Defending Against Low-rate TCP Attack: Dynamic Detection and Protection

SUN Haibin

A Thesis Submitted in Partial Fulfilment
of the Requirements for the Degree of
Master of Philosophy
in
Computer Science and Engineering

Supervised by

Prof. John, C.S. Lui

June 2005


香港中文大學
大學
圖書館系統
18 NOV 2006
UNIVERSITY
LIBRARY SYSTEM
THE CHINESE UNIVERSITY OF HONG KONG

Abstract of thesis entitled:

Defending Against Low-rate TCP Attack: Dynamic Detection and Protection

Submitted by SUN Haibin

for the degree of Master of Philosophy

at The Chinese University of Hong Kong in June 2005

In this thesis, we consider a distributed mechanism to detect and to defend against the low-rate TCP attack. The low-rate TCP attack is a recently discovered attack. In essence, it is a periodic short burst that exploits the homogeneity of the minimum retransmission timeout (RTO) of TCP flows and forces all affected TCP flows to backoff and enter the retransmission timeout state. When these affected TCP flows timeout and retransmit their packets, the low-rate attack will again send a short burst to force these affected TCP flows to enter RTO again. Therefore these affected TCP flows may be entitled to zero or very low transmission bandwidth. This sort of attack is difficult to identify due to a large family of attack patterns. We propose a distributed detection mechanism to identify the low-rate attack. In particular, we use the "*dynamic time warping*" approach to robustly and accurately identify the existence of the low-rate attack. Once the attack is detected, we use a fair resource allocation mechanism to schedule all packets so that (1) the number of affected TCP flow is minimized and, (2) provide sufficient resource protection to those affected TCP flows. We also develop a fluid model which is analyzed and proved to be precise to describe the behavior of TCP flows under the fair resource allocation mechanism. Experiments are carried out to quantify the robustness and accuracy of the proposed

distributed detection mechanism. In particular, one can achieve a very low false positive/negative when compare to legitimate Internet traffic. Our experiments illustrate the efficiency of the defense mechanism across different attack patterns and network topologies.

低頻 TCP 拒絕服務攻擊的防禦：動態監測及保護

碩士研究生論文摘要

在本文中，我們考慮使用一種分佈式的機制來監測和對抗低頻 TCP 拒絕服務攻擊。低頻 TCP 拒絕服務攻擊是一種近期被發現的新型拒絕服務攻擊。它是一種周期性的突發數據流。它通過利用目前 TCP 協議最小超時重傳時間（RTO）的一致性，強迫所有受影響 TCP 數據流進入擁塞迴避狀態和超時重傳的等待狀態。黨 TCP 數據流超時結束，開始重傳時，該低頻拒絕服務攻擊將產生下一周期的突發數據流，導致 TCP 再次進入擁塞迴避狀態和超時重傳的等待狀態。通過連續作用，TCP 數據流將無法得到或得到極少的數據傳輸帶寬，從而被實施拒絕服務攻擊。

因爲其攻擊模式的不確定性，這種攻擊一般很難被發現。在本文中，我們提出了一種分佈式俄機制來監測鑒別該攻擊。事實上，我們通過使用動態時間規整算法（Dynamic Time Warping，DTW），可以可靠而準確地檢測該低頻攻擊。一旦發現低頻攻擊，我們使用一种公平的網絡資源分配機制，使（1）受低頻攻擊的 TCP 數據流減到最少，（2）受低頻攻擊的 TCP 數據流得到足夠的網絡資源保護。同時我們提出了一個可以精確描述、估計 TCP 行爲的數學模型，並用之從理論上估算出 TCP 數據流在我們的防禦措施保護下的行爲方式。大量的實驗證明我們提出的分佈式檢測機制具有極高可靠性和準確度，同時實驗表明，在各種各樣的網絡拓撲結構中，該防禦措施能夠高效地對抗各種不同模式的低頻拒絕服務攻擊。

Acknowledgement

I would like to express my deepest appreciation to my supervisor, Professor John C.S. Lui, who guided me through the past two years' of my research. He provided me the most priceless advice and support on both technical and editorial issues. Without his help, encouragement, and patience, my work in these two years would not be so much fruitful. Both my interest and confidence in doing research were greatly strengthened by his generous encouragement and positive evaluations. John showed to me what it means to be a great and benevolent professor. Besides research field, he also taught me a lot in my daily life and he also trained me to be a self-disciplined student.

I would to specially thank Professor David K.Y. Yau from Purdue University. His creative guidance and constructive criticism have greatly contributed to my research work. During the cooperation with David, I have learned how to be an excellent researcher in academic field. His smartness and energetic attitude gave me deep impression.

I would also like to thank Professor Moon Chuen Lee and Professor Man Hon Wong on my thesis committee. They actively participated my term talks and have put a great deal of effort on strengthening and improving my research and thesis.

And I also like to show my gratitude to the professors and staffs in the department of Computer Science and Engineering, CUHK, for the provision of the high quality research environment and varied assistances. The chairman of department, Prof. Kwong-Sak Leung,

and the head of graduate division, Prof. Kam-Wing Ng, have kindly provided much help on all kinds of fairs of my postgraduate study.

Here are my most sincere thanks to all of my friends and colleagues in our department, who are so trustable and dependable when there were technical troubles or other frustrations. They are Joe W.J. Jiang, Bin Fan, Yan Gao, Hui Wang, Moe T.Y. Wong, Shi Lu, Steven C.H. Hoi, Sharon Ping Yan, Catherine Lin Zhou, Dan Wei, J.S. Chen, Alan T.S. Ip, Paul K.H. Pun, Chi Hang Wong, Walty H.W. Yeung.... So many, many people have been there whenever I need help; many of them may not have their names listed here, but I thank them from the bottom of my heart. I am obliged to their warm friendship, care and encouragement. I did enjoy a lot being around with them.

This work is dedicated to my beloved parents and family for their persistent love and support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Summary

This chapter provides a general introduction about the content focused in this thesis.

The TCP protocol provides the reliable data delivery and simplifies application design and is being used in many network applications including file transfers, e-commerce, and web HTTP access. In general, designing a reliable protocol for many heterogeneous users sharing an unreliable network is challenging since it involves many subtle issues. For example, under severe network congestion, TCP requires sources to reduce their congestion window to one packet and wait for a retransmission timeout (RTO) before attempting to resend any packet. If there is further packet loss, the RTO is doubled after each subsequent loss. The purpose of using the RTO is to ensure that TCP sources will give the network sufficient time to recover from a network congestion event. In [8], authors recommend a lower bound of one second for its value in order to achieve near-optimal network throughput.

Although the use of RTO in the TCP protocol stack can reduce and relieve the event of network congestion, this feature can also be exploited by a malicious user to crate a denial-of-service attack.

Recently, authors in [30] present a form of *low-rate* TCP attack, in which an attacker periodically sends attack traffic to overflow a router's queue and cause packet loss. Due to the packet loss event, a legitimate (or well behaved) TCP source will then back off to recover from the congestion and retransmit only after one RTO. If the attacker congests the router again at the times of the TCP's retransmission, then little or no real data packet can get through the router. Hence, by synchronizing the attack period to the RTO duration, the attacker can essentially shut off most, if not all, legitimate TCP sources even though the average bandwidth of the attack traffic can be quite low. The form of low-rate attack raises serious concern because it is significantly more difficult to detect than more traditional brute force, flooding based DDoS attacks. Existing rate-limiting approaches [38, 66], for example, are designed to control aggressive or flooding-based attackers only.

In this thesis, we propose a distributed mechanism to detect against such low-rate TCP attacks. Because TCP is widely implemented and deployed, a proposal which requires changes to existing TCP protocol stack will incur a widespread modification of users' software and therefore this type of proposal may not be practical. This motivates us to consider a solution approach that can be implemented in a resilient routing infrastructure and benefit a large community of legitimate TCP users.

For detecting the low-rate attack, because an attacker's primary objective is to ensure the periodic overflow of a router's buffer, a basic signature of an attack traffic will then be intermittent short bursts of high rate traffic in between periods of little or no activity (characterized by, say, a periodic square wave). In practice, however, attack traffic can deviate from this basic attack signature for various reasons: distortion caused by queueing in intermediate routers, aggregation with background traffic (e.g., UDP traffic), an attacker's own attempt to inject "noise" into its traffic to escape detection, . . ., etc. Moreover, in a distributed attack, the traffic from individual at-

tack sources may not have the expected traffic characteristics, but the aggregation of such attack traffic does. Therefore, it is essential to develop detection algorithms that are both robust to traffic distortions and at the same time, computationally efficient to execute.

Once a low-rate attack has been detected, we seek to neutralize the effects of the attack traffic and minimize damage to legitimate users. The strategy is to rate limit and preferentially drop packets in an attack burst in order to reduce the loss of good user traffic. Note that the defense method has to provide a near perfect isolation in the midst of low-rate attack and at the same time, has to have the property of low implementation cost.

The contribution of our work is:

- Provide a formal model to describe and to generate a large family of low-rate attack traffic.
- Provide a distributed detection mechanism which uses the *"dynamic time warping"* (DTW) approach to robustly and efficiently identify low-rate attack. We will show that the proposed detection mechanism has a very low false positive/negative, when comparing a low-rate attack with a legitimate traffic.
- Provide a computationally efficient defense method to isolate legitimate traffic from the ill-behaved low-rate attack.
- Develop a fluid model which is analyzed and proved to be precise to describe the behavior of TCP flows under the defense mechanism.

The outline of this thesis is as follows. Background knowledge and related work is given in Chapter 2. In Chapter 3, we provide a formal mathematical model to describe and generate a large family of low-rate attack. In Chapter 4, we present the distributed mechanism of detecting the existence of the low-rate TCP attack. We also show the robustness and accuracy of the propose detection method when comparing an attack traffic with legitimate Internet traffic. In

Chapter 5, we present the defense mechanism and its properties. The fluid model of TCP flows under the defense mechanism is proposed in Chapter 6. Simulations of the model are also provided and results are analyzed. Experiments are presented in Chapter 7 to illustrate the effectiveness of the defense scheme and Chapter 8 concludes.

□ **End of chapter.**

Chapter 2

Background Study and Related Work

Summary

In this chapter, we provided the background knowledge about the Denial of Service Attack. We presented a classification of all the DoS/DDoS attacks as well as the different countermeasures to each kind of attack.

Network denial of service is a well recognized problem of importance and urgency (e.g., [21, 43]). In a simple Denial of Service Attack (DoS), a malicious user exploits the connectivity of the Internet to cripple the service offered by a victim site, sometimes by flooding the victim with many requests, sometimes by more tricky techniques. Such kind of attacks can also originate from multiple hosts, which is called Distributed Denial of Service attacks (DDoS).

Denial of service attacks caused significant damage to the Internet each year. In February of 2000, several high-visibility Internet e-commerce sites including Yahoo [51] and Amazon were brought down by a series of massive DoS attacks. And in January of 2001, Microsoft's name server was disabled by a similar assault. Next, in October 2002, 8 out of the 13 DNS root servers were incapacitated.... Many other sites have also been victims, ranging from smaller com-

mercial sites, to educational institutions and government organizations. Moore et al had provided insight into the prevalence of DoS activity on the Internet [40]. He used backscatter analysis and detected over 12,000 attacks against over 5000 targets during a period of three weeks.

Nowadays, DoS attack can be launched in many different ways, and some new kinds of Denial of Service attacks appeared as time went by. Research on denial of service attacks is primarily focused on attack detection and defense mechanisms. In this chapter we will review several the most important kinds of DoS attacks as well as some possible response mechanisms (As shown in Figure 2.1).

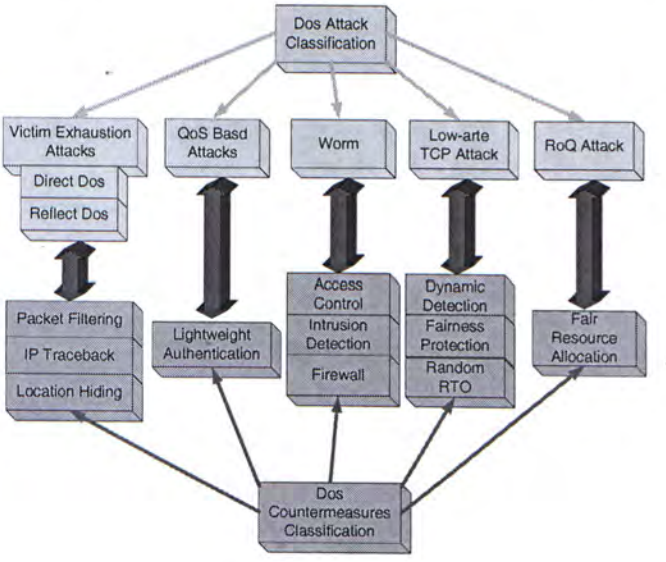


Figure 2.1: DoS Attacks and Countermeasures Classification.

2.1 Victim Exhaustion DoS/DDoS Attacks

One if the most common DoS attack is victim exhaustion attack, which purports to deny a victim providing or receiving normal service by over consuming the resources of victim (bandwidth, buffer...). In a successful attack, large volume of traffic or packets are involved.

Usually, two ways to launch such DoS attacks: one is direct attacks, the other is reflector attacks.

2.1.1 Direct DoS/DDoS Attacks

In a direct attack, an attacker arranges to send out a large number of attack packets directly towards a victim (Shown in Figure 2.2).

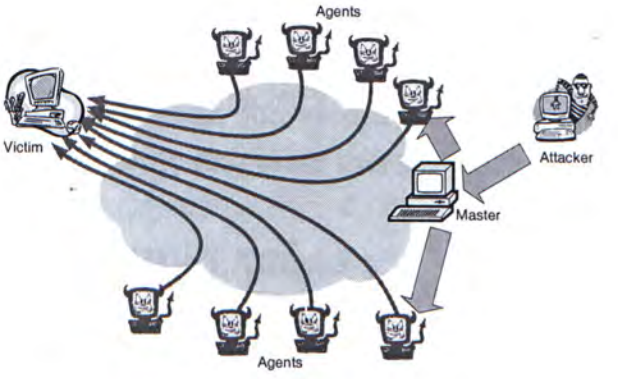


Figure 2.2: Direct DoS Attacks.

Usually, such attack is launched from multiple sources. An attacker first set up a DDoS attack network, consisting of one or more attacking hosts, a number of masters and a large number of agents (also called slaves or zombies). The attacking host is actually a compromised machine controlled by the real attacker to scan for vulnerable hosts and to implant attack master and slave programs, such as Trinoo [16], Tribe Flood Network 2000 [58] and Stacheldraht [12]. Each attacking host controls one or more masters, and each master in turn is connected to a group of slaves. With an attack network ready, the attacking hosts may launch a DDoS attack by issuing an attack command with the victim's address, attack duration, attack methods and other instructions to the masters; Each master, upon receiving the instructions, then pass the commands to its slaves for execution.

A detail process of building the attack network and launching attack is described in [17].

In a direct attack, attack packet can be TCP, UDP, ICMP, or a mixture of them. In the TCP case, one of the most commonly used methods is SYN Flood [1], in which a large number of TCP SYN packets are sent to the victim. By setting up more and more half-open connections, all the memories are consumed for the pending connections, thus preventing the victim from accepting new requests. Another type of TCP-base attack is to congest a victim's incoming link. In the UDP case, DoS attack can be launched between two UDP services [2]. And ICMP packets may also be used to achieve the same result. In this case, the victim usually responds with the corresponding ICMP reply and error messages.

2.1.2 Reflector DoS/DDoS Attacks

A reflector attack is a kind of indirect attack, in which intermediary nodes, known as reflectors, are used to launch the attack. The intermediary nodes can be servers and routers. As shown in Figure 2.3, an attacker sends packets which require response to the reflector with the source address forged to be a victim's address. Without realizing the spoofed address, the reflector returns response packet according to the type of attack packet. If the number of reflectors is large enough, the victim's link will be flooded and get congested.

As shown in Figure 2.3, the attack architecture to launch reflector attack is similar to the direct attack except that there are reflectors between the victim and agents. A reflector attack requires a set of predetermined reflectors, including DNS servers, HTTP servers, and even routers. Advantage of the reflector attack is that, the attacker does not need to implant attack agent to launch such kind of attack. More importantly, the reflected packets are in fact normal packets with legitimated address and packet types which is not easy to be filtered.

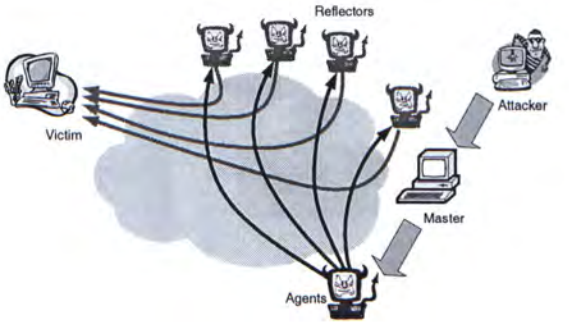


Figure 2.3: Reflector DoS Attacks.

Smurf [3] is a well-known reflector DoS attack which using a forged ICMP Echo Request packet. The attacking packets are forged to set source address by targeting victim's address and sent to a subnet-directed broadcast address. After receiving a redirected ICMP Echo Request, each machine within the subnet will send ICMP Echo reply to the victim. Thus the victim is overwhelmed. Since reflector attacks are based on the reflectors' ability to generating traffic, any protocol supports this response requirement can be exploited to launch reflector attack. Thus TCP and UDP packet can also be used within a reflector DoS attack [52].

2.1.3 Spoofed Packet Filtering

To defend these victim exhaustion kinds of DoS/DDoS attacks, one of the most intuitive and effective countermeasures is packet filtering. First, the DoS/DDoS attack or attack packet is identified, and then the filtering mechanism classifies those packets and drops them. The overall performance of this detect-and-filter mechanism depends on the effectiveness of both phrases. As described in Figure 2.4, the detect-and-filter approach can be performed in different places between the victim and the agents or reflectors.

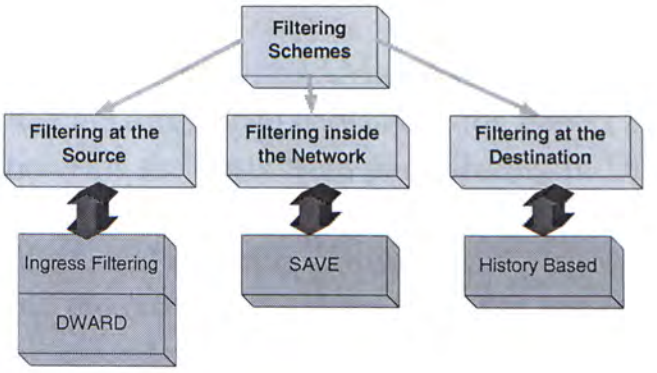


Figure 2.4: Spoofed Packet Filtering.

Filtering the Attack Traffic at the Source

Although it is difficult to detect DoS/DDoS attack at the source networks, it is quite effective to filter the packets as well as other illegitimate packets close to the source. A ubiquitous deployment of the ingress filtering [19] can drop all packets sent in direct attacks and all packets sent from agents to reflectors, if all the attack packets adopt randomly spoofed address. The ingress filtering mechanism does not allow outgoing packets with invalid IP address prefix. Thus, it will prohibit an attacker within the originating network form launching an attack using forged source addresses that do not conform to ingress filtering rule. An additional advantage of this type filtering is that it enables the originator to be easily traced to its true source, since the attacker would have to use a valid and legitimately reachable source address. Although it is easy for ISP to implement such type of filtering, it requires widely deployment to be effective against DDoS attacks. And this filtering mechanism actually does nothing to protect against flooding attacks which originate from valid prefixes (IP address), or even attacks using a forged source address of another host within the permitted prefix filter range. Also Mobile IP [49] will be specially affected by this ingress filtering. The "reverse tunnels" [40] need to be adopted to accommodate ingress

filtering for mobile IP.

Another solution proposed by J.Mirkovic is call **D-WARD** [25]. It is deployed at the source-end networks (stub networks or ISP networks) and prevents the hosts from participating in DDoS attacks. D-WARD is configured with a set of address whose outgoing traffic should be policed, and monitors two-way traffic between the police address set and the rest of the Internet. Traffic statistics are maintained per flow and per connection for each type of traffic. Statistical analysis is performed to detect malicious activity. Suspicious flows are rate limited.

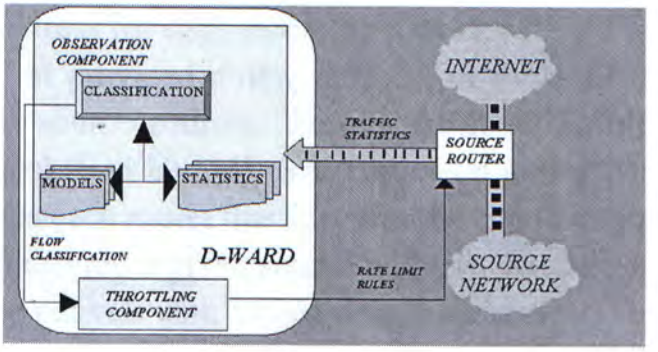


Figure 2.5: D-WARD Architecture.

As shown in Figure 2.5, D-WARD is a feedback system which consists of observation and throttling component. The observation component monitors all packets passing through the source router, gathers statistics on two-way traffic and detects the possible attacks. Periodically, statistics are compared to models of normal traffic and results are passed to the throttling component, which adjusts the rate limit of the source router. The imposed rate limits modifies the associate traffic flows and thus affect future observations. Since D-WARD is deployed close to the source of attack, it can effectively mitigate the damage. The attack flow can be filtered before they enter the Internet core

Filtering the Attack Traffic inside the Network

Ingress Filtering is to perform address validation at the source, and actually due to the wide deployment cost, it is far from enough to prevent all possible DDoS attack. Research has shown that unless ingress filtering is deployed almost everywhere, nearly arbitrary forgery is still possible [46]. Thus providing a solution to perform similar kind validation within the core network can greatly help network security, arrack tracing and network problem debugging. However, due to the asymmetries in today's Internet routing [47], routers do not have readily available information to verify the correctness of the source address for each incoming packet.

Jun Li et at provided a new protocol to force all IP packets to carry correct source address: **Source Address Validity Enforcement Protocol (SAVE)** [34]. SAVE runs on individual routers and build a table at each router that specifies the valid incoming interface for packets carrying a given source address. Routers use this table to filter those packets with forged source address.

SAVE assumes that each router is associated with a set of source addresses. All packets from this address space can only reach some set of destinations via this router. For each entry in its forwarding table, a SAVE router periodically generates SAVE updates directed toward the corresponding destination address space. Forwarding table changes will also trigger new SAVE updates. In both cases, an update specifies the originating source address space and carries the destination address space. Since SAVE updates arrive on the same incoming interface as valid IP packets, routers between the source and final destination can record the legitimate incoming interface for the specified source address space.

Filtering the Attack Traffic at the Destination

Due to the aggregation of attack traffic, it is relatively easy to detect the DoS/DDoS attack at the destination (victim's network). The

most difficult task is to differentiate between legitimate traffic and attack traffic, especially in case of highly distributed DDoS attacks. Tao Peng et al proposed a protection scheme using **History-based IP Filtering (HIF)** [48]. It is observed that most IP addresses of legitimate packets to a given network take a small set of values, representing regular users, while the attack packets which contain a false source address, are usually randomly generated [26]. As a result, it is effective to differentiate legitimate packets from malicious packets based on the history of previous successful connections. In the HIF scheme, the edge router keeps a history of all the legitimate IP addresses which have previously appeared in the network. When the edge router is overloaded, this pre-built IP address database is used to admit the incoming packets. This mechanism will work in most of the cases. However, if the attack is launched without forging the IP address or forging the address to a group of real hosts which have successful connected to the victim before, false negative will be quite high when filtering the attack packets.

2.1.4 IP Traceback

Attack source traceback and identification is usually an after-the-fact response to a DDoS attack. IP traceback refers to the problem, as well as the solution, of identifying the actual source information in the packet. Since during the DoS attack, especially the DDoS attack, information of the attack packets are usually spoofed, one need to identify the true originators of attack packets by constructing the attack path or attack tree. By doing effective IP traceback to the attack sources, system administrator can isolate or shut the attack facility down and meanwhile hold attackers accountable.

To evaluate the effectiveness of an IP Traceback Scheme, several features need to be considered. Accuracy is the most important. Scheme should provide accurate information about routers near the attack sources rather than those near the victim. A scheme also

should be immune to attacker's misleading, recognizing and excluding the false information injected by the attacker. **False Positive Rate (FPR)** and **False Negative Rate (FNR)** are the metrics used to quantitatively measure the effectiveness of traceback scheme. The FPR is given by the rate of wrongly identifying source that did not actually emit attack packet. And the FNR is given by the rate of failing to identify an attack source. Also, an effective traceback scheme should not bring large overhead to the system (router or host). It should avoid using a large amount of packets to construct the attack path or attack tree. If packet information is to be maintained at the intermediate routers then collecting this information must be efficient. Low processing and storage overhead are also necessary at the intermediate routers.

Up till now, many kinds of IP traceback schemes have been proposed like [22, 31, 58, 59, 64], which are mainly belongs to three categories: sampling based, logging based and traffic engineering based as shown in Figure 2.6.

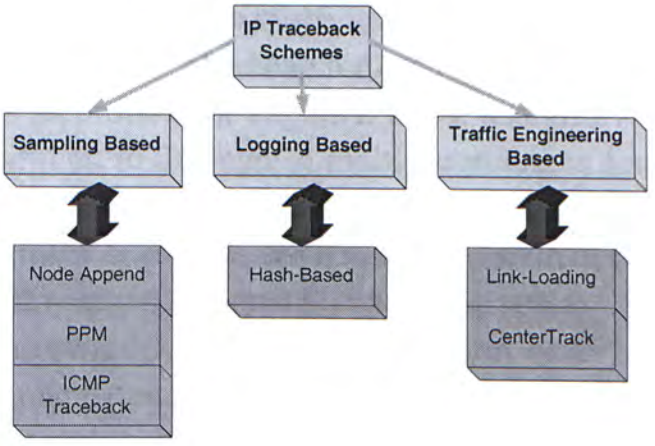


Figure 2.6: IP Traceback.

Sampling based IP Traceback Schemes

The simplest algorithm is **Node Append** which is to append each node's address to the end of the packet as it travels through the network from attacker to victim. The attack path can be constructed on the sequence of routers embedded in any of the attack packets. The node append algorithm is both robust and extremely quick to converge, since only one packet is required to obtain the list of routers. However, the disadvantages are the large overhead at routers and significant increase of packet size.

To reduce both the router overhead and the per-packet space requirement, some kinds of **Probabilistic Packet Marking (PPM)** schemes have been proposed [53]. For the PPM Node Sampling scheme, the path is sampled one node at a time. Router marks each packet with probability p. Thus the probability of receiving a marked packet from a router d hops away is $p(1-p)^{d-1}$. As a result, the order of the routers on the attack path can be deduced from the relative number of marked packets per router. Compared with Node Append, **PPM Node Sampling** has a much lower processing overhead at the router and lower packet overhead for each marked packet (Only one IP address is recorded.) on the other hand, using PPM Node Sampling, it is difficult to distinguish multiple attackers since routers at the same distance will give the same percentage of marked packets. And huge number of packets are needed to construct the attack path which can not respond fast enough under the attack.

A straightforward solution to the problems of PPM Node Sampling is to explicitly encode edges in the attack path rather than simply individual nodes: **PPM Edge Sampling**. Three pieces of information need to be recorded in each packet. The start and end router of each link and the distance of the edge sample from the victim. In a router with PPM Node Sampling Scheme, if router decides to mark a packet, it will write its own address in start field and zero in distance field. Otherwise, it will write address to the end field if

the distance is zero which means the packet has just been marked by the direct up stream router. Also, if decide not to mark a packet, it will increase the distance value in the distance field. During the attack, the attack path can be constructed according the following algorithm:

Edge Sampling Path Reconstruction Algorithm:
Let G be a tree with root v.
Let edges in G be tuples $(start, end, distance)$

1. For each packet w from attacker
2. if $(w.distance)$ == 0 then
3. insert edge $(w.start, v, 0)$ into G
4. else
5. insert edge $(w.start, w.end, w.distance)$ into G
6. Remove any edge (x, y, d) with $d \neq$ distance from x to v in G
7. Extract path $(Ri..Rj)$ by enumerating acyclic paths in G

Like Node Sampling, the router overhead of Edge Sampling is not large. And multiple attackers can be traced by this scheme. But additional packet space (72bits) is still a requirement. Some improved schemes are proposed to reduce the space requirement which encodes edge information in the 16 bit of IP header identification field. However, problems are also caused by the encoding process. Encoding requires more packets to process and when packet is fragmented, marking becomes impossible.

Similar to the Probabilistic Packet Marking Scheme is an **ICMP Traceback** scheme provided by Bellovin [9], in which routers generate ICMP packets to the destination with a low probability. For a significant traffic flow, the destination can gradually reconstruct the route that was taken by the packets in the flow. This scheme does

not need to write any information in the packets. But it will increase the network traffic. Also, ICMP packets may get discarded by some routers.

Logging based IP Traceback Schemes

Another important traceback scheme is to log packets at various points throughout the network and then use appropriate extraction techniques to discover the packets path through the network. Logging requires no computation on the routers fast path and, thus, can be implemented efficiently in todays router architecture. However, since attack packet may be transformed as it moves through the network, attack path reconstruction is difficult. And memory requirements are prohibitive at high line speeds which makes full packet storage problematic. Additionally, traffic repositories may aid eavesdroppers, making extensive packet logs a privacy risk.

To come over the challenges of logging based IP traceback schemes, Snoeren et al proposed a **Hash-Based IP Traceback Scheme** [57] that generates audit trails for traffic within the network, and can trace the origin of a single IP packet delivered by the network in the recent past. The Hash-Based IP Traceback Scheme which is actually called *Source Path Isolation Engine (SPIE)* use a Bloom filter to reduce the memory requirement for logging (down to 0.5% of link bandwidth per unit time). By storing only packet digests, and not the packets themselves, SPIE also does not increase a networks vulnerability to eavesdropping. SPIE therefore allows routers to efficiently determine if they forwarded a particular packet within a specified time interval while maintaining the privacy of unrelated traffic.

Figure 2.7 shows the three major architectural components of the SPIE system. Each SPIE-enhanced router has a Data Generation Agent (DGA) associated with it. The DGA produces packet digests of each packet as it departs the router, and stores the digests in bit-mapped digest tables. The digest tables are stored locally at the DGA for some period of time, depending on the resource con-

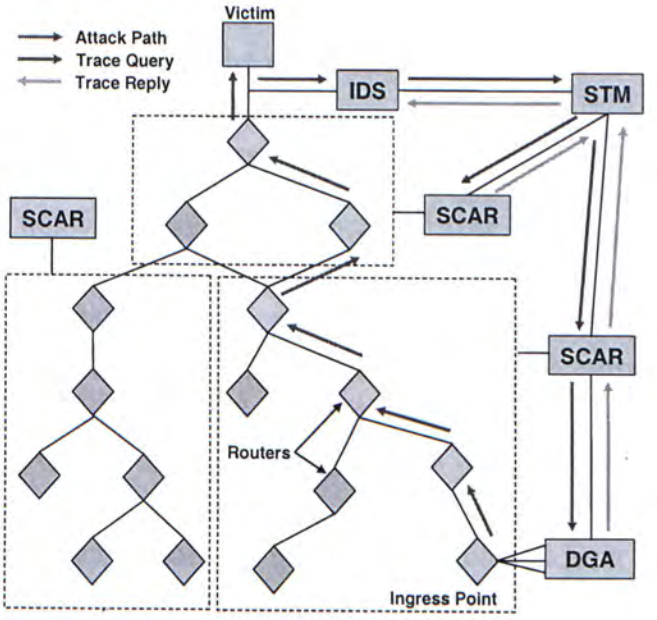


Figure 2.7: SPIE Architecture.

straints of the router. If interest is expressed in the traffic data for a particular time interval, the tables are transferred to a SPIE Collection and Reduction (SCAR) agent for longer-term storage and analysis. SCARs are responsible for a particular region of the network, serving as data concentration points for several routers. They collect digest information from DGAs and build local attack graph. The attack graphs from each SCAR are grafted together to form a complete attack graph by the SPIE Traceback Manager (STM). The STM controls the whole SPIE system. When a traceback request is presented to the STM, it verifies the authenticity of the request, dispatches the request to the appropriate SCARs, gathers the resulting attack graphs, and assembles them into a complete attack graph. Upon completion of the traceback process, STMs reply to intrusion detection systems with the final attack graph.

Note that DGA computes digests over the invariant portion of the IP header and the first 8 bytes of the payload. Frequently modified

header fields including TTL, TOS, checksum and certain IP options fields are masked prior to the digesting.

Traffic Engineering based IP Traceback Schemes

Burch and Cheswick have developed a traffic engineering based IP traceback schemes that does not require any support from network operators [11]. We call this technique **Link-Loading Traceback** or controlled flooding because it tests links by flooding them with large bursts of traffic and observing how this perturbs traffic from the attacker. Using a pre-generated map of Internet topology, the victim coerces selected hosts along the upstream route into iteratively flooding each incoming link on the router closest to the victim. Since router buffers are shared, packets travelling across the loaded link C including any sent by the attacker C have an increased probability of being dropped. By observing changes in the rate of packets received from the attacker, the victim can therefore infer which link they arrived from. As with other link testing schemes, the basic procedure is then applied recursively on the next upstream router until the source is reached.

Although this scheme is simple and easy to implement, it has many drawbacks. The most problematic weakness is that Link-Loading is itself a denial-of-service attack C exploiting vulnerabilities in unsuspecting hosts to achieve its ends. This drawback alone makes it unsuitable for routine use. Also it is poorly suited for tracing distributed denial-of-service attacks, because the link-testing mechanism is inherently noisy and it can be difficult to discern the set of paths being exploited when multiple upstream links are contributing to the attack. Finally, this scheme requires being done during the attack and cannot be used after the attack.

Stone have also developed another traffic engineering based IP traceback schemes called **CenterTrack** [61]. The CenterTrack is an overlay network which consists of edge routers and one or more trace routers using IP tunnels. The overlay network is used to selec-

tively reroute the "interesting" packets directly from edge routers to special tracking routers. Hop-by-hop tracking is then adopted, starting with the tracking router closest to the victim. Input debugging is performed on the router of each hop, thus packets are examined, then dropped or forwarded to appropriate egress point. Note that BGP is used for routing updates between the tracing network and the backbone network.

The tracing capability of CenterTrack only depend on edge routers and special purpose tracking routers, thus the number of routers required to perform tracking is reduced. However, similar to Link-Loading Scheme, traceback process of CeterTtrack must be accomplished during attack period which imposes a strict timing constrains on the administrators. And when attacks originate from the backbone network itself but not edge routers, CenterTrack is not able to track these attacks. Also packet rerouting overhead at edge routers can not be ignored.

2.1.5 Location Hiding

Mechanisms to defend against DoS attack can be divided into reactive approaches and proactive approaches. Most Filtering approaches and IP Traceback belong to the former approaches. Since all of them will wait for an attack to be launched before taking appropriate measures to protect the network. A proactive approach will eliminate all possibility of becoming a target by aggressively blocking all incoming packets whose source addresses are not approved. Sometimes the reactive approaches may not be very effective since reactive methods that filter traffic by looking for known attack patterns or statistical anomalies in traffic patterns can be defeated by changing the attack pattern and masking the anomalies that are sought by the filter.

A. Keromytis et al proposed a kind of proactive defense mechanism called **Secure Overlay Services (SOS)** [28]. They use a

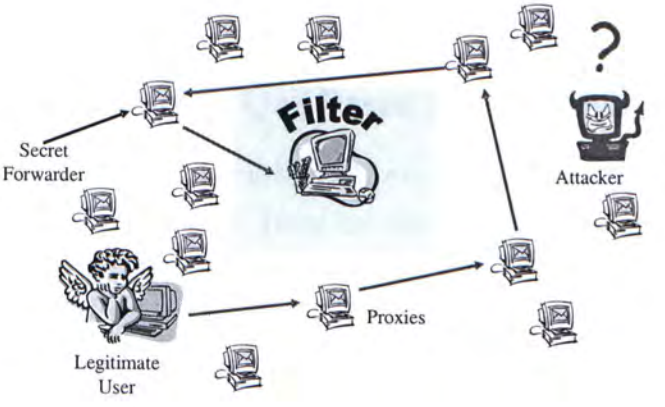


Figure 2.8: SOS: Secure Overlay Services.

combination of secure overlay tunneling, routing to construct the SOS architecture via consistent hashing and filtering. The goal of SOS architecture is to allow communication between a confirmed user and a target while hiding the target to unauthorized users. It is assumed that the set of nodes that participate in the overlay is known to the public and to the attacker as well. In a SOS network, a large number of proxies are created. The target selects a small subset of proxies as secret forwarders. The filter at the target only allows packets whose source address matches the address of secret forwarders to pass through. As a result, shown in Figure 2.8, if a host contacts any proxy, its legitimacy will be checked first. Then proxy randomly routes packet to another proxy. If destination proxy is secret forwarder, packet forwarded to target, otherwise packet randomly routed to another proxy. Thus, with filters, multiple proxies, and secret forwarder(s), attacker cannot focus the attack.

SOS can prevent the DoS attack in a proactive manner, however large number of hosts need to be involved to create the overlay. The deployment cost will be high. Also, it takes relatively a longer time for normal user to access the target due to the overlay routing. It is a trade off between the security and timely delivery.

2.2 QoS Based DoS Attacks

2.2.1 Introduction to the QoS Based DoS Attacks

To provide real-time communication services to multimedia applications or subscribed-based Internet users, the proposed network QoS (Quality of Service) infrastructure like Diff-Serv (Differentiated Service) [10] reserves network resources for real-time traffic. Within the network QoS architecture, however, the reserved network resource is susceptible to resource theft and abuse. Without a resource access control mechanism that can efficiently differentiate legitimate real-time traffic from attacking packets, the traffic conditioning and policing enforced at ISP (Internet Service Provider) edge routers cannot protect the reserved network resource from embezzlement. Such kind of attacks, targeting at reserved network resources and violating network QoS guarantees, can be called **Quality of Service based DoS Attacks** or **Denial of Quality of Service (DQoS) Attacks**.

Basically, there are two distinct kinds of QoS-based DoS attacks: The first class is control flow attacks, e.g., killer reservation in Resource Reservation Protocol (RSVP) [68], which directly attack the signaling/control protocol in the control plane for network resource reservation and connection setup. In the absence of authorization, a user may request and reserve any amount of resources. Additionally, a user may forge, delete, or illegally modify a reservation messages.

Another kind of QoS-based DoS attack is data flow attack (e.g., resource theft in the data plane), in which bogus data packets grab the reserved bandwidth from the owners or genuine real-time data flows.

2.2.2 Countermeasures to the QoS Based DoS Attacks

For the QoS Based DoS Attacks to the control flow, a secure communication tunnel is necessary for the control flow. ARQoS Project

from North Carolina State University [4] and Authenticated QoS Project from University of Michigan [5] are provided. They proposed a secure RSVP, in which resources are reserved online using strong authentication, and subsequently, compliance with the reservation request parameters is verified.

For the QoS Based DoS Attacks to the data flow, the overhead in data transmission for such kind of heavy-weighted authentication can not be negligible. Since latency caused by authentication may violate the delay requirements for delivering real-time data streams. Haining Wang et al proposed a fast and light-weighted IP network edge resources access control mechanism, called **IP Easy-pass** , to prevent unauthorized access to reserved network resources at edge network. A unique pass is generated and attached to each legitimate real-time packet so that an ISP edge router can validate the legitimacy of an incoming packet very quickly and simply by checking its pass. Generation of pass is done at the sending host. A sequence of passes are generated according to a priori agreed upon rules between the sending host and the ISP edge router. The passes are encrypted using RC-5 because it is fast, fully parameterized and secure. The verification of passes is done at edge router.

Note that the IP Easy-pass mechanism which is adopted in the data plane is base on an important assumption. It is assumed that there exists a secure channel for QoS signaling in the control plane between a given end-host and the ISP edge router that connects the end-host to the Internet. Prior to data transfer, through the secure QoS signaling channel, the end-host and the ISP edge router must communicate shared secrets for generating Easy-passes.

2.3 Worm based DoS Attacks

2.3.1 Introduction to the Worm based DoS Attacks

Internet worms, which is malicious code that propagates over the Internet without or with very little human assistance [29], have become a new threat towards the Internet. Massive worm-driven DDoS attacks have appeared and have been reported to cause great damage due to the bandwidth consumption [33], e.g. the Slammer Worm [41] which was the fastest worm in history, infected more than 90 percent of vulnerable hosts within only 10 minutes, causing significant disruption to the Internet. And hosts infected by Code Red Worm [15] in 2001, were going to launch the Denial of Service Attack against *www1,whitehouse.gov* from the 20th to the 28th of each month. *windowsupdate.com* was also attacked by the Blaster Worm in 2003 [6].

Internet Worm is a new effective tool to launch Denial of Service attack. Since it can replicate itself independently and has many other automation properties. Unlike traditional DoS attack tools (Trioo, TFN200...), attackers do not need to compromise large number of hosts before launching the attack. Attack can be originated much easier than before. Usually worm is affected through system weakness (e.g. buffer overflow) or E-Mail, like worm Nimda can replicated itself not only through both the two ways above but also spread through windows file sharing protocol (SMB). It is reported that the disruption caused by Internet Worms significantly exceed that caused by other DoS attack tools [15].

2.3.2 Countermeasures to the Worm Based DoS Attacks

Many mechanisms to defend the Internet worm have been proposed [16], [62], [67]. Measures to counter the Internet worm are basically belongs to prospects: passive and active [35]. The passive measures attempts to block or slow down the worm traffic to prevent spread

and the active measure will try to proactively patch vulnerable hosts or remove infections from hosts in response to an attack.

One of the most simple passive measures is **Content Filtering**. Defenses based on content filtering suppose that worm packets have discernible signatures. They discover and disseminate these signatures to a distributed infrastructure that filters traffic to eliminate discovered worm packets. Success of these defenses depends on very fast response to an attack, and a filtering infrastructure that protects a large fraction of the network.

Another passive defense is based on blocking traffic from certain specific IP addresses (**Address Blacklisting**) that are known or suspected to be infected by the worm. This method tends to be less effective than content filtering due to the delay in detecting every new infected host.

Actually, to be effective, the passive measures require widespread deployment in the Internet. While active measures rather than just blocking worm scans, attempt to actively inhibit the spreading worm, such as launching a second worm that tries to patch vulnerable hosts to prevent infection or remove the first worm from infected systems, do not require infrastructure to be in place. Nicol et al presented five defense types (Empty defense, Simple Patch, Spreading Patch, Nullifying Patch and Sniper Defense) based on evaluating the level of activeness [44]. Moreover, worms that attempt to patch against or remove another worm have actually been seen in the wild, as in the case of the Welchia worm [20] removing Blaster. However, evidence has shown that Welchia has caused at least as much of a problem as Blaster. Thus more issues have to be considered for a successful active worm defense mechanism including both technical and legal issues. And research done by Michael Liljenstam et al provided a comparison between the passive and active worm countermeasures in terms of the effectiveness in preventing worm infections. With sufficient deployment, passive measures such as content based quarantining defenses are more effective than the active worms.

2.4 Low-rate TCP Attack and RoQ Attacks

2.4.1 General Introduction of Low-rate Attack

The low-rate TCP attack is first described by Kuzmanovic and Knightly [30], who characterize the attack and point out important challenges of detection and defense. It is a new kind of TCP targeted denial of service attack which is quite different from the common DoS/DDoS attacks.

As describe before, in a successful common DoS/DDoS attack, a large number of hosts or agents will be involved. Although it is harmful, it is also relatively easy to be detected due to the large volume of attack traffic. For low-rate TCP DoS attack, the aim is to deny the bandwidth of legitimate TCP flows. The attacker will send a low volume of attack traffic to avoid detection. At the same time, the attack exploits the TCP congestion control feature and sends a periodic burst to the victim, either a web site or any upstream routers of the victim site.

Since the average throughput is small, normal throughput based intrusion detection can not work. And information in the attack packet can be spoofed, there is no signature in each packet which means that the per packet analysis approach also may not be effective. Traditional packet filtering schemes are seriously challenged due to the above properties. There are several pieces of work addressing this problem as shown in [37], [12], [63], [65].

Since low-rate attacks are most effective when the retransmission attempts by TCP sources are synchronized following a congestion, randomizing the TCP RTO is an intuitive solution approach and has been shown to be effective in [65]. However, randomizing the RTO requires widespread updates of existing end user software and may reduce the performance of TCP under non-attack conditions [8]. In comparison, we seek a solution at the router level (Refer to the following chapters of the thesis).

2.4.2 Introduction of RoQ Attack

Another work addressing similar problem appeared in [23]. The **Reduction of Quality (RoQ) Attack** presents a more general class of adversarial network traffic exploiting the transients of adaptation. It is shown that a well orchestrated attack could introduce significant inefficiencies that could potentially deprive a network element from much of its capacity, or significantly reduce its service quality, while evading detection by consuming an unsuspicious, small fraction of that elements hijacked capacity. A mathematical model was proposed and a measurement was carried at to illustrate the attack potency in [23]. Since the attack form presented [23] is similar to the low-rate attack (also periodic burst), we believe at least, our detection and defense mechanism presented in later chapters will shed light towards the solution to such RoQ attack.

□ **End of chapter.**

Chapter 3

Formal Description of Low-rate TCP Attacks

Summary

This chapter provides a mathematical description of the low-rate TCP attack in Section 3.1. This model covers the whole family of possible forms of attack that we consider to detect and defend. In Section 3.2, some more general forms of low-rate attack based on the math model are talked about.

Because the low-rate attack can appear in many different forms (as describe below), let us first provide a formal model in describing a low-rate TCP attack. Given this mathematical description, one can generate a large family of low-rate attacks. We then proceed to describe how one can extract "*signatures*" from this large family of low-rate TCP attack flows.

3.1 Mathematical Model of Low-rate TCP Attacks

A low-rate TCP attack is essentially a periodic burst which exploits the homogeneity of the minimum retransmission timeout (RTO) of

TCP flows. Consider a router with capacity C (with unit of bps), one form of attack is a periodic square wave as reported in [30]. The period of this square wave is denoted by T, which is approximately one second so as to effectively forcing other TCP flows to enter the retransmission period. Within each period, the square wave has a magnitude of zero except for l units of time. During this time, the square wave has a magnitude of a normalized burst of R. Note that in this work, the magnitude of the burst is *normalized* by the router's capacity C, therefore $R \in (0, 1]$. Although it is possible that in reality R may exceed 1, in our model we mainly focus on the range $(0, 1]$, since when $R > 1$, it will clearly cause packet loss and can be treated as the same class with $R = 1$. The average normalized bandwidth, of this periodic square wave is Rl/T. Again, the objective of the low-rate attack is that for a short duration l, the attack packets will fill up the buffer of a victimized router so that packets of any TCP flows have to be discarded by the router and forcing most, if not all TCP flows to enter the retransmission state. Also note that to be considered as a low-rate TCP attack, the ratio of l/T has to be small or else system administrators can easily detect a high volume attack.

A general model of a low-rate TCP attack can be described by five parameters (T, l, R, S, N). The parameters T, l and R have the same meaning as described above, S denote the amount of time-shift, starting from the initial measurement instant of the signal (e.g., $t = 0$) to the beginning of the attack pulse, while N denote the amount of *background noise* or *traffic*. The background noise is due to other UDP flows, which will not backoff in the midst of congestion, or other TCP flows which are not in the retransmission period. Figure 3.1 illustrates an example of low-rate TCP attack traffic.

Let us define the valid range of these five parameters.

- **Values for T:** As indicated in [30], the most effective value for the periodic low-rate attack is $T = 1$ second. In our study, we consider a larger range of T, which is $T \in [1.0, 1.5]$.

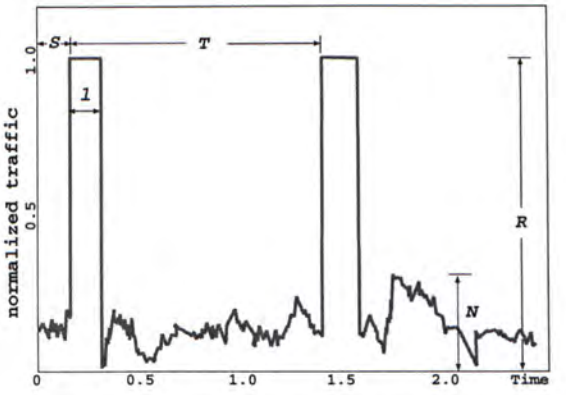


Figure 3.1: Low-rate attack traffic with parameters (T, l, R, S, N).

- **Values for l:** Assume that we have K TCP flows which are affected by the low-rate TCP attack. Let RTT_i represents the round-trip time from the source i of the TCP flow to the victimized router. To have an effective attack, the low-rate attack burst length should last long enough to keep the router's queue full for all RTT timescales. Therefore, $l \geq \max_i\{RTT_i\}$, for $i = 1, 2, \ldots, K$. Since the aim of the low-rate TCP attack is to avoid sending a high volume of traffic so as to avoid easy detection, the value of l cannot be very close to T. In our study, we have $\max_i\{RTT_i\} \leq l \leq \beta_1 T$ where $\beta_1 \leq 0.3$.

- **Values for R:** Since this is a normalized burst with respect to the router's capacity C, we have $R \in (0, 1]$.

- **Value of S:** The amount of time-shift S, starting from the initial point of measurement (e.g., $t = 0$) to the beginning of the attack pulse, has a valid range of $0 \leq S \leq T - l$.

- **Value of N:** The amount of normalized background noise due to other UDP or TCP packets, it has a valid range of $0 \leq N \leq \beta_2 R$ where $\beta_2 \leq 0.5$. Note that background noise is a general assumption, which exists most of the time in realtime signal processing. Note that adding a background noise to the model

makes the detection a more challenging task. Yet, in realistic situation, noise is always present in the sampled traffic.

3.2 Other forms of Low-rate TCP Attacks

Based on the mathematics model above, more general attack wave form can be generated. For example, the attack traffic can be of the form of sine wave and an attacker can also generate different burst patterns within each sub-period. Figure 3.2 illustrates an instance of the general attack traffic which has three attack sub-periods, each sub-period has a different attack characteristic. The first sub-period has $T_1 = 0.8$ sec and $l_1 = 0.1$ sec, the second sub-period has $T_2 = 1.2$ sec and $l_2 = 0.3$ sec while the last sub-period has $T_3 = 1.0$ sec and $l = 0.2$ sec. The generality of attack waveforms makes it difficult and challenging to characterize, detect and defend the low-rate TCP attack.

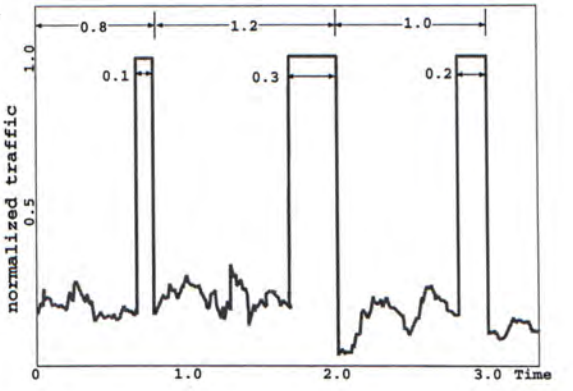
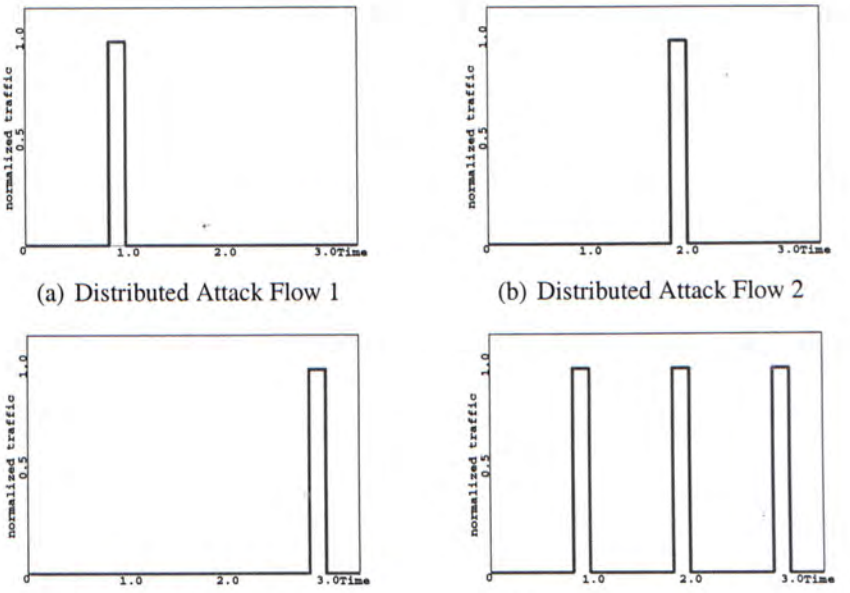


Figure 3.2: General attack traffic with a varying pattern within each sub-period.

Another important point that is worth mentioning is that the low-rate TCP attack can be launched by either a single source, or by multiple distributed sources. For the single source attack, it is easy to generate and it is effective when there is sufficient bandwidth along the path between the attack source and the victimized router. For

the distributed low-rate TCP attack, it is also possible to synchronize attacks over independent sources on the Internet, since jitter on the Internet is usually small, and it is on the order of 1RTT <100ms. However, compared with single source attack, issues concerning different propagation and transmission delays to the victimized router still need to be addressed. Thus, more effort is needed to generate a distributed attack. There are at least two approaches to generate a distributed attack. For the first approach, each of the M attack sources generates a homogeneous and periodic attack waveform with a normalized burst size of $R \geq 1/M$. These flows will converge into a sufficient large burst at the victimized router and force all affected TCP flows to backoff. Another possible form of distributed attack, which has a lower synchronization requirement, is that each attack source generates a large burst but for a longer period. For example, each of the M attack sources generates a homogeneous and periodic attack waveform with $T = M$ seconds and a normalized burst size of $R = 1$. This kind of attack is illustrated in Figure 3.3 for three distributed attackers. The i^{th} attack source sends the attack burst at the i^{th} attacking sub-period and keeps silent for the remaining sub-periods. The converged attack traffic is illustrated in Figure 3.3(d).

□ **End of chapter.**



(a) Distributed Attack Flow 1

(b) Distributed Attack Flow 2

(c) Distributed Attack Flow 3

(d) Aggregated Attack Flow

Figure 3.3: Distributed low-rate attack with period $T = 3$ and $M = 3$ attack sources.

Chapter 4

Distributed Detection Mechanism

Summary

This chapter provides the detection solution to the low-rate attack. First, the general idea of distributed detection mechanism is presented (Section 4.1). Then the detail detection algorithm is provided from Section 4.2 to Section 4.6. Finally, groups of experiments of detection solution are conducted in Section 4.7.

4.1 General Consideration of Distributed Detection

Before we discuss how to defend against this family of low-rate TCP attacks, the first issue we need to address is how to perform an effective *detection* and that the detection method has to be computational efficient. Unlike other intrusion detection or DDoS detection methods [38] [66], one cannot simply install the detection mechanism at the victim site, say S (i.e., a web server). The reasons are as follows: First, to install the detection mechanism at the victim site, status of thousands of incoming TCP flows need to be monitored which maybe a burden to the server. More importantly, the low-rate TCP attack has the intrinsic characteristic to throttle legitimate TCP

traffics at the victim site $\mathcal{S}$. Therefore, an attacker does not necessary need to aim the attack at the victim site, but rather at a "subset of upstream routers" of $\mathcal{S}$ so as to throttle all TCP flows passing through these routers. Thus installing a detection mechanism at the victim will be ineffective since it provides no information for the victim site to determine where the attack is occurring, or the attack traffic is originated from which part of the network. As a result, any detection method installed at the victim site may not be very effective because the victim site only may not even detect the existence of attack. Instead, the victim site may think that only few users are interested to access information from the site $\mathcal{S}$ if is under the low-rate attack.

In this work, we propose a distributed detection mechanism that is installed at a set of routers which are $k \geq 1$ hops away from the victim site. Each router needs to perform the low-rate TCP attack detection on the *output port* that is forwarding packets to the victim site $\mathcal{S}$. If a low-rate TCP attack is detected, then the router needs to determine which input port(s) the low-rate attack is coming from. Detection will then be carried out on all these input ports of the affected router. If a low-rate attack is detected on an input port, say $\mathcal{P}$, then the affected router will push back the detection to all upstream routers that are connected to the input $\mathcal{P}$. If the affected router cannot detect any low-rate attack on any of its input port, this implies that the low-rate attack is carried out in a distributed manner, then the defense mechanism (which we will discuss in Chapter 5) will be carried out. Note that there are several important features of using the above distributed detection mechanism. They are:

- Detection is carried out from the output port to the input ports.
- Pushing the detection of low-rate attacks as close as possible to the attack sources so as to minimize the damage to other legitimate TCP flows when adopting the defense mechanism (Chapter 5).

The overall detection mechanism is as follows:

Distributed Detection Mechanism
Let $\mathcal{R}$ be the enabled router. $\mathcal{P}_i$ is the set of input port of $\mathcal{R}$, $\mathcal{P}_o$ is the output port $\mathcal{R}$ uses to forward packet to the victim site $\mathcal{S}$.

1. $\mathcal{R}$ determines the existence of low-rate attack on $\mathcal{P}_o$;
2. **If** (low-rate attacks exist) {
   determine the existence of low-rate attack on $\mathcal{P}_i$;
3. **If** (attack exits for input port $\mathcal{P} \in \mathcal{P}_i$) {
4. signals all upstream routers connected to $\mathcal{P}_i$ to perform distributed low-rate attack detection;
5. }
6. execute the defense mechanism describes in Sec. 5;
7 }

Let us describe in detail about the low-rate attack detection algorithm.

4.2 Design of Low-rate Attack Detection Algorithm

Because attack packets can be easily generated, all information in the packets' header can be spoofed, e.g., IP source addresses and types of transport protocol used, and there is no easy way to accurately differentiate low-rate TCP attacking packets from legitimate packets. The proper approach for the low-rate TCP attack detection is to compare the incoming traffic with attack pattern signatures.

The detection mechanism will be installed at enabled routers and the detection mechanism involves the following steps.

- **Statistical sampling of incoming traffic:** traffic will be sampled and normalized based on the transmission capacity of the link/port.
- **Noise filtering:** since other packets which arrive during the non-active period of the low-rate attack will also be included in the sampling process, therefore, one needs to perform filtering before the feature extraction process.
- **Feature extraction:** perform a computationally efficient feature extraction that is immune to time and space shift of the input signals.
- **Signatures comparison:** compare the extracted features of the incoming traffic with the signature of the low-rate TCP attack.

In the following, let us describe in detail the individual step involves in the distributed detection mechanism.

4.3 Statistical Sampling of Incoming Traffic

The router needs to periodically sample the incoming traffic at a constant rate. Note that each sample consists of a record of throughput of the link interface. The record of throughput is the measured throughput between two sample points. The rate of sampling should be frequent enough to record slight variation of the throughput, and at the same time, it should not put a heavy computational burden on the router. In our experiments, we set the rate of sampling to be 100 samples per second which means we will estimate the throughput every 0.01 second. Note that statistical sampling can be easily achieved using standardized algorithms. Additionally, we use T_s to denote the length of each sampling period, which should be properly chosen. In order to capture the periodicity property of the low-rate TCP attack, the sampling period should be lower bounded by $T_S \geq 2T$ according to the sampling theory. One should also put an

upper bound on T_s. Note that a high value of T_s implies a higher storage cost and a higher computational cost for features extraction at the later stage and larger delay in detecting the attack. In our experiments and prototype, we set $T_S = 3$ seconds. Thus we have 300 estimated values of throughput in each sampling period when we set the sampling rate to be 100 samples per second. Another technical issue we have to consider is the *traffic normalization*. Since different link interface may have different line speed, to facilitate feature extraction and comparison at the later stage, the sampled traffic signal of a given link interface will be normalized based on its line speed such that

$$\text{Normalized Throughput} = \frac{\text{Sampled Throughput}}{\text{Maximum Line Capacity}}.$$

4.4 Noise Filtering

Since other packets which arrive during the inactive period of a low-rate attack will also be included in the sampling process, one has to perform filtering before the feature extraction process. Note that beside the potential low-rate TCP attack traffic, some other packets may also be included in the sampling process. These packets include:

- Packets that got forwarded to the same interface but they are not designated to the victim site $\mathcal{S}$.
- TCP packets, especially from flows with large RTT, which may be able to survive under the low-rate TCP attack. Please refer to [30].
- UDP packets which will not backoff in the face of low-rate attack or network congestion.

These types of traffic have either a higher frequency or a smaller magnitude, as compared with the burst magnitude of a low-rate attack. To get a clean signal, a low-pass filter can be used to filter the

high frequencies and at the same time, clamp all sampled signal to zero if it is less than or equal to a fraction β of the peak value R. In our experiments and prototype, we set it to be less or equal to the maximum value of the normalized background noise N, or $\beta \leq 0.5$.

4.5 Feature Extraction

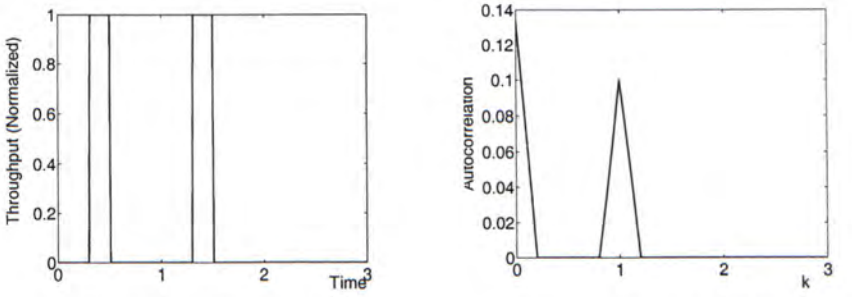
Auto-correlation is used to extract the periodic signatures of an input signal. Using the auto-correlation measure not only because it is easy to calculate (i.e., for a sampled input of size n, the computational complexity is $\Theta(n^2)$), but one can also check the randomness or periodicity of a given signal in the presence of the time shifting variable S.

Auto-correlation is calculated with the unbiased internal normalization. The unbiased normalization is necessary if the input signal has a finite sequence. Consider an input signal with n values $(x_0, x_1, \cdots, x_{n-1})$ and all other $x_i = 0$. The unbiased normalized auto-correlation $A(k)$ can be calculated as follows:

$$A(k) = \frac{1}{n-k} \sum_{i=0}^{n-k+1} x_{i+k} x_i \quad k = 0, ..., n-1. \tag{4.1}$$

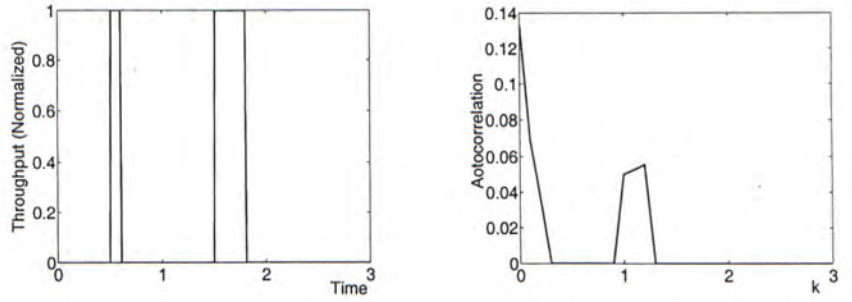
To illustrate this concept, consider the following auto-correlation plots. Figure 4.1(a) shows the noise-filtered input signal with time shift $S = 0.3$ sec and periodic property of $T = 1$ and $l = 0.2$ seconds respective. Note that this is the *"classical"* low-rate attack wave. Figure 4.1(b) shows the corresponding auto-correlation plot. One important observation is that the *peak-to-peak* distance is 1, which captures the *period* of the input signal and that the auto-correlation plot is the *same* independent on the time shift value S. Consider a more complicated attack wave which is illustrated in Figure 4.2(a). In this attack, the time shift $S = 0.5$, the first period $T = 1$ second. For the first attack period, the burst length is $l_1 = 0.1$ while the second attack period has the burst length of $l_2 = 0.3$. The

auto-correlation plot in Figure 4.2(b) reveals the existence of a period (e.g., the peak-to-peak distance in the auto-correlation plot) and that bursts may have different durations.



(a) Input sampled traffic (b) Autocorrelation plot

Figure 4.1: Auto-correlation of input signal $T = 1, S = 0.2, l = 0.2, R = 1.0$.



(a) Input sampled traffic (b) Autocorrelation plot

Figure 4.2: Auto-correlation of input signal $S = 0.5, T = 1, l_1 = 0.1, l_2 = 0.3$.

We extract the feature of auto-correlation plot from an input signal, not only because it captures the periodicity property of the input signal but it also eliminates the problem of time shifting. For the remaining question, we need to address how to compare the auto-correlation plot of an input signal with the auto-correlation plot (or signature) of a low-rate attack.

4.6 Pattern Matching via the Dynamic Time Warping (DTW) Method

After the first three steps, features are extracted from the sampled input, one has to compare the *similarity* of the extracted features with the signature of the low-rate attack traffic and decide whether there is an on going low-rate attack. Note that an example signature of the low-rate attack is depicted in Figure 4.1(b). If the auto-correlation plot of the sampled input is exactly the same as this signature, one can easily conclude the existence of a low-rate attack. However, not all auto-correlation plots of sampled inputs will match exactly as the signature, for instance, the auto-correlation plot in Figure 4.2b). Therefore, one has to do proper processing so as to make an accurate decision.

The mechanism we adopted is called the dynamic time warping (DTW) [27, 52]. It is a robust and computational efficient method to compare the similarity between a template signature and an input signal, even when the input signal is subjected to changes in time scale and magnitude. The dynamic time warping algorithm can be described as follows. Suppose there are two time series, the template $\mathcal{S}$ and an input signal $\mathcal{I}$, of length n and m respectively, where

$$\begin{aligned} \mathcal{S} &= s_1, s_2, s_3, ..., s_n, \quad \text{and} \\ \mathcal{I} &= i_1, i_2, i_3, ..., i_m. \end{aligned}$$

To compare the similarity of these two time series using DTW, one can construct an n-by-m distance matrix $\mathcal{D}$ where $d(x, y)$ of $\mathcal{D}$ represents the Euclidean distance between the signature value s_x and the input signal value i_y, that is

$$d(x, y) = \| s_x - i_y \| \quad \text{for } 1 \leq x \leq n; 1 \leq y \leq m.$$

A warping path $\mathcal{W}$, is a contiguous set of matrix element $\mathcal{D}$ that defines a mapping between the template $\mathcal{S}$ and input $\mathcal{I}$. The k^{th} element of $\mathcal{W}$ is defined as $w_k = d(i_k, j_k)$ where $\mathcal{W} = w_1, w_2, w_3, ..., w_k, ..., w_K$ and $\max(m, n) \leq K \leq m + n + 1$.

The construction of the warping path $\mathcal{W}$ is subjected to the following constraints:

1. *Boundary constraint:* $w_1 = d(1,1)$ and $w_K = d(n,m)$, this requires the warping path $\mathcal{W}$ to start and finish in diagonally opposite corner cells of the matrix $\mathcal{D}$.

2. *Continuity constraint:* Given $w_k = d(a,b)$ then $w_{k+1} = d(a',b')$ where $a' - a \leq 1$ and $b' - b \leq 1$. This restricts the allowable steps in the warping path to be adjacent cells.

3. *Monotonicity constraint:* Given $w_k = d(a,b)$ then $w_{k+1} = d(a',b')$ where $a' - a \geq 0$ and $b' - b \geq 0$. This restricts points in $\mathcal{W}$ to be monotonically spaced in time.

Note that there are many warping paths that satisfy the above constraints. However, we are interested in the path that minimizes the warping cost of $\mathcal{S}$ and $\mathcal{I}$. Formally:

$$DTW^*(\mathcal{S},\mathcal{I}) = \min\left(\sqrt{\sum_{k=1}^{K} w_k}\right). \tag{4.2}$$

In other words, the lower the value of $DTW^*(\mathcal{S},\mathcal{I})$, then the input string $\mathcal{I}$ has higher similarity degree as compare with the signature $\mathcal{S}$. The minimum cost warping path can be found using the *dynamic programming* approach. That is, we construct a matrix γ with dimension of n-by-m, the entry $\gamma(x,y)$ in cell (x,y) defines the *cumulative distances* of the warping path $\mathcal{W}$ from position $(1,1)$ to positive (x,y). The minimum of the cumulative distances of the adjacent elements $\gamma(x,y)$ is:

$$\begin{aligned}\gamma(x,y) &= d(x,y) + \\ &\quad \min\{\gamma(x-1,y-1), \gamma(x-1,y), \gamma(x,y-1)\},\end{aligned} \tag{4.3}$$

where $1 \leq x \leq n; 1 \leq y \leq m$. At each step of calculating the value of $\gamma(x,y)$, if the $\min\{\gamma(x-1,y-1), \gamma(x-1,y), \gamma(x,y-1)\} =$

$\gamma(x-1,y)$ or $\gamma(x,y-1)$, it means that there is one point in the input signal $\mathcal{I}$ that has been matched twice to the template $\mathcal{S}$, or there is one point in $\mathcal{S}$ that has been matched twice to $\mathcal{I}$.

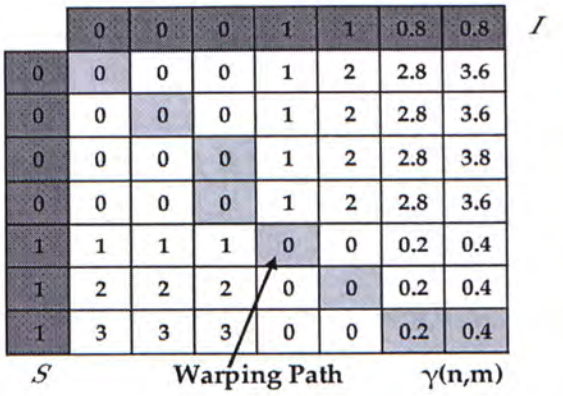
From Equation (4.3), one can see that similar but not identical patterns can match each other with DTW value of 0, i.e, patterns with the same magnitude of burst but different periods like $\{0,0,0,1,1,1,0\}$ and $\{0,0,0,1,1,0,0\}$. Although this scenario is common in other applications like speech recognition and can be viewed as the homology of the input and the template, they should not be treated as identical attack traffic pattern. As a result, we made a modification to the original DTW algorithm that adds some adaptive penalty p for this kind of vertical or horizontal "movement" in the warping path so as to evaluate the similarity while still distinguish the slight difference. Note that the value of the penalty should not be too large since it will increase the DTW value of similar attacks, thus, increase the possibility of false positive or false negative in the detection process. In general, the upper limit of this penalty should not exceed the average value of the template's autocorrelation. As a result, the function of calculating the cumulative distances in our system is:

$$\begin{aligned}\gamma(x,y) \;=\; \parallel s_x - i_y \parallel + \min\{&\gamma(x-1,y-1),\\ &\gamma(x-1,y)+p, \gamma(x,y-1)+p\}\end{aligned} \tag{4.4}$$

After creating this matrix γ, the value $\gamma(n,m)$ is the minimum cumulative distances of the DTW between the template $\mathcal{S}$ and the input $\mathcal{I}$ and it is the solution to Equation (4.2).

To illustrate, consider the following example wherein $\mathcal{S} = \{0,0,0,0,1,1,1\}$ and $\mathcal{I} = \{0,0,0,0,1,1,0.8,0.8\}$. The matrix γ and the warping path $\mathcal{W}$ are depicted in Figure 4.3. In general, a lower value of DTW implies that the input signal $\mathcal{I}$ is very similar to the signature $\mathcal{S}$.

Additionally, from Figure 4.3, the process of generating the matrix γ by using *dynamic programming* approach to find the minimum



	0	0	0	1	1	0.8	0.8
0	0	0	0	1	2	2.8	3.6
0	0	0	0	1	2	2.8	3.6
0	0	0	0	1	2	2.8	3.8
0	0	0	0	1	2	2.8	3.6
1	1	1	1	0	0	0.2	0.4
1	2	2	2	0	0	0.2	0.4
1	3	3	3	0	0	0.2	0.4

Figure 4.3: Distance matrix γ and the warping path $\mathcal{W}$ with $p = 0$.

DTW value can be seen vividly. The matrix is built column by column, from left to right and from top down for each column.

The whole procedure of the detection mechanism inside each deployed router can be stated as follows:

Detection Procedure

Assume the capacity of each input port or output port of the router is $C_{\mathcal{P}}$ and the size of sampled input traffic is m.

1. Sample the incoming traffic of the current input port or output port, call it I_{real};
2. Normalize the throughput: $I_N = \frac{I_{real}}{C_{\mathcal{P}}}$
3. **For** $i = 1$ to m { `/* remove noise */`
   If $(I_N(i) < NoiseThreshold)$
   $I_F(i) = 0$;
   Else $I_F(i) = I_N(i)$;
   }
4. Calculate the auto-correlation of the filtered input $I_A = Auto - correlation(I_F)$
5. Using dynamic programming approach to calculate the DTW value of input signal I_A and the template signal S

$D = DTW(S, I_A)$;
6. **If** ($D <=$ Attack Threshold)
Low-rate TCP Attack = True;
7. **Else** Low-rate TCP Attack = False;

To implement such detection mechanism, one may choose to put the dynamic detection within a router, or outside a router. To put the detection outside a router, one needs to use a signal splitter so that traffics from a port can be copied to a computing node and the computing node can then perform the dynamic detection on a port by port basis. It is important to point out that the computational complexity of the detection algorithm is very low and it can be carried out in a polynomial time using a dynamic programming approach. In particular, for an input size of m and template size of n, the computational complexity of this DTW is $\Theta(mn)$.

4.7 Robustness and Accuracy of DTW

Let us consider the robustness and accuracy of using the DTW method to detect a low-rate TCP attack. The experiment setup is as follows. or the template of low-rate attack signature, we consider $T = 1.2$ sec, $l = 0.2$ sec, $R = 1.0$ and $S = N = 0$. Note that although we choose this signature values as the default template in our experiments of the detection, our methodology is general enough for detecting a large family of attack traffic. For the input traffic, we sample 100 times per second and the sampling duration is three seconds per detection. We set the noise filter threshold $\beta_2 = 0.3$, the maximum average throughput of low-rate attack, so that all background traffic that is less than or equal to 30% of the maximum link capacity C will be clamped to zero. Under the DTW, we set the penalty value $p = 0.01$. We consider the following four types of attack traffic:

- ***Strictly Periodic Square Burst (SPSB):*** a strictly periodic signal with a single burst of length l within a period T. The values of l and T are the same for each period.
- ***Random Periodic Square Burst (RPSB):*** a randomly generated periodic signal with a single burst of length l within a period T. The values of l and T between different periods can be different and they are drawn from a uniform distribution (as described below).
- ***Strictly Periodic General burst (SPGB):*** a strictly periodic signal which is generated by a sine wave with period T with an added random noise N. The values of T and N are the same for each period. In reality, the general burst may not be limited to sine wave, and it can be any periodic burst waveform.
- ***Random Periodic General burst (RPGB):*** a randomly generated periodic sine wave with a period of T and with and added random noise N. The values of T and N are drawn from uniform distributions (as described below) and these values may be different from one period to another period.

4.7.1 DTW values for low-rate attack:

To generate an input traffic, the period T is uniformly distributed within $[1, 1.5]$. The burst length is uniformly distributed within $(0, 0.5]$, The background noise N is uniformly distributed in $[0, 0.5]$, the time shift S is uniformly distributed in $[0, T]$ and the magnitude of the burst is set to $R = 1$. We generate around 3000 samples for each of the four types of input traffic discussed above. The results are illustrated in Table 4.1. From the result, one can observe that a large family of low-rate attack has a DTW value which is less than or equal to 35.66.

Values of DTW	SPSB	RPSB	SPGB	RPGB
Max DTW	34.88	35.66	34.08	34.69
Min DTW	0	0.80	0.84	1.20
Mean DTW	10.68	9.63	10.89	10.48

Table 4.1: DTW values for three types of attack traffic.

4.7.2 DTW values for legitimate traffic (Gaussian):

The detection mechanism must distinguish legitimate traffic from the attack stream so as to avoid possible false positive or false negative alert. Therefore, it is desirable to achieve that the *minimum* DTW value of the legitimate traffic be larger than the *maximum* DTW value of any attack traffic so as to reduce the possibility of false positive/negative during the detection.

We carry out the following experiment on legitimate traffic. Based on our assumption before, if there is no low-rate attack, the TCP flows will not back off, all the traffic including TCP and UDP traffic will go through the router properly. We assume that the normal traffic consists of a major constant throughput with some Gaussian noises. In other words: legitimate traffic $= C_1 + \text{random}[0, N]$, where $C_1 \in [0.3, 1]$ and $N \in [0, 0.5]$. We vary the value of C_1 by a step size of 0.01 and for each value of C_1, we generate around 100 different values of N. The results are depicted in Table 4.2. As one can observe, the minimum DTW value for the Gaussian legitimate traffic is above 110 which is much higher than the maximum DTW value of attack traffic reported above. And Figure 4.4 illustrates the *probability density function* of the DTW values for attack and Gaussian flows respectively. From the figure, we observe that there is a clear gap between the Gaussian legitimate traffic and the low-rate attack traffic. Finding a pint to differentiate between legitimate or attack traffic can be easily carried out.

	Gaussian Traffic
Max DTW	286.53
Min DTW	113.50
Mean DTW	236.95

Table 4.2: DTW values for legitimate traffic (Gaussian).

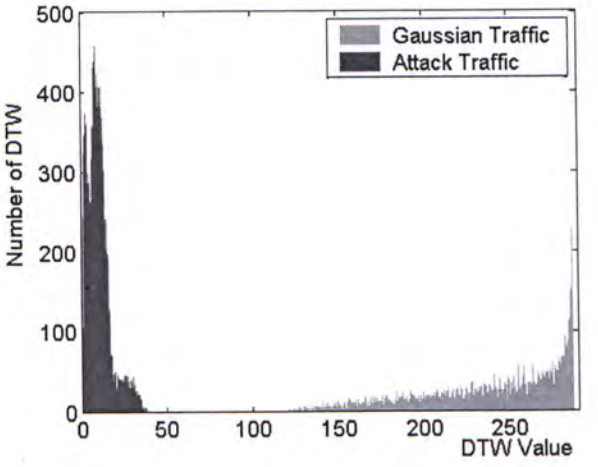


Figure 4.4: Probability density functions of DTW values for the attack and the Gaussian legitimate traffic.

4.7.3 DTW values for legitimate traffic (Self-similar):

As Gaussian traffic may not perfectly represent all legitimate traffics, we also consider using the self-similar Traffic Model to represent legitimate traffic. It is shown that both the Ethernet local area network [32] and the World Wide Web traffic [14] are statistically self-similar.

Self-similar traffic can be described mathematically. Let $X = (X_t, t = 1, 2, 3, ...)$ be a time series with the mean μ and variance σ^2. The limit of the autocorrelation function $r(k) = E[(X_t - \mu)(X_{t+k} - \mu)]/E[(X_t - \mu)^2]$, $(k = 0, 1, 2, ...)$, when k is approaching infinity,

is

$$lim_{k\to\infty} r(k) = k^{-\beta}, \tag{4.5}$$

where $0 < \beta < 1$. For each $m = 1, 2, 3, ...$, there is a new time series $X^{(m)} = (X_t^{(m)}, t = 1, 2, 3, ...)$, which is generated by dividing the original series $X = (X_t, t = 1, 2, 3, ...)$ into m non-overlapping segments, where $X_t^{(m)} = 1/m(X_{tm-m+1} + ... + X_{tm}), t \geq 1$. If the autocorrelation of $X^{(m)}$ has the same structure as that of X, i.e.,

$$r^{(m)}(k) = r(k),$$

then X is said to be (asymptotically) second order self-similar with degree $H = 1 - \beta/2$, where H is called *Hurst Parameter*. Previous works have shown that the Hurst Parameter H for common Internet traffic is around 0.80.

Based on the definition before, we generate a large number of self-similar traffics using the FARIMA model [24, 36]. We generate the self-similar traffic with *Hurst Parameter* H from 0.75 to 0.85 by the step of 0.01 and 1000 samples for each H with the average rate of throughput ranging from 0.05 to 0.95. The results are depicted in Table 4.3 and the *probability density function* of the DTW values for attack and Self-similar flows is illustrated in Figure 4.5.

	Self-similar Traffic
Max DTW	238.16
Min DTW	28.01
Mean DTW	130.73

Table 4.3: DTW values for self-similar legitimate traffic.

One can observe that the minimum DTW value for self-similar legitimate traffic is less than the maximum DTW value of attack traffic before. Therefore, some false positive and false negative may occur during the detection. However, as shown in Figure 4.5, the value of self-similar traffic is mainly distributed from 28 to around

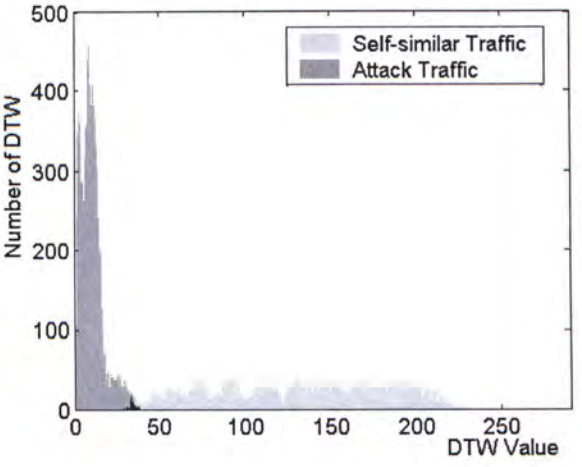


Figure 4.5: Probability density functions of DTW values for the attack and the self-similar legitimate traffic.

238, the intersection area of the attack traffic and the self-similar traffic is rather small compared with both the area of attack traffic and Self-similar traffic separately. Thus the detection mechanism can still be efficient by restricting the false positive and false negative to a small proportion. As depicted in Table 4.4, among 11000 values of self-similar traffic that we generated only 141 of them are less than the maximum DTW value of the attack traffic. Thus the maximum possible false positive is only a around 1% if one sets the attack threshold as the maximum DTW value of attack traffic (i.e., 35.66). Similarly, the maximum possible false negative is around 3.5% if one sets the attack threshold as the minimum DTW value of the self-similar legitimate traffic (i.e., 28.01). In summary, the

False Self-similar	141	**False Attack**	378
Total Self-similar	11000	**Total Attack**	11492
Max False Positive	1.28%	**Max False Negative**	3.54%

Table 4.4: False detection between attack and Self-similar traffic.

proposed detection mechanism can successfully distinguish the attack traffic and legitimate traffic with low false positive and/or false negative.

□ **End of chapter.**

Chapter 5

Low-Rate Attack Defense Mechanism

Summary

This chapter provides the defense solution to the low-rate attack. First, based on the algorithm of Deficit Round Robin, we provide the defense mechanism (Section 5.1). Then theoretical analysis of DRR algorithm is presented in Section 5.2.

5.1 Design of Defense Mechanism

As we discussed in the distributed detection mechanism in Chapter 4, an enabled router $\mathcal{R}$ first determines the existence of low-rate attack on an output port $\mathcal{P}_o$ which it uses to forward packet to a victim site. When a low-rate attack is discovered, $\mathcal{R}$ will then determine the input port that the low-rate attack is coming from. In other word, $\mathcal{R}$ needs to execute the detection algorithm on each of its input port. If the low-rate attack is coming from the input port $\mathcal{P}$, then $\mathcal{R}$ needs to signal all upstream routers which are directly connected to $\mathcal{P}$ to execute the distributed low-rate detection algorithm and execute the

defense mechanism. The motivation of this type of push back is to determine the attack as close to the source as possible. This way, we minimize the number of affected TCP flows.

When a router $\mathcal{R}$ discovers the existence of low-rate attack on its output port but *cannot* discover the existence of low rate attack on any of its input ports, this implies that the attack may be using a distributed approach in launching the low-rate attack, for example, sending a short burst at each input port of $\mathcal{R}$ so that these short bursts will converge to a low-rate attack an the output port of $\mathcal{R}$. Under this scenario, the router $\mathcal{R}$ needs to perform the necessary resource management so as to minimize the damaging effect to TCP flows going through the output port $\mathcal{P}_0$.

In our work, we use the deficit round robin (DRR) algorithm to provide the bandwidth allocation and resource protection. The motivation of using DRR is its near perfect isolation of ill-behaved source at a very low implementation cost.

In our case, instead of classifying packet based on its flow, we classify packet according to the input port of $\mathcal{R}$. Let $\mathcal{P}_i$ denote the set of input ports of the router $\mathcal{R}$ and $|\mathcal{P}_i|$ denote the number of input ports. We have $|\mathcal{P}_i|$ classes and packets coming from input port i which are forwarded to output port $\mathcal{P}_0$ will be classified as class i, where $i = 1, \ldots, |\mathcal{P}_i|$. The DRR assigns a Quantum[i] of service to each class i in each round and attempts to serve packets from each class on a per round basis. Each class has a deficit counter, which is deficit_counter[i] and it is initialized to zero, for $i = 1, \ldots, |\mathcal{P}_i|$. At the beginning of a round, deficit counter of each non-empty class i will be increased by the Quantum[i] value (Usually the values of all Quantum[i] are unique and set as Quantum). A packet from class i will be served if the size of the packet is less than or equal to the value in deficit_counter[i]. When a packet is transmitted from class i, its deficit value will be adjusted by deficit_count[i] -= packet's size. If there is no packet in class i, then we reset the deficit counter as deficit_count[i] = 0. Note that the deficit of the previous rounds

gets carried over to the next round and it is only reset to zero whenever there is no packet in that class.

5.2 Analysis of Deficit Round Robin Algorithm

During the traffic scheduling, although packets from different classes (or input ports) can have different sizes, fairness can still be achieved. As shown in [55,56], the difference in the normalized bytes sent between classes within a certain interval (t_1, t_2) is bounded by a small constant.

We say that class i is *backlogged* during an interval (t_1, t_2) of a DRR execution if the queue for class i is never empty during the interval. We define c_i as the *class share* obtained by the class i that $c_i = \frac{\text{Quantum[i]}}{\text{Quantum}}$ where Quantum $= Min(\text{Quantum[i]})$. Let $sent_i(t_1, t_2)$ be the total number of bytes sent on the output port by class i in the interval (t_1, t_2). Therefore, the measurement of fairness $FM(t_1, t_2)$ can be expressed as the maximum difference in the normalized bytes sent for class i and j during (t_1, t_2):

$$FM(t_1, t_2) = \max\left(sent_i(t_1, t_2)/c_i - sent_j(t_1, t_2)/c_j\right).$$

Lemma 1 For any class i, during the execution of DRR algorithm, the deficit_counter[i] is bounded below by 0 and bounded above by Max, where Max is the maximum packet size of all possible packets. Formally, we have

$$0 \leq \text{deficit\_counter[i]} < Max. \tag{5.1}$$

Proof : Please refer to the appendix. ∎

Lemma 2 During any period in which class i is backlogged, the number of bytes sent on the behalf on class i is bounded by

$$m \cdot \text{Quantum[i]} - Max \leq sent_i(t_1, t_2) \leq m \cdot \text{Quantum[i]} + Max$$

where m is the number of round-robin service opportunities received by class i during this interval.

Proof : Please refer to the appendix. ∎

The above two lemmas provide bounds for deficit_counter[i] and $sent_i(t_1, t_2)$. Now we can provide an upper bound on the fairness measure.

Theorem 1 Under the DRR service discipline, for an interval (t_1, t_2), we have the following fairness measure:

$$FM(t_1, t_2) \leq 2 \cdot Max + \text{Quantum}, \quad where \ \text{Quantum} = Min(\text{Quantum[i]}). \tag{5.2}$$

Proof : Please refer to the appendix. ∎

As a result, it is easy to observe that the fairness between classes achieved using DRR algorithm. Additionally, The DRR algorithm is also known to be efficient and can be easily implemented compared with other scheduling algorithm [50]. In general, the processing cost of DRR is $O(1)$ per packet. As a matter of fact, DRR has already been implemented in some of the Cisco's routers [7].

□ **End of chapter.**

Chapter 6

Fluid Model of TCP Flows

Summary

To estimate the effectiveness of defense mechanism, we develop a fluid model of TCP flows. Based on the model, a set of simulations are designed and results are analyzed.

6.1 Fluid Math. Model of TCP under DRR

To anticipate the effectiveness of DRR, we model the behavior of different TCP flows using Differential Equations. A TCP model with droptail router was first presented based on the fluid model of TCP [39]. We extend the original model with slow start phrase and one second timeout mechanism so as to explore the transient behavior of TCP flows. Then, we came with the model of TCP under DRR scheduling. Simulations were conducted in order to compare the effectiveness of different scheduling mechanism in busy router.

6.1.1 Model of TCP on a Droptail Router

In a congested router with Droptail scheduling algorithm, let N TCP flows labeled $i = 1...N$ traverse the router. For each output interface, there is a common droptail queue which starts to drop packets

when the queue is full. The dropping function of the droptail queue can be written like:

$$p(x_i) = \begin{cases} 0, & x_i \leq Q_i \\ 1, & x_i > Q_i \end{cases} \tag{6.1}$$

where Q_i is the size of queue i in the router and x_i is the length of queue i. Usually, the size of queues in a router are the same Q.

Base on the original fluid model [39], the queue length x_i will change as follows:

$$\frac{dx_i}{dt} = \sum_{i=1}^{N} \rho_i(t) \cdot [1 - p(x_i(t))] - 1_{x_i(t)} C \tag{6.2}$$

The first term demotes the increase of queue length due to all the incoming traffic from N TCP flows destined to current output queue. And the second term represents the packets sending rate of the queue.

For the Fluid TCP model, as shown in [39], the instantaneous throughput of TCP flow i is:

$$\rho_i(t) = \frac{W_i(t)}{R_i(t)} \tag{6.3}$$

Here $W_i(t)$ and $R_i(t)$ denote the TCP window size and round trip time at $t \geq 0$, of flow i,$i = 1, ...N$. In a router with Droptail scheduling, the estimated round trip time of past TCP flows should take from:

$$R_i(t) = a_i + \frac{x_i(t)}{C} \tag{6.4}$$

where a_i is the fixed propagation delay and $x_i(t)/C$ approximates the queuing delay with the capacity C of the current queue.

For the behavior of the window size $W_i(t)$ for flow i, from [39], we have the following equation:

$$\begin{aligned}\frac{dW_i(t)}{dt} &= \frac{1}{R_i(t)} \\ &-[1-q(W_i(t))]\frac{W_i(t)W_i(t-t')}{2R_i(t-t')}p(x(t-t')) \\ &+q(W_i(t))(1-W_i(t))\frac{W_i(t-t')}{R_i(t-t')}p(x(t-t'))\end{aligned} \tag{6.5}$$

Let $q(W_i(t))$ denote the probability that the loss is caused by a Timeout, given the window size at the time of loss $W_i(t)$. As in [39, 45], we use the simplified function $q(W_i(t)) = Min(1, 3/W_i(t))$ to express $q(W_i(t))$. Intuitively, this expression for Q is based on the assumption that all packets in a particular round are equally likely to be dropped, with at most one drop per round. In that case, any one of the last 3 packets in a round can cause a timeout if dropped. The first term on the right of Equation 6.5 describes the congestion avoidance in which the TCP window size will increase by one per round trip time and the last two terms express the multiple decrease of window size when TCP receives triple duplicated ACKs or incurs timeout.

To add the slow start phrase of TCP to the model, another term , the threshold H, should be imported. The threshold will remain the same until loss occurs. In case of loss, the threshold becomes half of the current window size.

$$H_i(t) = H_i(t-t')[1-p(x_i(t-t'))] + \frac{W_i(t)}{2}p(x_i(t-t')) \tag{6.6}$$

TCP stays in the slow start phrase in which windows size doubles per round trip time until its window size $W_i(t)$ exceeds the threshold H_i. Thus by using the unit step function (6.7),

$$u(n_i) = \begin{cases} 0, & n_i < 0 \\ 1, & n_i \geq 0 \end{cases} \tag{6.7}$$

we can add slow-start phrase to equation (6.5):

$$\frac{dW_i(t)}{dt} = \frac{W_i(t)}{R_i(t)}u[H_i(t) - W_i(t)] + \frac{1}{R_i(t)}u[W_i(t) - H_i(t)] \quad (6.8)$$
$$-[1 - q(W_i(t))]\frac{W_i(t)W_i(t-t')}{2R_i(t-t')}p_i(x(t-t'))$$
$$+q(W_i(t))(1 - W_i(t))\frac{W_i(t-t')}{R_i(t-t')}p_i(x(t-t'))$$

To model the detail transient behavior of TCP, it is necessary to include the timeout feature. As we know, in a congested router, when loss occurs and TCP detects loss via the timeout event, TCP will decrease the window size to one, enter the slow start phrase and retransmit the lost packet with a timer initialized to be the default timeout value [8]. If this packet gets lost again, TCP will keep silent until the timer expires. In our model we assume that in a busy router with both TCP and UDP traffic, once the traffic is congested and loss happens, TCP flow will incur multiple continuous losses and thus will keep silent at least for the default timeout. Also we bypass the exponential back off of TCP in our model and use the default lower bound timeout value 1 second as recommended in [8] constantly to approximate the TCP timeout feature. We use T_i to denote the start time of latest continuous losses of flow i. And T_i will be updated by the current time t when timeout happens and TCP is in 'non-silent' state. As shown in Equation (6.9):

$$\frac{dT_i(t)}{dt} = (t - T_i(t))p(x(t-t'))q(W_i(t))u[t - (1 + T_i(t))] \quad (6.9)$$

step function $u[t - (1 + T_i(t))]$ express the state if TCP is silent due to the timeout and $p(x(t-t'))q(W_i(t))$ gives the probability of loss detected via timeout.

And finally, the behavior of TCP window size can be modified

into Equation (6.10)

$$\frac{dW_i(t)}{dt} = (\frac{W_i(t)}{R_i(t)}u[H_i(t)-W_i(t)]+\frac{1}{R_i(t)}u[W_i(t)-H_i(t)])\cdot u[t-(1+T_i(t)] \tag{6.10}$$
$$-[1-q(W_i(t))]\frac{W_i(t)W_i(t-t')}{2R_i(t-t')}p_i(x(t-t'))$$
$$+q(W_i(t))(1-W_i(t))\frac{W_i(t-t')}{R_i(t-t')}p_i(x(t-t'))$$

Now we have a group of differential equations (6.1),(6.2),(6.3), (6.4),(6.6),(6.7),(6.9),(6.10) to describe the behavior of TCP flows under Droptail scheduling. By solving these equations numerically, we could get a meaningful overview of TCP droptail scheduling.

6.1.2 Model of TCP on a DRR Router

Our model of DRR scheduling of TCP will mainly derived from the Droptail model above. Thus the model is divided into two parts: DRR Queue management in the router and TCP throughput adaptation.

DRR Queue management: When to drop a packet in a DRR queue? Actually DRR works similar a drop tail router which start to drop packet when the queue is full. So the dropping function of DRR is the same as function (6.1).

We model the change of the queue length x_i per round. Thus $\Delta t = \tau$, τ is the time length of each round. Here we use the time label t to denote the very round which is running at time t. (It starts at t_1 and ends at t_2, $t_1 \leq t \leq t_2$ and $t_2 - t_1 = \tau$.)

We draw the model based on the following assumptions:

1. During each round t assume the dropping probabilities of all coming packets of flow i are the same $p(x_i(t))$. (They may not be the same in reality especially when the queue size is not large enough or in case of high throughput.)

2. All the coming packets during one round just come at t^- (Actually should be denoted as t_1^-), the moment before the current round DRR scheduling starts.

During the DRR algorithm, the queue length x_i will change as follows:

$$\frac{dx_i}{dt} = \rho_i(t) \cdot [1 - p(x_i(t))] - 1_{x_i(t)} \frac{Min(Quantum_i, x_i(t))}{\tau_t} \quad (6.11)$$

The first term is the rate of the queue length increase which is the incoming throughput $\rho_i(t)$ of flow i by the probability of dropping no packets in this round $1 - p(x_i(t))$, since we assume the identical dropping probability during each round. And the second term represents packets sending rate in $bits/s$ during round t, where the length of this round is τ_t. And since in a busy router with DRR queues, traffic sent for each queue approximate $Quantum$ in the long run, in our model, we skip the detail of deficit count accumulation between different rounds and use this general approximation instead.

And the time length for each round τ_t is just the sum of time sending a certain traffic of each queue. Thus, we have:

$$\tau_t = \frac{\sum_i^N Min(Quantum_i, x_i(t))}{C} \quad (6.12)$$

where C is the capacity in $bits/s$ of the busy router.

TCP throughput adaption: For the TCP throughput adaption in DRR scheduling, the estimated round trip time of past TCP flows should be modified as follows:

$$R_i(t) = a_i + \frac{N \cdot x_i(t)}{C} \quad (6.13)$$

where a_i is the fixed propagation delay and $N \cdot x_i(t)/C$ approximates the queuing delay when N flows share the capacity C of the busy router.

From our model of Droptail, it is easy to see that, the regulations of TCP behavior under DRR scheduling are almost the same

as under Droptail scheduling except the behavior of queue length and calculation of RTT shown above. Thus, by combining these equations:(6.1),(6.11),(6.12),(6.3),(6.13),(6.6),(6.7), (6.9),(6.10), we come with the fluid model of TCP on a DRR router. Solving the group of differential equations, we will get the numerical description of the TCP behavior under the DRR scheduling.

6.2 Simulation of TCP Fluid Model

To test the effectiveness of the fluid model, we designed two groups of simulations: Attack with single TCP flow and attack with multiple TCP flows on both droptail router and DRR router.

6.2.1 Simulation of Attack with Single TCP Flow

The first group of simulation is described in Figure 6.1 and Figure 6.2. We designed a single TCP flow with a single low-rate attack passing through the same congested router. The router may use droptail queues or DRR queues. The settings of simulation are the follows: The low-rate attack is a square burst with $T = 1.1s$, burst length $l = 0.1s$ rate of burst $R = 300kb/s$ which contribute to the average throughput of less than $30kb/s$. It will start $2s$ after the beginning of the simulation. The capacity of the router is $100kb/s$ all together. And we assume the propagation delay is $0.1s$ for the single TCP flow. Under the DRR scheduling, we set the quantum of each round $1kb$ and the buffer size approximates $10kb$.

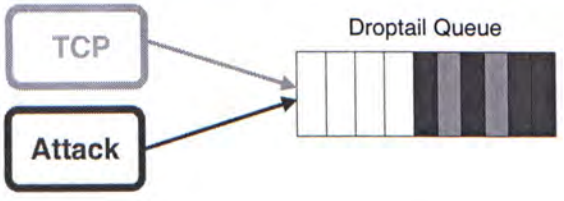


Figure 6.1: Attack with single TCP flow on Droptail router

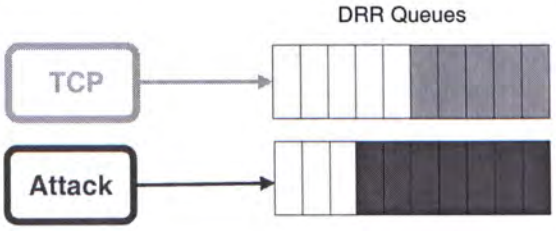


Figure 6.2: Attack with single TCP flow on DRR router

The results are shown in Figure 6.3 and Figure 6.4. As depicted in Figure 6.3, on the droptail router, the TCP flow enjoys a full bandwidth (around $100kb/s$) during the first two seconds before the attack appears. While in case of the low-rate attack, TCP flow only obtains less than $20kb/s$ throughput and around 50% of the bandwidth has not been used.

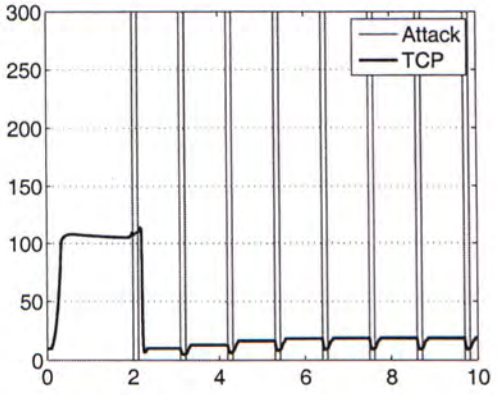


Figure 6.3: Result of attack with single TCP flow on Droptail router

And on the router which adopts DRR scheduling, TCP flow is protected from the low-rate attack. In Figure 6.4, the throughput of TCP flow only decreases a little at the moment of burst and TCP has never entered the silent timeout period. From the figure, we can see that the TCP flow obtains around 70% of the capacity on average during the attack.

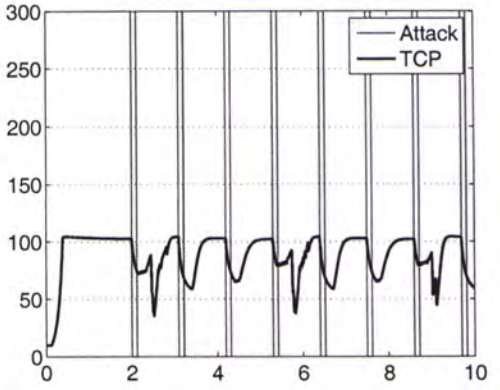


Figure 6.4: Result of attack with single TCP flow on DRR router

6.2.2 Simulation of Attack with Multiple TCP flows

In the second group of simulations, we design three scenarios which focus on the effectiveness of different scheduling method to multiple TCP flows. First we arrange four unsynchronized TCP flows with one single low-rate attack passing through one congested router. The common settings of these simulations remain the same as before.

Case 1: In the first case (Figure 6.5), all the four TCP flows share the same droptail queue with the low-rate attack in the router. The propagation delay for the unsynchronized TCP flows are $0.1s$, $0.2s$, $0.4s$ and $0.8s$ each.

The result is shown in Figure 6.6. Although the burst of attack is $300kb/s$, we plot the figure with Y-axis from 0 to 100 for clear exhibition of four TCP flows,(and so do the next two plots of result). As we can see, before the attack, all four TCP flows share the capacity of the link proportional to their round trip time. TCP1 with the minimum delay enjoys the largest throughput. In case of low-rate attack, almost all the TCP flows are affected. The throughput of TCP1 decreases sharply from around $80kb/s$ to around $20kb/s$. TCP1 is the most responsive due to the shortest latency and TCP4 has been affected least due to its long propagation delay. On average

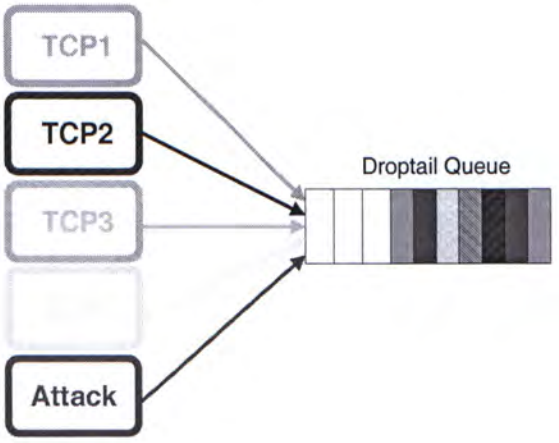


Figure 6.5: Attack with multiple TCP flows on Droptail router

there are 40% of capacity wasted.

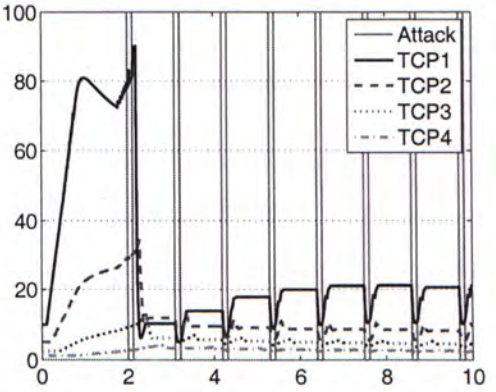


Figure 6.6: Result of attack with multiple TCP flows on Droptail router

Case 2: In the second scenario (Figure 6.7), each of the four TCP flows and the low-rate attack is assigned to one DRR queue in the router and share the capacity of $100kb/s$ together. The propagation delay for TCP flows are the same as the first case.

From Figure 6.8, we can see that DRR has successfully isolated the attack traffic. None of the TCP flows have been affected by the attack. Another important discovery is that fairness is obtained

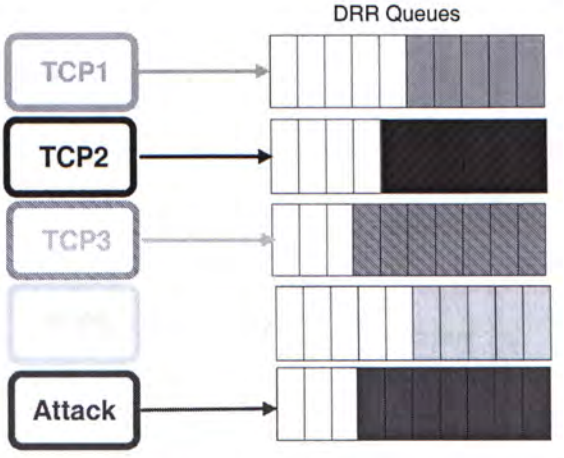


Figure 6.7: Attack with multiple TCP flows on DRR router

among the five flows. As time goes by, the throughput of each TCP flow converges to around $20kb/s$ which is one fifth of the capacity.

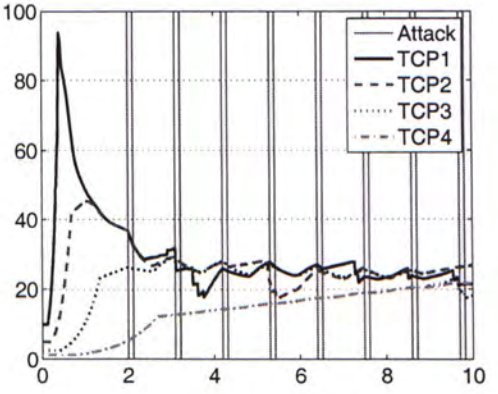


Figure 6.8: Result of attack with multiple TCP flows on DRR router

Case 3: We have designed another comprehensive scenario, as shown in Figure 6.10, three TCP flows come from one input interface of the router, and the other TCP flow comes from another input interface with the low-rate attack. Each interface possesses a DRR queue. The attack settings and router settings are the same as before. The propagation delay for the first three TCP flows are $0.1s$, $0.2s$ and

$0.4s$. TCP4 has a $0.1s$ propagation delay.

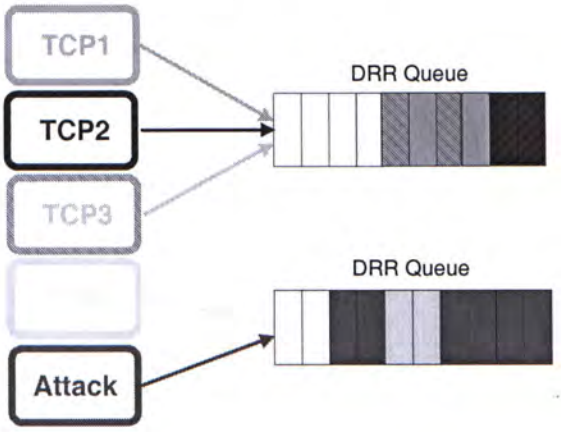


Figure 6.9: Attack with multiple TCP flows on DRR router

From the result (Figure 6.10), we can see that, DRR scheduling provides excellent fair allocation of resource between different queues. The first three TCP flows have not suffered from the attack and they share more than half of the capacity proportional to their latencies. While the throughput of TCP4 degenerate a lot after the start of low-rate attack. So if we want to protect TCP4, defending mechanism need to be pushed back to upstream routers.

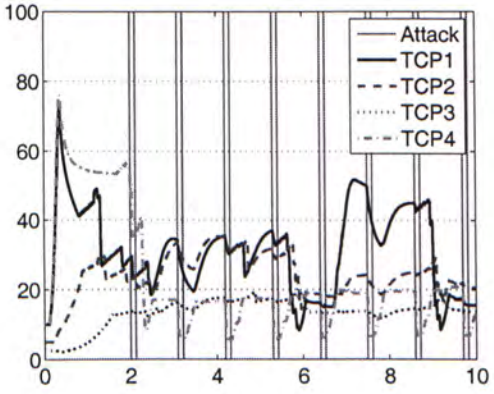


Figure 6.10: Result of attack with multiple TCP flows on DRR router

□ **End of chapter.**

Chapter 7

Experiments

Summary

In this chapter, we carry out experiments using NS-2 to determine the effectiveness of the proposed defense mechanism.

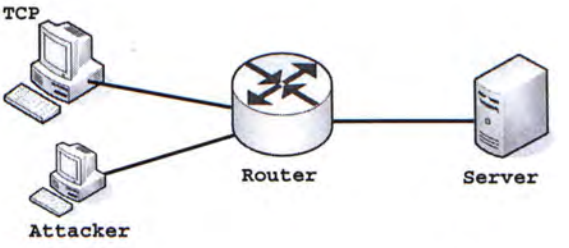


Figure 7.1: Single low-rate attack and single TCP flow.

7.1 Experiment 1 (Single TCP flow vs. single source attack)

The first experiment is depicted in Figure 7. We consider a single low-rate TCP attack and a single TCP flow going through the same router. The latency of each link is 5ms, with the minimum Round

Trip Time (RTT) being 20 ms. And the capacity of each link is 5 Mbps. The low-rate attack is a square burst with $T = 1.0$ sec, burst length $l = 0.2$ sec, burst rate of 5 Mbps or $R = 1$. The low-rate attack uses UDP with packet size of 100 bytes. The packet size of the TCP flow is 500 bytes. Under the DRR, we set the quantum size of each round to be 500 bytes and the buffer size is 5000 bytes. The result is illustrated in Table 7.1 and Figure 7.1. Note that without the defense mechanism, the router simply uses the conventional scheduling (e.g., drop tail or FCFS) to handle packets. And we observe that the TCP flow can only utilize around 4% of the link's bandwidth. On the other hand, when one uses the DRR, we observe an improvement in the TCP's throughput from 224.37 Kbps to 3.402 Mbps, or an improvement from 4.49% to 68.04% of the link capacity.

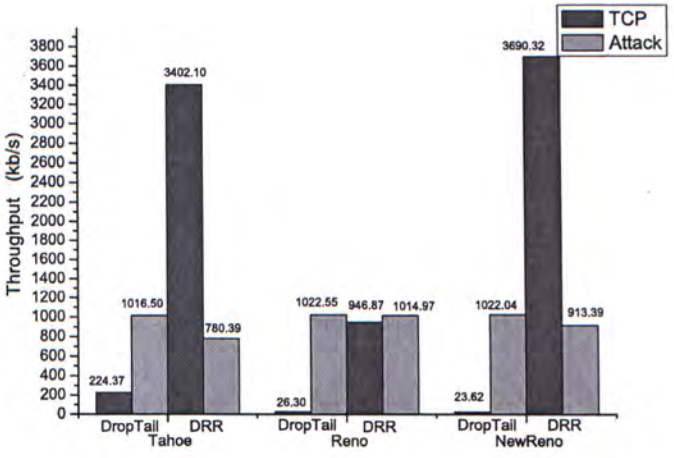


Figure 7.2: Result for Low-rate Attack to Single TCP Flow using Tahoe, Reno and New Reno

Table 7.2 and 7.3 depict the same performance when we use TCP Reno and new Reno. One may observe that it is not quite effective for TCP Reno, as the throughput can only be increased to less than 20% when DRR is adopted. This is due to the congestion con-

trol algorithm of TCP Reno which will have a performance problem when multiple packets are dropped from one transmission window. As mentioned in [18], when TCP Reno incurs multiple packets drop, although it can retransmit the first lost packet after receiving three duplicated ACKs, it is unable to employ Fast Retransmit again and must instead await a retransmission timeout which will then put the sender into the Slow-Start phase. Therefore the DRR will not achieve a good performance for TCP Reno in case there are multiple packets dropped. One possible solution is to increase the DRR buffer. As shown in Table 7.4, we repeat the experiment with different sizes of DRR buffer while all other parameters remain the same. One can observe that the throughput gradually increased to about 85% when the buffer is 30000 bytes.

The result shows the effectiveness of the defense mechanism to protect the TCP flows from the ill-behaved attacking flow.

	TCP		**Attack flow**	
	throughput (Kbps)	% of link capacity	throughput (Kbps)	% of capacity
Drop tail	224.37	4.49%	1016.5	20.33%
DRR	3402.10	68.04%	780.39	15.61%

Table 7.1: Result for Low-rate Attack to Single TCP Flow using Tahoe.

	TCP		**Attack flow**	
	throughput (Kbps)	% of link capacity	throughput (Kbps)	% of capacity
Drop tail	26.30	0.53%	1022.55	20.45%
DRR	946.87	18.94%	1014.97	20.30%

Table 7.2: Result for Low-rate Attack to Single TCP Flow using Reno.

	TCP		Attack flow	
	throughput (Kbps)	% of capacity	throughput (Kbps)	% of capacity
Drop tail	23.62	0.47%	1022.04	20.44%
DRR	3690.32	73.81%	913.39	18.27%

Table 7.3: Result for Low-rate Attack to Single TCP Flow using New Reno.

	TCP		Attack flow	
(Bytes)	throughput (Kbps)	% of link capacity	throughput (Kbps)	% of capacity
5000	946.87	18.94%	1014.97	20.30%
15000	1786.92	35.74%	1000.67	20.01%
30000	4286.68	85.73%	656.26	13.13%

Table 7.4: Result for Reno TCP Flow with different DRR Buffer size.

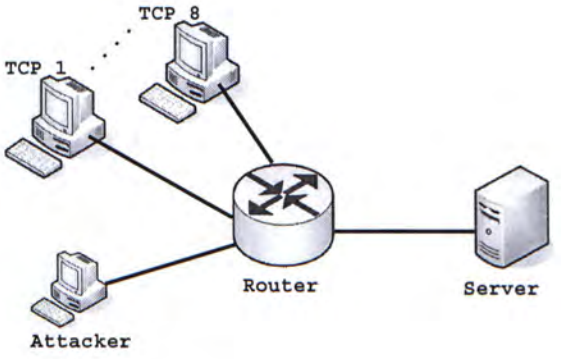


Figure 7.3: Single low-rate attack and Multiple TCP flows.

7.2 Experiment 2 (Multiple TCP flows vs. single source attack)

The second experiment is depicted in Figure 7.3. We consider a single low-rate TCP attack and 8 TCP flows going through the same router. Parameters are the same as Experiment 1 except that the

buffer size of the DRR-enabled router is 12.5 Kbytes. So the minimum RTT remains the same as 20 ms while the upper bound of RTT is increased to 25 ms. The result is illustrated in Table 7.5 and Figure 7.4. Again, using the conventional drop tail scheduling, the total TCP bandwidth is only around 8% of the link's bandwidth. When one uses the DRR, we can improve the throughput of all TCP flows from 423.92 Kbps to 4.390 Mbps, or an improvement from 8.48% to 87.80% of the link capacity. From Figure 7.4, it is easy to see that flow TCP 4 gains more average throughput than others on the drop tail router. The reason is that TCP 4 has not been completely synchronized by the low-rate attack, and can still transmit several packets during some silent periods between bursts. Table 7.6, 7.7 and Figure 7.5, 7.6 depict the same performance gain when we use TCP Reno and new Reno. This shows the effectiveness of the defense mechanism.

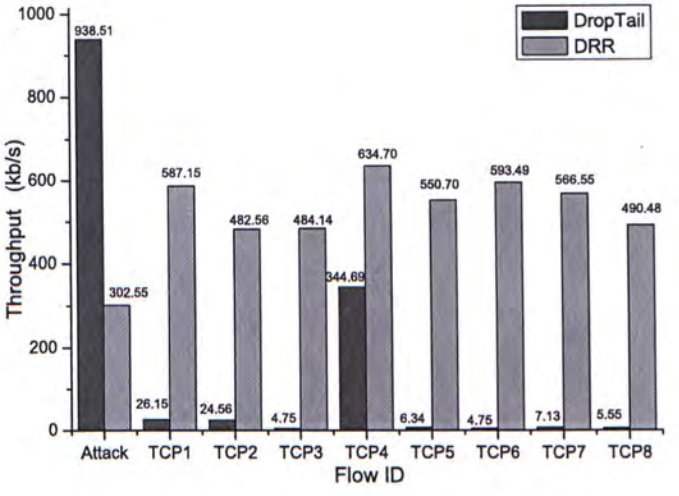


Figure 7.4: Result for Single Low-rate Attack to Multiple TCP Flows using Tahoe.

	Drop Tail		DRR	
	throughput (Kbps)	% of link capacity	throughput (Kbps)	% of link capacity
Attack flow	938.51	18.77%	302.55	6.05%
TCP 1	26.15	0.52%	587.15	11.74%
TCP 2	24.56	0.49%	482.56	9.65%
TCP 3	4.75	0.10%	484.14	9.68%
TCP 4	344.69	6.89%	634.70	12.69%
TCP 5	6.34	0.13%	550.7	11.01%
TCP 6	4.75	0.10%	593.49	11.87%
TCP 7	7.13	0.14%	566.55	11.33%
TCP 8	5.55	0.11%	490.48	9.81%
Total TCP	**423.92**	**8.48%**	**4389.77**	**87.80%**

Table 7.5: Result for Single Low-rate Attack to Multiple TCP Flows using Tahoe.

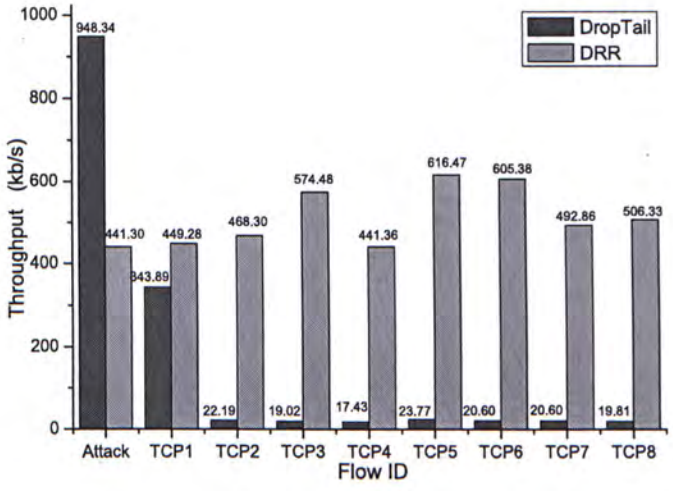


Figure 7.5: Result for Single Low-rate Attack to Multiple TCP Flows using Reno

7.3 Experiment 3 (Multiple TCP flows vs. synchronized distributed low-rate attack)

The third experiment is depicted in Figure 7.7. We consider a distributed low-rate TCP attack and 8 TCP flows going through the

	Drop Tail		DRR	
	throughput (Kbps)	% of link capacity	throughput (Kbps)	% of link capacity
Attack flow	948.34	18.97%	441.30	8.83%
TCP 1	343.89	6.88%	449.28	8.99%
TCP 2	22.19	0.44%	468.30	9.37%
TCP 3	19.02	0.38%	574.48	11.49%
TCP 4	17.43	0.35%	441.36	8.83%
TCP 5	23.77	0.48%	616.47	12.33%
TCP 6	20.60	0.41%	605.38	12.11%
TCP 7	20.60	0.41%	492.86	9.86%
TCP 8	19.81	0.40%	506.33	10.13%
Total TCP	**487.31**	**9.75%**	**4154.45**	**83.09%**

Table 7.6: Result for Single Low-rate Attack to Multiple TCP Flows Using Reno

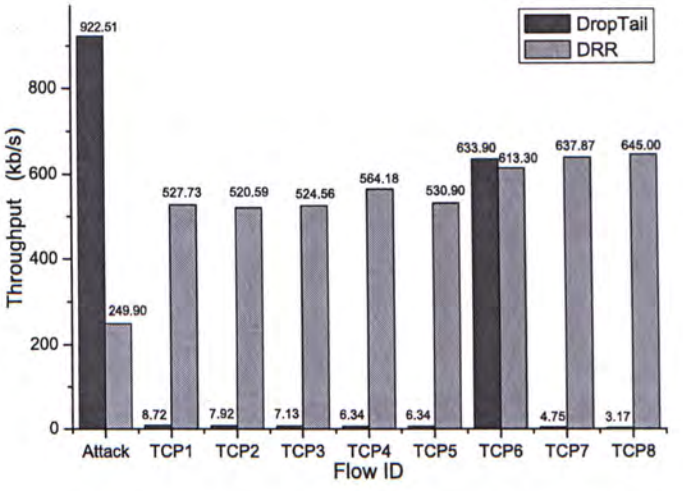


Figure 7.6: Result for Single Low-rate Attack to Multiple TCP Flows using New Reno

same router. Parameters are the same as Experiment 2 except we replace a single attacker by three distributed attackers. Each attacker

	Drop Tail		DRR	
	throughput (Kbps)	% of link capacity	throughput (Kbps)	% of link capacity
Attack flow	922.51	18.45 %	249.90	5.00 %
TCP 1	8.72	0.17 %	527.73	10.55 %
TCP 2	7.92	0.16 %	520.59	10.41 %
TCP 3	7.13	0.14 %	524.56	10.49 %
TCP 4	6.34	0.13 %	564.18	11.28 %
TCP 5	6.34	0.13 %	530.90	10.62 %
TCP 6	633.90	12.68 %	613.30	12.27 %
TCP 7	4.75	0.10 %	637.87	12.76 %
TCP 8	3.17	0.06 %	645.00	12.90 %
Total TCP	**678.28**	**13.57%**	**4564.11**	**91.28%**

Table 7.7: Result for Single Low-rate Attack to Multiple TCP Flows Using New Reno

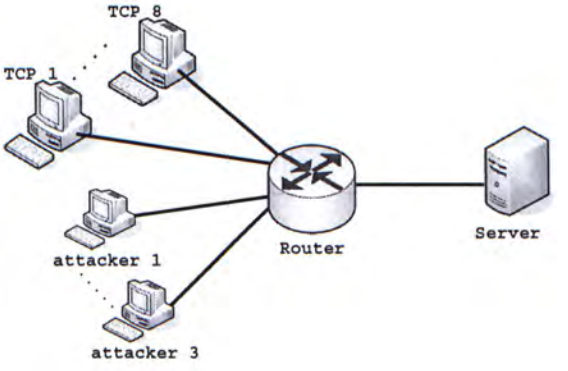


Figure 7.7: Distributed low-rate attack and Multiple TCP flows.

sends a periodic attack burst every $T = 3.0$ seconds. The i^{th} attacker sends a burst with $R = 1$ during the i^{th} sub-period so that the converged attack becomes a low-rate attack with period $T = 1.0$ sec. The result is illustrated in Table 7.1 and Figure 7.8. One can observe that with DRR, we can improve the throughput of all TCP flows from 469.67 Kbps to 4.296 Mbps, or an improvement from

9.39% to 85.94% of the link capacity. Table 7.9, 7.10 and Figure 7.9, 7.10 depict the result when we use TCP Reno and TCP new Reno respectively. Similar observation can be made and this shows the effectiveness of the defense mechanism.

	Drop Tail		DRR	
	throughput (Kbps)	% of link capacity	throughput (Kbps)	% of link capacity
Attack 1	355.30	7.11%	120.98	2.42%
Attack 2	305.91	6.12%	116.34	2.33%
Attack 3	309.63	6.19%	111.17	2.22%
TCP 1	218.47	4.37%	818.58	16.37%
TCP 2	16.64	0.33%	748.80	14.98%
TCP 3	9.13	0.18%	748.80	14.98%
TCP 4	6.98	0.14%	489.54	9.79%
TCP 5	6.44	0.13%	436.93	8.74%
TCP 6	203.97	4.08%	432.64	8.65%
TCP 7	5.36	0.11%	314.55	6.29%
TCP 8	2.68	0.05%	307.03	6.14%
Total attack	**970.84**	**19.42%**	**348.49**	**6.97%**
Total TCP	**469.67**	**9.39%**	**4296.87**	**85.94%**

Table 7.8: Result for Synchronized Distribute Low-rate Attack to Multiple TCP Flows using Tahoe.

7.4 Experiment 4 (Network model of low-rate attack vs. Multiple TCP flows)

The fourth experiment is depicted in Figure 7.11. The transmission bandwidth of all links is 5 Mbps and the propagation delay is 5 ms. Thus, the minimum RTT for TCP1,TCP2 and the attacker is 50 ms and the RTT for TCP3 and TCP4 are 40 ms and 30 ms respectively. The attacker is located at router R_1 and it sends a periodic attack

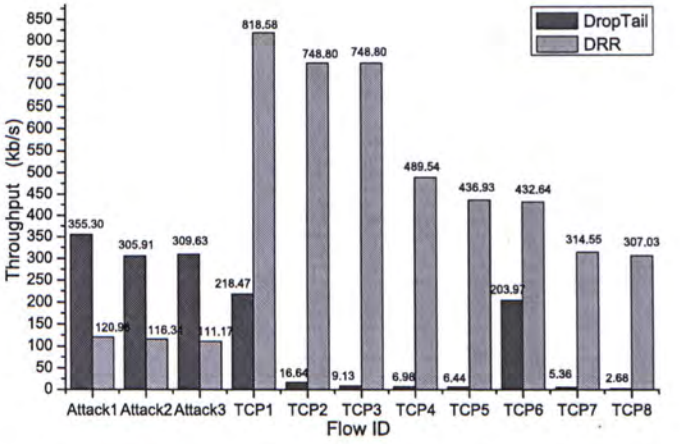


Figure 7.8: Result for Synchronized Distribute Low-rate Attack to Multiple TCP Flows using Tahoe .

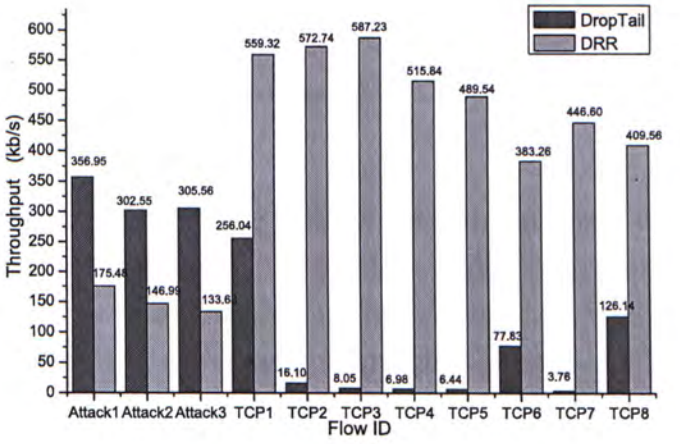


Figure 7.9: Result for Synchronized Distribute Low-rate Attack to Multiple TCP Flows using Reno

with $T = 1$ sec, $l = 0.2$ sec and $R = 1$. There are four TCP flows, TCP 1 is attached to R_1, TCP 2 is attach to R_3, TCP 3 is attached to R_5 and the TCP 4 is attached to R_7. All of them try to upload

	Drop Tail		**DRR**	
	throughput (Kbps)	% of link capacity	throughput (Kbps)	% of link capacity
Attack 1	356.95	7.14 %	175.48	3.51 %
Attack 2	302.55	6.05 %	146.99	2.94 %
Attack 3	305.65	6.11 %	133.68	2.67 %
TCP 1	256.04	5.12 %	559.32	11.19 %
TCP 2	16.10	0.32 %	572.74	11.45 %
TCP 3	8.05	0.16 %	587.23	11.74 %
TCP 4	6.98	0.14 %	515.84	10.32 %
TCP 5	6.44	0.13 %	489.54	9.79 %
TCP 6	77.83	1.56 %	383.26	7.67 %
TCP 7	3.76	0.08 %	446.60	8.93 %
TCP 8	126.14	2.52 %	409.56	8.19 %
Total attack	**965.16**	**19.30%**	**456.15**	**9.12%**
Total TCP	**501.35**	**10.03%**	**3964.08**	**79.28%**

Table 7.9: Result for Synchronized Distribute Low-rate Attack to Multiple TCP Flows Using Reno

files to the server. Table 7.11 shows the throughput of attack and TCP flows when no defense mechanism is deployed (under the drop tail column), as well as the throughput of various flows when DRR is employed at different routing elements. The table shows that enabling the DRR at different routing elements will achieve different TCP throughput. In particular, when DRR is enabled in R_6 only, the bandwidth of TCP 4 is approximately equal to the sum of bandwidth of all upstream flows (e.g., TCP 1 to TCP 3 and the attack traffic). Under the proposed distributed defense mechanism, routers R_1, R_2, R_4 and R_6 will discover the presence of low-rate attack and they will enable the DRR scheduling. One can observe fairness is achieved wherein all TCP flows will achieve approximately the same amount of bandwidth and they are protected and isolated from the

	Drop Tail		**DRR**	
	throughput (Kbps)	% of link capacity	throughput (Kbps)	% of link capacity
Attack 1	354.84	7.10 %	124.65	2.49 %
Attack 2	302.25	6.04 %	105.81	2.12 %
Attack 3	314.32	6.29 %	90.68	1.81 %
TCP 1	23.62	0.47 %	859.38	17.19 %
TCP 2	12.35	0.25 %	573.27	11.47 %
TCP 3	10.20	0.20 %	672.04	13.44 %
TCP 4	152.98	3.06 %	626.95	12.54 %
TCP 5	59.58	1.19 %	461.63	9.23 %
TCP 6	228.67	4.57 %	462.70	9.25 %
TCP 7	2.15	0.04 %	440.15	8.80 %
TCP 8	3.22	0.06 %	385.94	7.72 %
Total attack	**971.41**	**19.43%**	**321.14**	**6.42%**
Total TCP	**492.76**	**9.86 %**	**4482.06**	**89.64%**

Table 7.10: Result for Synchronized Distribute Low-rate Attack to Multiple TCP Flows Using New Reno

ill-behaved attack flow.

Lastly, we like to comment about the practicality of the proposed method. The purpose of our proposed methodology is to provide a practical solution for detecting and defending against the low-rate attack. Consider a victimized core router with 10 interface cards. Although the low-rate attack will converge to one interface card (in which the victim site is attached to that interface card), by performing our defending mechanism, at least 90% TCP flows to the victim site will be isolated from the attack traffic. Additionally, with the cooperation of routers, the pushback mechanism [66] can successfully push the detection and protection as close to the source as possible. Thus more TCP flows will get protected.

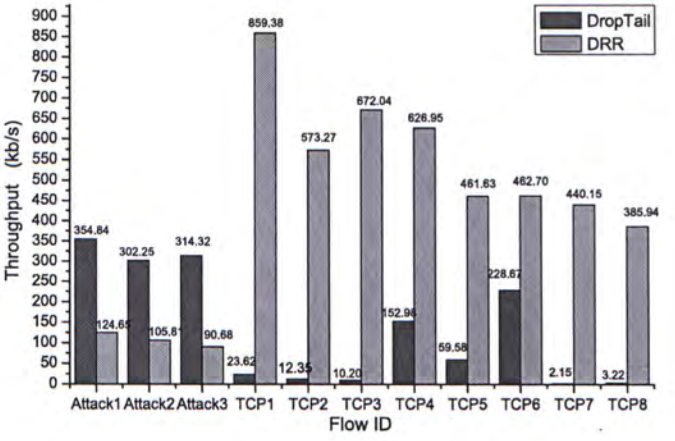


Figure 7.10: Result for Synchronized Distribute Low-rate Attack to Multiple TCP Flows using New Reno

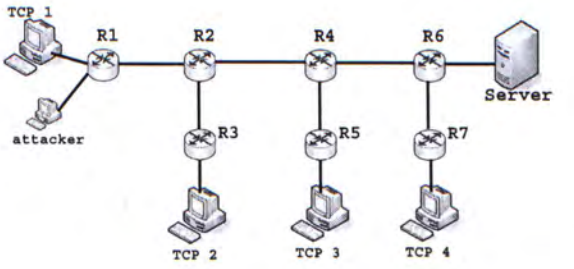


Figure 7.11: Network model of Low-rate attack and Multiple TCP flows .

□ End of chapter.

	Drop tail	**DRR on** R_6	**DRR on** R_6, R_4	**DRR on** R_6, R_4 R_2	**DRR on** R_6, R_4 R_2, R_1
	throughput (kbps)	**throughput** (kbps)	**throughput** (kbps)	**throughput** (kbps)	**throughput** (kbps)
Attack	640.00	561.00	453.00	419.00	404.00
TCP 1	386.00	358.00	311.00	314.00	778.00
TCP 2	264.00	329.00	282.00	874.00	763.00
TCP 3	324.00	251.00	1,245.00	924.00	788.00
TCP 4	425.00	1,719.00	1,154.00	966.00	765.00
Total TCP	1,399.00	2,657.00	2,992.00	3,078.00	3,094.00

Table 7.11: Throughput of various TCP flows when different routers enabled the defense mechanism.

Chapter 8

Conclusion

Summary

This chapter concludes the work we have done in this thesis.

In this thesis, we present a distributed and efficient approach to dynamically detect and defend against low-rate TCP attacks. We present a formal model to describe a large family of low-rate TCP attack patterns, and then we propose a distributed detection mechanism which uses the dynamic time warping algorithm to compare the feature of the sampled input with the signature of the low-rate attack. We show that the detection mechanism is robust and accurate in identifying the existence of low-rate attack. In particular, one can achieve very low false positive/negative when compare to legitimate Internet traffics. When the low-rate attack is present, we use a push back mechanism so as to identify the attack as close to the attack source as possible. The rationale of this push back is to minimize the number of affected TCP flows. We show that one can use the deficit round-robin (DRR) approach to protect the TCP flows and isolate them from the attack traffic. Fluid model of TCP are developed to precisely describe the behavior of TCP flows under DRR approach. Experiments are carried out to quantify the robustness and accuracy of the proposed detection. Extensive simulations are

carried to quantify the merits and effectiveness of the proposed defense mechanism.

□ **End of chapter.**

Appendix A

Lemmas and Theorem Derivation

Summary

Here are the proof of all the lemmas and theorem presented in Chapter 5.

Note: The lemmas, theorem and their proof are derived based on the work in [56].

Lemma 1: *For any class i, during the execution of DRR algorithm, the* deficit_counter[i] *is bounded below by* 0 *and bounded above by* Max*, where* Max *is the maximum packet size of all possible packets. Formally, we have*

$$0 \leq \text{deficit\_counter[i]} < Max.$$

Proof : At the beginning of the algorithm, we set deficit_counter[i] $=$ 0, which is obviously less than Max. At the end of the service round of class i, we need to consider two cases:

1. If there is a packet left in the queue of class i, then its size is greater than deficit_counter[i]. Since the size of any packet is no more than Max, we have deficit_counter[i] $< Max$ and deficit_counter[i] > 0.

2. If the queue of class i is empty, then deficit_counter[i] is reset to zero. ∎

Lemma 2: *During any period in which class i is backlogged, the number of bytes sent on the behalf on class i is bounded by*

$$m \cdot \text{Quantum[i]} - Max \leq sent_i(t_1, t_2) \leq m \cdot \text{Quantum[i]} + Max$$

where m is the number of round-robin service opportunities received by class i during this interval.

Proof : Let use use the term "*round*" to denote service opportunities received by class i within an interval (t_1, t_2). We number these rounds from 1 to round m. With loss of generality, we treat t_1, the start of an interval, as the end of round 0. Define deficit_counter[i][k] as the value of deficit_counter[i] at the end of round k. We also define $bytes_i(k)$ as the bytes sent by class i in round k and $sent_i(k)$ be the bytes sent by class i from round 1 through k. We have $sent_i(k) = \sum_{k=1}^{m} bytes_i(k)$.

It is easily observed that $bytes_i(k)$+deficit_counter[i][k] = Quantum[i]+ deficit_counter[i][k − 1]. As in round k, the accumulated allocation to class i is Quantum[i]+deficit_counter[i][k − 1]. Therefore, if class i sends $bytes_i(k)$, then the reminder will be stored in deficit_counter[i][k]. Since the queue for class i never empties during the interval (t_1, t_2), we will have:

$$\begin{aligned} bytes_i(k) =& \text{Quantum[i]}+ \\ & \text{deficit\_counter[i][k} - 1] - \text{deficit\_counter[i][k]}. \end{aligned}$$

Summing this over all m rounds of servicing of class i, and because

$sent_i(k) = \sum_{k=1}^{m} bytes_i(k)$, we have

$$sent_i(m) = m \cdot \text{Quantum[i]} + \text{deficit\_counter[i][0]} - \text{deficit\_counter[i][m]}.$$

Then the result follows because deficit_counter[i] is always non_negative and upper bounded by Max (by Lemma 1). ∎

Theorem 1: *Under the DRR service discipline, for an interval* (t_1, t_2), *we have the following fairness measure:*

$$FM(t_1, t_2) \leq 2 \cdot Max + \text{Quantum}, \quad where \ \text{Quantum} = Min(\text{Quantum[i]}).$$

Proof : Consider an interval (t_1, t_2) under the DRR algorithm and any two class i and j that are backlogged in this interval. As each class is serviced in a strict round-robin mode, therefore, if we let m be the number of the round-robin opportunities given to class i in the interval (t_1, t_2), and we let m' be the number of round-robin opportunities given to class j in the same interval, then we have $| m - m'| \leq 1$. From Lemma 2 we get:

$$sent_i(t_1, t_2) \leq m \cdot \text{Quantum[i]} + Max.$$

Based on the definition, c_i, the share given to any class i, is equal to Quantum[i]/Quantum. Therefore the normalized service received by class i is

$$sent_i(t_1, t_2)/c_i \leq m \cdot \text{Quantum} + Max/c_i.$$

Similarly, for class j, we can obtain:

$$\begin{aligned} sent_j(t_1, t_2) &\geq m' \cdot \text{Quantum[j]} - Max \\ &= m' \cdot \text{Quantum[j]} - Max \end{aligned}$$

and

$$sent_j(t_1, t_2)/c_j \geq m' \cdot \text{Quantum[j]} - Max/c_j.$$

Subtracting the equations for the normalized service for class i and j, and using the fact that $m - m' \leq 1$, we get

$$\frac{sent_i(t_1, t_2)}{c_i} - \frac{sent_j(t_1, t_2)}{c_j} \leq \mathsf{Quantum} + \frac{Max}{c_i} + \frac{Max}{c_j}$$

As both c_i and c_j are greater than 1, the theorem follows. ∎

□ **End of chapter.**

Bibliography

[1] TCP SYN flooding and IP spoofing attacks. CERT Advisory CA-96.21. available at http://www.cert.org/.

[2] Udp port denial-of-service attack. CERT Advisory CA-1996-01. available at http://www.cert.org/advisories/CA-1996-01.html.

[3] Smurf IP denial-of-service attacks. CERT Advisory CA-1998-01, January 1998. avaliable at www.cert.org/advisories/CA-98.01.html.

[4] The arqos project, February 2002. Available at http://arqos.csc.ncsu.edu/.

[5] The authenticated qos project, August 2002. Available at http://www.citi.umich.edu/projects/qos/.

[6] W32.blaster.worm, August 2003. Available at http://securityresponse.symantec.com/avcenter/venc/data /w32.blaster.worm.html.

[7] Understanding and configuring mdrr/wred on the cisco 12000 series internet router, Jan. 2004. http://www.cisco.com/warp/public/63/mdrr_wred_overview.html.

[8] M. Allman and V. Paxson. On estimating end-to-end network path properties. In *Proc. ACM SIGCOMM*, Vancouver, Canada, September 1999.

[9] S. Bellovin. The icmp traceback message, 2000.

[10] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An architecture for differentiated services, rfc 2475. 1998.

[11] H. Burch. Tracing anonymous packets to their approximate source. In *LISA '00: Proceedings of the 14th USENIX conference on System administration*, pages 319–328, Berkeley, CA, USA, 2000. USENIX Association.

[12] Y. Chen, K. Hwang, and Y. K. Kwok. Collaborative defense against periodic shrew ddos attqcks in frequency domain, May 2005. Available at http://gridsec.usc.edu/files/TR/ACMTISSEC-LowRateAttack-May3-05.pdf.

[13] P. J. Criscuolo. Distributed denial of service: Trin00, tribe flood network, tribe flood network 2000, and stacheldraht ciac-2319. Feb. 2000.

[14] M. E. Crovella and A. Bestavros. Self-similarity in world wide web traffic: Evidence and possible causes. *IEEE/ACM TRANSACTIONS ON NETWORKING*, 5(6):835–846, 1997.

[15] J. B. David Moore, Colleen Shannon. Code-red: a case study on the spread and victims of an internet worm. In *Proceedings of the Internet Measurement Workshop 2002*, 2002.

[16] S. Deshpande, M. Thottan, and B. Sikdar. Early detection of bgp instabilities resulting from internet worm attacks. *Global Telecommunications Conference, 2004. GLOBECOM '04. IEEE*, 4:2266 – 2270, 2004.

[17] D. Dittrich. The dos project's trinoo distributed denial of service attack tool, 1999. University of Washington, http://staff.washington.edu/dittrich/misc/trinoo.analysis.txt.

[18] K. Fall and S. Floyd. Simulation-based comparisons of tahoe,reno,and sack tcp. *ACM Computer Comm. Review*, 5(3):5–21, 1996.

[19] P. Ferguson and D. Senie. Network ingress filtering: Defeating denial of service attacks which employ ip source address spoofing, rfc 2827. May 2000.

[20] P. Ferrie, P. Szor, and F. Perriot. Worm wars, October 2003. Available at http://www.peterszor.com/welchia.pdf.

[21] V. Gilgor. A note on the denial-of-service problem. In *Proceedings of the 1983 IEEE Symposium on Security and Privacy*, 1983.

[22] M. T. Goodrich. Efficient packet marking for large-scale ip traceback. In *CCS '02: Proceedings of the 9th ACM conference on Computer and communications security*, pages 117–126, New York, NY, USA, 2002. ACM Press.

[23] M. Guirguis, A. Bestavros, and I. Matta. Exploiting the transients of adaptation for roq attacks on internet resources. In *Proceedings of the 12th IEEE International Conference on Network Protocols*, Oct 2004.

[24] J. Ilow. Parameter estimation in farima processes with applications to network traffic modeling. In *Proceedings of the Tenth IEEE Workshop on Statistical Signal and Array Processing*, August 2000.

[25] G. P. J. Mirkovic and P. Reiher. Attacking ddos at the source. In *Proceedings of ICNP 2002*, pages 312–321, Paris, France, 2002.

[26] J. Jung, B. Krishnamurthy, and M. Rabinovich. Flash crowds and denial of service attacks: Characterization and implications for cdns and web sites, 2002.

[27] E. Keogh. Exact indexing of dynamic time warping. In *Proceedings of the 28th VLDB Conference*, Hong Kong, China, Aug. 2002.

[28] A. Keromytis, V. Misra, and D. Rubenstein. Sos: Secure overlay services, 2002.

[29] D. M. Kienzle and M. C. Elder. Recent worms: a survey and trends. In *WORM '03: Proceedings of the 2003 ACM workshop on Rapid Malcode*, pages 1–10, New York, NY, USA, 2003. ACM Press.

[30] A. Kuzmanovic and E. Knightly. Low-rate TCP-targeted denial of service attacks. In *Proc. ACM SIGCOMM*, Karlsruhe, Germany, August 2003.

[31] K. T. Law, J. C. S. Lui, and D. K. Y. Yau. You can run, but you can't hide: An effective methodology to traceback ddos attackers. In *Proc. of IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems (MASCOTS'02)*, 2002.

[32] W. E. Leland, M. S. Taqqu, W. Willinger, and D. V. Wilson. On the self-similar nature of ethernet traffic (extended version). *IEEE/ACM TRANSACTIONS ON NETWORKING*, 2(1):1–15, 1994.

[33] R. Lemos. Year of the worm: Fast-spreading code is weapon of choice for net vandals, March 2001. Available at http://news.com.com/Year+of+the+Worm/2009-1001_3-254061.html.

[34] J. Li, J. Mirkovic, M. Wang, P. Reiher, and L. Zhang. Save: source address validity enforcement protocol. In *Proceedings of IEEE Infocom*, 2002.

[35] M. Liljenstam and D. Nicol. Comparing passive and active worm defenses. In *Proceedings of the First International Conference on the Quantitative Evaluation of Systems (QEST04)*, pages 18 – 27, 2004.

[36] J. Liu. Traffic modeling based on farima models. In *Proceedings of the 1999 IEEE Canadian Conference on Electrical and Computer Engineering*, May 1999.

[37] X. Luo and R. K.C.Chang. On a new class of pulsing denial-of-service attacks and the defense. In *Proceedings of Network and Distributed System Security Symposium (NDSS)*, Feb. 2005.

[38] R. Mahajan, S. Floyd, and D. Wetherall. Controlling high-bandwidth flows at the congested router. In *Proc. ICNP*, Riverside, CA, November 2001.

[39] V. Misra, W.-B. Gong, and D. F. Towsley. Fluid-based analysis of a network of AQM routers supporting TCP flows with an application to RED. In *SIGCOMM*, pages 151–160, 2000.

[40] G. Montenegro. Reverse tunneling for mobile ip, rfc 2344. May 1998.

[41] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, and N. Weaver. Slammer worm dissection: Inside the slammer worm. *IEEE Security and Privacy*, 1(4):33–39, 2003.

[42] D. Moore, G. Voelker, and S. Savage. Inferring internet denial-of-service activity,. In *Proceedings of Usenix Security Symposium*, Washington D.C., 2001.

[43] R. Needham. Denial of service: An example. *Communications of the ACM*, 37(11):842–47, 1994.

[44] D. M. Nicol and M. Liljenstam. Models of active worm defenses. In *Proceedings of the IPSI Studenica Conference, Serbia*, 2004.

[45] J. Padhye, V. Firoiu, D. Towsley, and J. Krusoe. Modeling TCP throughput: A simple model and its empirical validation. *Proceedings of the ACM SIGCOMM '98 conference on Applications, technologies, architectures, and protocols for computer communication*, pages 303–314, 1998.

[46] K. Park and H. Lee. On the effectiveness of route-based packet filtering for distributed dos attack prevention in power-law internets. In *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 15–26, New York, NY, USA, 2001. ACM Press.

[47] V. Paxson. End-to-end routing behavior in the Internet. In *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, volume 26,4 of *ACM SIGCOMM Computer Communication Review*, pages 25–38, New York, August 1996. ACM Press.

[48] T. Peng, C. Leckie, and K. Ramamohanarao. Protection from distributed denial of service attacks using history-based ip filtering. In *Proceedings of IEEE International Conference on Communications*, volume 1, pages 482 – 486, May 2003.

[49] C. Perkins. Ip mobility support, rfc 2002. 1996.

[50] P.McKenney. Stochastic fairness queueing. *Internetworking: Research and Experience*, 2:157–173, 1986.

[51] Reuters. Yahoo on trail of site hackers, 2000. http://wired-vig.wired.com/news/business/0,1367,34221,00.html.

[52] H. Sakoe and H. Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Trans. Acoustics, Speech, and Signal Proc.*, ASSP-26 No.1, Feb. 1978.

[53] S. Savage, D. Wetherall, A. Karlin, and T. Anderson. Practical network support for IP traceback. In *Proc. ACM SIGCOMM*, Stockholm, Sweden, August 2000.

[54] S.Gibson. Distributed reflection denial of service: Description and analysis of a potent, increasingly prevalent, and worrisome internet attack, February 2002. http://www.grc.com/dos/drdos.htm.

[55] S.Golestani. A self clocked fair queueing scheme for broadband applications. In *Proc. IEEE INFOCOM*, 1994.

[56] M. Shreedhar and G. Varghese. Efficient fair queueing using deficit round robin. *IEEE/ACM Transactions on Networking.*, 4(3), June 1996.

[57] A. C. Snoeren, C. Partridge, L. A. Sanchez, C. E. Jones, F. Tchakountio, S. T. Kent, and W. T. Strayer. Hash-based ip traceback. In *ACM SIGCOMM*, San Diego, August 2001.

[58] A. C. Snoeren, C. Partridge, L. A. Sanchez, C. E. Jones, F. Tchakountio, B. Schwartz, S. T. Kent, and W. T. Strayer. Single-packet ip traceback. *IEEE/ACM Trans. Netw.*, 10(6):721–734, 2002.

[59] D. Song and A. Perrig. Advanced and authenticated techniques for IP traceback. In *Proc. IEEE INFOCOM*, Anchorage, Alaska, 2001.

[60] S. M. Specht and R. B. Lee. Distributed denial of service: Taxonomies of attacks, tools, and countermeasures. In *Proceedings of the 17th International Conference on Parallel and Distributed Computing Systems, 2004 International Workshop on Security in Parallel and Distributed Systems*, pages 543–550, 2004.

[61] R. Stone. Centertrack: An ip overlay network of tracking dos floods. In *Proceedings of the 9th USENIX Security Symposium*, August 2000.

[62] Y. Xie, V. Sekar, D. A. Maltz, M. K. Reiter, and H. Zhang. Worm origin identification using random moonwalks. In *Proceedings of the 2005 IEEE Symposium on Security and Privacy*, pages 242 – 256, May 2005.

[63] Y. Xu and R. Guerin. On the robustness of router-based denial-of-service (dos) defense system. In *ACM SIGCOMM Computer Communication Review*, 2005.

[64] P. Yan and M. C. Lee. Towards an adaptive packet marking scheme for ip traceback. In *ICETE (2)*, pages 119–126, 2004.

[65] G. Yang, M. Gerla, and M. Y. Sanadidi. Defense against low-rate TCP-targeted denial-of-service attacks. In *Proc. ISCC*, Alexandria, Egypt, 2004.

[66] D. K. Yau, J. C. Lui, F. Liang, and Y. Yeung. Defending against distributed denial-of-service attacks with max-min fair server-centric router throttles. *Accepted for publication in IEEE/ACM Transactions on Networking.*

[67] W. Yu. Analyzing the performance of internet worm attack approaches. In *Proceedings of the 13th International Conference on Computer Communications and Networks, ICCCN 2004*, pages 501 – 506, 2004.

[68] L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala. RSVP: A new resource reservation protocol. *IEEE Network Magazine*, September 1993.