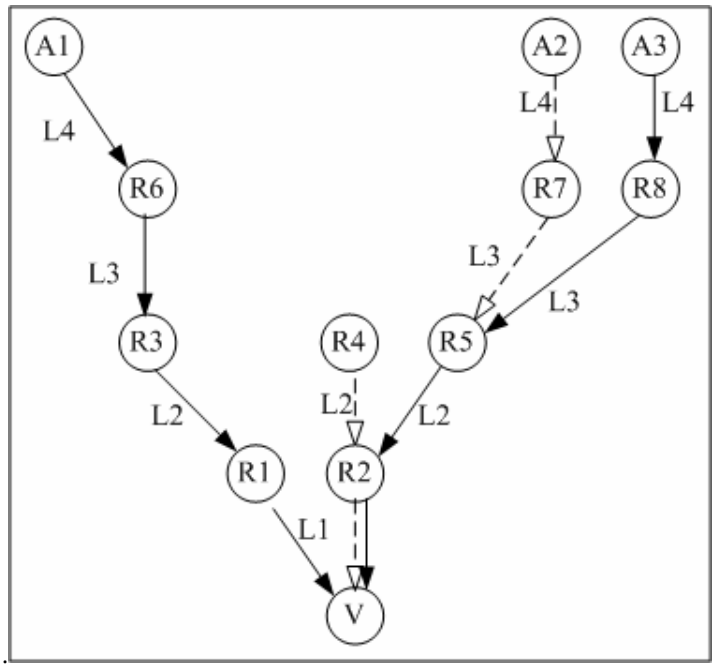

Figure 4.22 Random Network Topology in DDoS Cases (con't)

topology. However, to implement the HTF-DDoS mechanism in real DDoS scenario, we need change the filtering signal generation functionality slightly.

In our primary design of this functionality, we define the procedure of filtering signal generation based on the binary tree topology which is not feasible in non binary-tree construction. Hence we redefine it as follows. Suppose a link R1-R2's nearer end and further end are R2, R1 from the side of the victim respectively. The direct downstream router of R2 is R3 from the side of the victim. We revise the procedure as follows.

```
  For the link R1-R2 tested in the current step
If the link is a bad one
   If there are links whose nearer end is R2 remain untested
      Send an ICMP filter-on packet for R2_n aimed at R1_f
   Else
      Send an ICMP filter-on packet for R2 aimed at R1
      Send an ICMP filter-off packet for R3 aimed at R2
Else
   If there are links whose nearer end is R2 remain untested
     Send an ICMP linktesting-off packet for R2 aimed at R1
      Send an ICMP linktesting-on packet for R2 to test other links
  Else
       Send an ICMP filter-off packet for R3 aimed at R2
```

Figure 4.23 Revised Signal Generation Procedure

As we have already discussed in chapter 3, link testing itself is a kind of DoS attacks. When we apply it in DDoS scenarios, the problem becomes even more serious. In DDoS scenarios, there are more than one attacker, and hence, the number of suspicious links that we need to identify one by one is great. The time we may spend in identifying the bad links is expected to be much longer, which results in a more serious damage brought by link testing. However, as we have discussed in chapter 3, there still exist the feasibility of the HTF-DDoS mechanism. Firstly, if the attack is expected to last even longer than the whole process time, the implementation of the HTF-DDoS mechanism is acceptable in a long run as the benefit is far beyond the damage. Our mechanism can improve the output of good packets greatly and also provide the information of the original attacking source which can help us catch the bad guy effectively. If the attack lasts for only a short time,

perhaps before we get the true source of the attacking stream, we need to consider the use of HTF-DDoS mechanism carefully as mentioned in chapter 3 (please refer to section 3.3). In addition, if the victim has enough knowledge to predict that the attacks are highly integrated instead of diversely distributed, we can reduce the scope of bad links searching. In this condition, the DDoS scenarios will not differ too much from the DoS scenarios. The feasibility of the HTF-DDoS mechanism is still up to our consideration.

4.4 Summary

In this chapter, we proposed an extended version of the HTF-DoS mechanism which is called the HTF-DDoS mechanism. Different from the HTF-DoS mechanism which is only practicable in DoS cases, the HTF-DDoS mechanism is feasible in DDoS cases.

In the HTF-DDoS mechanism, the routers have the same functionality as those in the HTF-DoS mechanism but the victim is entitled with more complex functionality. In this chapter, a tuning parameter T is introduced to the HTF-DDoS mechanism. T is aimed to make a quantitative specification on how much change could be a sign of bad stream perturbing and we specify that only when the intensity of the attack stream after link testing is less than the value of T out of the intensity before link testing, the link under test is identified being on the attack path. Otherwise, the link under test is identified as a good one. The value of the tuning parameter T not only affects the victim's sensitivity in bad link identification and also indirectly decides the highest level that the HTF-DDoS process can be propagated up to. By running extensive simulations on NS2, we have proved that a bigger value of T enables the victim be more sensitive in attack link identification and enables the HTF-DDoS process reach a higher level of the internet topology.

By extensive simulations on NS2, we also evaluate the performance of the HTF-DDoS mechanism. Simulation results show that our mechanism is able to drop 30% of bad packets at the start of the process and up to 100% when the traceback is completed whatever the scale of DDoS attacks and the value of T are. Our simulation also shows that the false positive ratio ranges from 15% to 0% at the beginning and end of the traceback-filtering process respectively. As a result, our mechanism can improve the throughput of legitimate packets greatly. Theoretical analysis shows that our mechanism can save link bandwidth consumption. For instance, when the number of attacker is 50, the link bandwidth consumption can be saved by more than 500 percent. And the link bandwidth can be more saved when the distributed denial of service is more intense.

CHAPTER 5
CONCLUSIONS

DoS/DDoS attacks aim to deny or limit the legitimate users' access to a certain host or network. Based on the different objective of a DoS/DDoS attack, we can classify the attacks into two categories, namely flood attacks and logic or software attacks. In flood attacks, an Internet host is overwhelmed by a continuous flood of traffic designed to consume resources at the targeted server (CPU cycles and memory) and/or in the network (bandwidth and packet buffers). These attacks result in degraded service or a complete site shutdown. While in software or logic attacks, a small number of malformed packets are designed to exploit known software bugs on the target system. These attacks are relatively easy to counter either through the installation of software patches that eliminate the vulnerabilities or by adding specialized firewall rules to filter out malformed packets before they reach the target system. These attacks can cost the target a great deal of time and money. For instance, Yahoo, Buy.com and many other large commercial sites has been the target of DDoS attacks and was out of service for several hours. Due to the grave threat to internet security that DoS/DDoS attacks have posed, many security efforts aim at exploiting a certain feature of current attacks to prevent them or to constrain their effect have been made recently. Generally the countermeasures are categorized into four folds: Prevention, Detection, Filtering and Traceback. Although those approaches can effectively defend against DoS/DDoS attacks to some extent, few address the issue of minimizing false positive ratio and link bandwidth consumption when the DoS/DDoS attack is raging on.

5.1 Contribution

In this thesis, we propose a Hybrid Traceback-Filtering (HTF)-DoS mechanism to use traceback to enhance packet filtering. In this scheme, the packet-dropping function at routers is recursively activated hop by hop using traceback. In so doing, the bandwidth consumption of upstream links is reduced as compared to other proposals where only routers closest to the victim perform filtering. In HTF-DoS, we use controlled flooding to reconstruct the attack path, and then filter packets only at the routers which are on the attack path. This minimizes the false positive ratio caused by imprecise filtering. To minimize the reaction delay against DoS attacks, packet-filtering and route-reconstructing are conducted simultaneously. Simulation results show that our mechanism is able to drop 30% of bad packets at the start of the process and up to 100% when the traceback is completed. Our simulation also shows that false positive ratio ranges from 15% to 0% at the beginning and end of the traceback-filtering process respectively. As a result, our mechanism can improve the throughput of legitimate packets. Theoretical analysis shows that our mechanism can save link bandwidth consumption. For instance, when the generation rate of bad packets is 500 times that of the good packets, the link bandwidth consumption can be saved by more than 56 percent.

And to release the limitation of the HTF-DoS mechanism which is only practical in DoS cases, we extend the HTF-DoS mechanism to DDoS cases and the new mechanism is named HTF-DDoS. We entitle the victim in DDoS cases with more responsibility than the one in DoS cases and keep the routers' functionality unchanged. By extensive simulation results and theoretical analysis, we show that HTF-DDoS mechanism is effective in bad packet dropping, minimizing false positive ratio, improving throughput of good packets and saving link bandwidth consumption in DDoS cases as well as the HTF-DoS mechanism does in DoS cases. Simulation results show that HTF-DDoS is able to drop 30% of bad packets at the start of the process and up to 100% when the traceback is

completed whatever the scale of the DDoS attacks and the value of the tuning parameter. Our simulation also shows that the false positive ratio ranges from 15% to 0% at the beginning and end of the traceback-filtering process respectively. As a result, our mechanism can improve the throughput of legitimate packets greatly. Theoretical analysis shows that our mechanism can save lin k bandwidth consumption. For instance, when the number of attacker is 50, the link bandwidth consumption can be saved by more than 500 percent. And the link bandwidth can be more saved when the distributed denial of service is more intense.

5.2 Future Work

In our hybrid mechanism, controlled flooding is chosen as our traceback method that assists in identifying attack links. As we have mentioned in section 3, controlled flooding has many advantages over other traceback approaches, however, a great drawback of controlled flooding method is that itself is a kind of DoS attacks. When flooding the links, even if the abrupt of stream is of a tiny period, there are still a lot of good packets will be affected by the artificial congestion. Fortunately, our mechanism succeeds in minimizing the collateral damage on the good streams that may make up this drawback to some extent. There is a tradeoff between the benefits we may get from the HTF filtering and the bad effects brought by the HTF traceback. To strengthen the benefits brought by our hybrid mechanism, in the future we may focus on how to improve the Attack Path Identification Functionality of the victim. If the victim is able to identify a bad link within the minimum time, the period of artificial flooding can be minimized; therefore, less packets are influenced by the controlled flooding itself.

More effectively filtering rules are expected in future. For instance, the setting of base values for each router should be considered more carefully. In our mechanism, the base value follows the simply linear function and more reliable exponential functions should

be employed to find out the ideal setting of base values.

REFERENCES

[1] Understanding Denial-of-Service Attacks. [http://www.us-cert.gov/cas/tips/ST04-015.html]

[2] Yahoo/Amazon/CNN DDoS Attack. [http://www.nta-monitor.com/newrisks/feb2000/yahoo. htm].

[3] Results of the Distributed-Systems Intruder Tools Workshop. [http://www.cert.org/reports/dsit_workshop-final.html].

[4] What is DDoS and How It is Hurting the Internet, E-Commerce and Business. [http://www.cs3-inc.com/pk_whatisddos.html].

[5] P.Ferguson, D.Senie. RFC 2827: Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source IP Address Spoofing, May 2000.

[6] Types of DDoS Attacks. [http://www.anml.iu.edu/ddos/types.html].

[7] TCP SYN Flooding and IP Spoofing Attacks. [http://www.cert.org/advisories/CA-1996-21.html].

[8] Smurf IP Denial-of-Service Attacks. [http://www.cert.org/advisories/CA-1998-01.html].

[9] UDP Port Denial-of-Service Attack. [http://www.cert.org/advisories/CA-1996-01. html].

[10] Ping of Death. [http://www.insecure.org/sploits/ping-o-death.html].

[11] Jon Postel. RFC 793: Transmission Control Protocol.

[12] SYN Cookies. [http://www.liquifried.com/docs/security/scookies.html].

[13] SYN Cookies. [http://cr.yp.to/syncookies.html].

[14] Christoph L. Schuba, Ivan V. Krsul, Markus G. Kuhn, Eugene H. Spafford, Aurobindo Sundaram, Diego Zamboni. Analysis of a Denial of Service Attack on TCP. In Proc. IEEE Symposium on Security and Privacy' 97, pages 208—223, May 1997.

[15] Jon Postel . RFC 864: Character Generator Protocol.

[16] Linux/Unix Commander: Echo. [http://linux.about.com/library/cmd/blcmdl1_echo.htm].

[17] Jon Postel .RFC 792: Internet Control Message Protocol.

[18] Ping of Death. [http://compnetworking.about.com/library/glossary/bldef-pingofdeath.htm]

[19] Distributed Denial of Service Attacks and the Zombie Ant Effect. [http://www.computer.org/itpro/cover_stories/mar_apr/perspectives_1.htm]

[20] C. Perkins. RFC 2002: IP Mobility Support.

[21] An Introduction to Intrusion Detection Systems and the Dragon IDS Suite. [http://www.intrusion-detection-system-group.co.uk/].

[22] Snort-The Open Source Network Intrusion Detection System. [http://www.snort.org/].

[23] Identification of Repeated Attacks Using Network Traffic Forensics. [http://www.isi.edu/~johnh/PAPERS/Hussain03c.html]

[24] Haining Wang and Danlu Zhang, Detecting SYN Flooding Attacks. In Proc IEEE InfoComm'2002, June 2002.

[25] R.Mahajan, S.Bellovin, S.Floyd, Controlling High Bandwidth Aggregates in the Network. In Proc ACM SIGCOMM Computer Communication Review'2002, V.32 N.3, July 2002.

[26] Minho Sung, Jun Xu. IP Traceback-Based Intelligent Packet Filtering: A Novel Technique for Defending Against Internet DDoS attacks. In Proc IEEE Transactions on Parallel and Distributed Systems, 14(9):861-872, Sept. 2003.

[27] David K. Y. Yau, John C.S. Lui. Defending Against Distributed Denial of Service Attacks. In Proc the 3rd Symposium on Operating Systems Design and Implementation (OSDI'99), New Orleans, LA, February 1999..

[28] DoS Defenses. [http://infosecuritymag.techtarget.com/articles/september01/cover.shtml].

[29] Steve Bellovin et al. ICMP Traceback messages. IETF Internet Draft "draft-ietf-itrace-04.txt", Feb 2003.

[30] Henry C. J. Lee, Vrizlynn L. L. Thing, Yi Xu and Miao Ma. ICMP Traceback with Cumulative Path, An Efficient Solution for IP Traceback. 5th International Conference on Information and Communications Security (ICICS), China, October 2003, (Springer Lecture Notes in Computer Science, Vol. 2836, pp. 124-135, September 2003).

[31] S. Savage, D. Wetherall, A. Karlin, and T. Anderson. Practical Network Support for IP Traceback. In Proc. ACM SIGCOMM 2000, pages 295-306, Aug.2000..

[32] Hal Burch, Bill Cheswick. Tracing Anonymous Packets to Their Approximate Source. LISA XIV, December 3-8, 2000-New Orleans, LA. 2000.

[33] Strayer, W. Timothy. Single-Packet IP Traceback. In Proc IEEE/ACM Transactions on Networking. Dec. 2002.

[34] Allison Mankin, Dan Massey ,Chien-Lung Wu,S.Felix Wu and Lixia Zhang. On Design and Evaluation of "Intention-Driven" ICMP Traceback. In Proc of 10th International Conference on Computer Communications and Networks, 2001, PP.159-165..

[35] Dawn Xiao dong Song and Adrian Perrig. Advanced and Authenticated Marking Schemes for IP Traceback. In Proc IEEE INFOCOM, pages 878--886, 2001.

[36] Tao Peng, Christopher Leckie, and Kotagiri Ramamohanarao. Adjusted Probabilistic Packet Marking for IP Traceback. In Proc. Networking, pages 697-708, May 2002.

[37] D. Dean, M. Franklin and A. Stubblefield. An algebraic approach to IP traceback. In Proc Networks and Distributed System Security Symposium (NDSS)' 2001, Pages .3-12.

[38] M.T.Goodrich. Efficient packet marking for large-scale IP traceback. In Proc of the 9th ACM conference on computer and communications security, 2002, pages 117-126, Washington DC, U.S.A.

[39] Abraham Yaar, Adrian Perrig, Dawn Song. Pi: A Path Identification Mechanism to Defend Against DDoS Attacks. IEEE Symposium on Security and Privacy, pages 93-107, May, 2003..

[40] The Network Simulator-NS2. [www.isi.edu/nsnam/ns/].

[41] Breadth-First Traversal. [http://mathworld.wolfram.com/Breadth-FirstTraversal.html].