Change-Point Monitoring for Detection of DoS Attacks*

Haining Wang Danlu Zhang Kang G. Shin

Abstract

This paper presents a simple and robust mechanism, called *Change-Point Monitoring* (CPM), to detect denial of service (DoS) attacks. The core of CPM is based on the inherent network protocol behaviors, and is an instance of the Sequential Change Point Detection. To make the detection mechanism insensitive to sites and traffic patterns, a non-parametric Cumulative Sum (CUSUM) method is applied, thus making the detection mechanism robust, more generally applicable and its deployment much easier. CPM does not require per-flow state information and only introduces a few variables to record the protocol behaviors. The statelessness and low computation overhead of CPM make itself immune to any flooding attacks. As a case study, the efficacy of CPM is evaluated by detecting a SYN flooding attack — the most common DoS attack. The evaluation results show that CPM has short detection latency and high detection accuracy.

Keywords — CUSUM algorithm, DoS attacks, intrusion detection, protocol behavior

1 Introduction

The growing number of denial of service (DoS) attacks impose a significant threat on the availability of network services, and the vulnerability of the Internet to DoS attacks has been witnessed by the frequent attacks on Internet servers and their resultant disruption of services [15, 21, 37]. Due to the readily available tools and its simple nature, flooding packets is the most common and effective DoS attack. While flooding tools have been becoming more sophisticated, they have been getting easier to use. An adversary without much knowledge of programming can download a flooding tool and then launch a DoS attack. The flooding traffic of a DoS attack may originate from either a single source or multiple sources. We call the latter case a distributed denial of service (DDoS) attack. Briefly, a DDoS attack works as follows. An attacker sends control packets to the previously-compromised flooding sources, instructing them to target at a given victim. The flooding sources then collectively generate and send an excessive number of flooding packets to the victim, but with fake and randomized source addresses, so that the victim cannot locate the flooding sources.

*The authors were with the Department of Electrical Engineering and Computer Science, the University of Michigan, Ann Arbor, MI 48109-2122. Haining Wang (hnw@cs.wm.edu) is now with College of William and Mary, Danlu Zhang (dzhang@qualcomm.com) with Qualcomm Inc., and Kang G. Shin (kgshin@umich.edu) with the University of Michigan.

To foil DoS attacks, researchers have designed and implemented a number of countermeasures. In general, the countermeasures of DoS attacks can be classified into three different categories: detection, defense (or mitigation), and IP trace-back mechanisms. Detecting DoS attacks in real time is the first step of combating DoS attacks. An automated and fast detection is essential to the protection against DoS attacks. Upon timely detection of a DoS attack, more sophisticated defense mechanisms will be triggered to shield victim servers or link bandwidth from DoS traffic, and block the prorogation of DDoS traffic at routers. At the same time, we can perform more expensive IP trace-back to single out flooding sources. Unlike defense and trace-back mechanisms, detection itself should be an alway-on function with little overhead, causing minimal disruption to normal operations and withstanding any flooding attacks.

Basically, detecting DoS attacks belongs to network-based intrusion detection. A network-based intrusion detection system (NIDS) is based on the idea that an intruder's behavior will be noticeably different from that of a legitimate user and that many unauthorized actions are detectable. A commonly-used detection approach is either signature-based or anomaly-based. A signature-based NIDS inspects the passing traffic and searches for matches against already-known malicious patterns. In practice, several signature-based NIDSes have been developed and deployed at firewalls or proxy servers, such as Bro [41] and Snort [45]. By contrast, an anomaly-based NIDS observes the normal network behavior and watches for any divergence from the normal profile. Most of DoS detection systems are anomaly-based, like MULTOPS [17] and D-WARD [36]. However, their normal traffic models are mainly based on flow rates. Due to the diversity of user behaviors and the emergence of new network applications, it is difficult to obtain a general and robust model for describing the normal traffic behaviors.

We have observed that the server-client or peer-to-peer model of Internet applications demonstrates a unique *request vs. reply* protocol behavior, and the reliable data delivery leads to the inherent *data vs. acknowledgment (ACK)* protocol behavior. Based on these distinct network protocol behaviors, instead of traffic rates, in this paper we propose a simple and robust mechanism, called *Change-point Monitoring* (CPM), to detect DoS attacks. The rationale behind CPM is that there exists a strong positive correlation between requests (data) and the corresponding replies (ACKs) in the Internet as the inherent protocol behaviors, and DoS attacks easily destroy this strong correlation. In particular, we employ the non-parametric Cumulative Sum (CUSUM) method [6] to detect the cumulative effect of the deviation from normal protocol behaviors caused by a DoS attack. The key features of CPM include:

- CPM utilizes the inherent protocol behaviors for DoS detection. Since the protocol behaviors are

determined by solely the protocol specifications and the service models of Internet applications, CPM is independent of traffic flow rates or specific applications.

- CPM is insensitive to sites and traffic patterns due to its reliance on the non-parametric CUSUM method [6], thus making CPM robust, much more generally applicable, and its deployment easier.

- CPM plays a dual role in detecting DoS attacks: the first-mile (egress) CPM and the last-mile (ingress) CPM. Due to its close proximity to the flooding sources, the first-mile (egress) CPM not only alarms on the ongoing DoS attacks, but also helps reveal the origins of the flooding sources.

The simplicity (hence attractiveness) of CPM lies in its statelessness and low computation overhead — only a few variables are introduced to record the protocol behaviors with a few CPU cycles burned. Besides monitoring the ongoing traffic at firewalls, CPM can be installed at a leaf router that connects a stub network[1] to the Internet, or at an ISP edge router that connects a customer network to the ISP. Moreover, CPM can work independently at either a leaf (edge) router or a firewall, and it does not need any coordination with other routers or end-hosts. The independence of CPM determines that CPM can be incrementally deployed and its implementation overhead is low.

As a case study, we use CPM to detect a SYN flooding attack — the most common DoS attack. The efficacy of CPM is evaluated by extensive trace-driven simulations. Traces taken from different sites at different times are employed to evaluate the sensitivity of CPM. First, our trace-based study validates the coherence of TCP protocol behaviors, clearly showing their independence of sites and sampling times. Then, we inject SYN flooding traffic with different rates and investigate the detection sensitivity of CPM at different sites. The evaluation results show that CPM has short detection latency and high detection accuracy.

The remainder of the paper is organized as follows. Section 2 details our statistical detection methodology — the proposed CUSUM algorithm for detecting abnormal protocol behaviors. Section 3 describes the CPM framework, including the placement and structure of CPM and its dual role in detection. Section 4 presents our case study for detecting a SYN flooding attack. Section 5 evaluates the performance of CPM using trace-driven simulations for SYN flooding detection. Section 6 discusses related work. Finally, Section 7 states conclusions and future directions.

[1] A stub network only carries packets to and from local hosts.

2 Statistical Attack Detection

Like most statistical anomaly-detection systems, CPM compares the observed sequence with the profile that represents the user's normal behavior, and detects any significant deviation from the normal behavior. The key difference of CPM from others is that CPM exploits the inherent network protocol behaviors, instead of traffic patterns, for detecting network anomalies.

In general, for any associations among IP packets, TCP segments, or application-level messages that are determined solely by protocol specifications, we regard them as inherent protocol behaviors. For instance, according to the specification of TCP/IP protocol [54], in its normal operation, a FIN (RST) is paired with a SYN at the end of data transmission. The other network behaviors, to name a few, include TCP data segments and ACKs, ICMP requests and replies, DNS queries and replies, etc.

After distilling the inherent protocol behaviors from raw traffic flows, we can apply the CPM to detect an ongoing flooding attack by observing the violation of normal protocol behaviors. CPM can achieve more accurate detection with a shorter latency: the strong positive correlation between requests (or queries) and the corresponding replies enables CPM to detect abnormal behaviors quickly.

2.1 Change-Point Detection

The objective of Change-Point Detection is to determine if the observed time series is statistically homogeneous, and if not, to find the point in time when the change happens. This has been studied extensively by statisticians. See [2] and [6] for a good survey. There have been various tests for different problems. They can be largely divided into two categories: posterior and sequential. Posterior tests are done off-line where the entire data is collected first and then a decision of homogeneity or a change point is made based on the analysis of all the collected data. On the other hand, sequential tests are done on-line with the data presented sequentially and the decisions are made on-the-fly. Our attack detection algorithm belongs to the Sequential Change Point Detection [2].

We adopt the sequential test for quicker response when an attack occurs. It also saves memory and computation. One difficulty, however, is the modeling of requests' arrival process. For instance, despite the existence of a number of previous results on the modeling of TCP connection request arrivals [9, 10, 43, 49], there is no consensus on whether it should be modeled as self-similar or Poisson. For dynamic and complex systems like the Internet, it may not be possible to model the total number of session request[2] arrivals by a simple parametric description. So, we seek robust tests which are not

[2]A session request could be a TCP connection request, an ICMP request, or a DNS query, etc.

model-specific. In fact, non-parametric methods fit this requirement very well. Specifically, we use the non-parametric CUSUM (Cumulative Sum) method [6] for the detection of DoS attacks. This method enjoys all the virtues of sequential and non-parametric tests, and the computation load is very light. When the time series is independent identically distributed (i.i.d.) with a parametric model, CUSUM is asymptotically optimal for a wide range of Change Point Detection problems [2, 6].

2.2 The CUSUM Algorithm

For ease of presentation, we only show how it works in the request (QST) vs. reply (RLY) pair scheme, which is similar to the data vs. acknowledgment (ACK) pair scheme except that the collection of QSTs and RLYs is replaced with that of data and ACKs.

Let $\{\Delta_n, n = 0, 1, \cdots\}$ be the number of QSTs minus that of the corresponding RLYs collected within one sampling period. To further alleviate its dependence on the time, traffic pattern and size of the network, $\{\Delta_n\}$ is normalized by the average number, $\bar{R}$, of RLYs during each sampling period. $\bar{R}$ can be estimated in real time and updated periodically. An example of recursive estimation and update of $\bar{R}$ is:

$$\bar{R}(n) = \alpha \bar{R}(n-1) + (1-\alpha)\text{RLY}(n), \tag{1}$$

where n is the discrete time index and α is a constant, whose the default value is 0.01, lying strictly between 0 and 1 that represents the memory in the estimation. Let $X_n = \Delta_n / \bar{R}$, then

$\{X_n\}$ is no longer dependent on the network size or time-of-day. Its dynamics are solely the consequence of the protocol specification. So, we can consider $\{X_n\}$ as a stationary random process. Under the normal condition, the mean of X_n, denoted as c, is much less than 1 and close to 0.

$\{X_n\}$ is assumed to satisfy the following two conditions.

C1: $\{X_n\}$ is ψ-mixing, meaning that the $\psi(s)$ parameters, defined below, approach 0 as $s \to \infty$:

$$\psi(s) \stackrel{def}{=} \sup_{t \geq 1} \sup_{\substack{A \in \mathcal{R}_1^t, \\ B \in \mathcal{R}_{t+s}^{\infty}, \\ P(A)P(B) \neq 0}} \left| \frac{P(AB)}{P(A)P(B)} - 1 \right|, \tag{2}$$

where $\mathcal{R}_1^t$ is the σ-algebra generated by $\{X_1, X_2, \cdots, X_t\}$ and $\mathcal{R}_{t+s}^{\infty}$ is the σ-algebra generated by $\{X_{t+s}, X_{t+s+1}, \cdots\}$. $\psi(s)$ is affected by the dependency among the $\{X_n\}$ samples: highly dependent $\{X_n\}$ has $\psi(s)$ that decays slowly as $s \to 0$. In addition, the `sup` above means supremum — the tightest upper bound of the variable in the formula [46].

C2: The marginal distribution of $\{X_n\}$ satisfies the following regularity condition: $\exists t > 0$ such that $E(e^{tX_n}) < \infty$.

The details of these conditions can be found in [6]. Note that ψ-mixing is a much looser requirement than independence, and X_n being ψ-mixing only indicates that X_n is not "extremely" dependent. In practice, both conditions are mild and easily satisfiable, even for long-range dependent arrival processes. In general, $E(X_n) = c \ll 1$. We choose a parameter a that is an upper bound of c, i.e., $a > c$, and define $\tilde{X}_n = X_n - a$ so that it has a negative mean during normal operation. When a DoS attack takes place, $\tilde{X}_n$ will suddenly increase and become a large positive number. Suppose, during an attack, the increase in the mean of $\tilde{X}_n$ can be lower-bounded by h. Our change detection is based on the observation of $h \gg c$.

Let

$$\begin{aligned} y_n &= (y_{n-1} + \tilde{X}_n)^+, \\ y_0 &= 0, \end{aligned} \tag{3}$$

where x^+ is equal to x if $x > 0$ and 0 otherwise. The meaning of y_n can also be understood as follows: if we define $S_k = \sum_{i=1}^{k} \tilde{X}_i$, with $S_0 = 0$ at the beginning, it is straightforward to show that

$$y_n = S_n - \min_{1 \le k \le n} S_k, \tag{4}$$

i.e., the maximum continuous increment until time n. A large $\{y_n\}$ is a strong indication of an attack. Since Eq. (3) is recurrent and much easier to compute than Eq. (4), we will use it in making detection decisions.

Let $d_N(.)$ be the decision at time n: '0' for normal operation (homogeneity) and '1' for attack (a change occurs). Here N represents the flooding threshold:

$$d_N(y_n) = \begin{cases} 0 & \text{if } y_n \le N, \\ 1 & \text{if } y_n > N. \end{cases} \tag{5}$$

In other words, $d_N(y_n) = I(Y_n > N)$, where $I(.)$ is the indicator function. The purpose of introducing a is to offset the possible positive mean in $\{X_n\}$ caused by network anomalies so that the test statistic y_n will be reset to zero frequently and will not accumulate with time.

Let P_m and E_m be the probability measure and the expected value generated by $\{\tilde{X}_n\}$ with the attack occurring at time m (change point at time m); let P_∞ and E_∞ be the counterparts without any

attack (no change point). There are two fundamental performance measures for the sequential change point detection.

False alarm time (the time without false alarm): the time duration with no false alarm reported when there is no attack.

Detection time: the detection delay after the attack starts.

One would want the second measure to be as small as possible while keeping the first measure as large as possible. However, they are conflicting goals and cannot be simultaneously achieved. Therefore, the design philosophy of a statistical change point detection is to minimize the detection time subject to a certain false alarm tolerance. In order to compare the performance of different detection schemes, some criteria of false alarms must be specified, like average time between two consecutive false alarms, worst-case false alarm time, and so on. An algorithm is said to be *optimal* with respect to a certain criterion if it minimizes the detection time for an attack among all the detection schemes subject to the false alarm constraint. The CUSUM rule has been shown to be asymptotically optimal with respect to the worst-case mean false alarm time in the change-point detection problems involving a known parametric model and independent observations [2].

Due to the lack of a complete model for $\{\tilde{X}_n\}$, it is difficult to discuss optimality. The choice of CUSUM is based on its simplicity in computation and non-parametric implementation, as well as its generally excellent performance. It has been shown in [6] that, with the choice of a, the upper bound in case of normal operation, and N, the flooding threshold, as N becomes large, we have

$$\sup_n |\ln P_\infty(d_N(n) = 1)| \sim O(N), \tag{6}$$

which is equivalent to

$$P_\infty\{d_N(n) = 1\} \sim c_1 \exp(-c_2 N) \tag{7}$$

where c_1 and c_2 are constants, depending on the marginal distribution and mixing coefficients of $\{\tilde{X}_n\}$. In other words, the time between consecutive false alarms grows exponentially with N. The burstiness of traffic is reflected by the mixing coefficients $\psi(s)$, and thus, does impact the detection performance. However, the constants c_1 and c_2 only play a secondary role and can be ignored in practice.

In order to study the detection time, let us define

$$\begin{aligned} \tau_N &= \inf\{n : d_N(.) = 1\}, \\ \rho_N &= \frac{(\tau_N - m)^+}{N}, \end{aligned} \tag{8}$$

where m represents the starting time of the attack and ρ_N represents the normalized detection time after the occurrence of a change, and `inf` means infimum, or the greatest lower bound [46]. In CUSUM, for any $m \geq 1$, if h is the actual increase in the mean of $\{\tilde{X}_n\}$ during an attack, we have

$$\rho_N \rightarrow \gamma = \frac{1}{h - |c - a|}, \tag{9}$$

where $h - |c - a|$ is the mean of $\{\tilde{X}_n\}$ when $n > m$ (after an attack starts). However, since h is a bound rather than a true value, the above is a conservative estimation (an upper bound) of the actual detection time. The exact choice of parameters a, h and N will be detailed in Section 4.4 when we apply this method for detecting SYN flooding attacks.

3 The Framework of CPM

In this section, we first briefly describe the placement of CPM, then present the structure of CPM, and discuss the dual role of CPM.

3.1 Placement of CPM

As has been done in most NIDSes, it is possible that CPM could be placed on the link that connects a stub (customer) network to the Internet by monitoring the bidirectional traffic on that link. However, besides the extra specialized equipment and manpower required, during high peak (near saturation) flow rates, almost no event of any kind would be logged by NIDSes— they either have to drop packets at a very high rate or require a high-performance multi-CPU architecture for packet state analysis.

Therefore, unlike the traditional NIDS that passively monitors bidirectional traffic on network links, CPM is transparently interposed at either a leaf router or an ISP edge router, and is implemented as a loadable module of the router. In addition to its installation at leaf or ISP edge routers, CPM can also be placed at the firewall or the proxy server of a large organization which has only a single connection to the external world. All packets of a session must pass through the same CPM. However, we do not recommend the CPM to be installed at core routers mainly because (1) it is close to neither flooding sources nor the victim; (2) packets of the same flow could traverse different paths; (3) it is not always possible to accurately classify different transport-layer packets at core routers due to the possible use of IPSec; and (4) it cannot detect the reflected flooding attacks [42] easily, since malicious packets are diffused before reaching the core router.

3.2 Structure of CPM

Installed at a leaf (ISP edge) router, CPM consists of two *sniffers*, one at the in-bound interface and the other at the out-bound interface of the router. We refer to the traffic from the Internet to the stub (customer) network as *in-bound*, and the traffic in the other direction as *out-bound*. The *in-bound sniffer* (*out-bound sniffer*) monitors the incoming (outgoing) traffic. Figure 1 illustrates the structure of CPM placed at a leaf router.

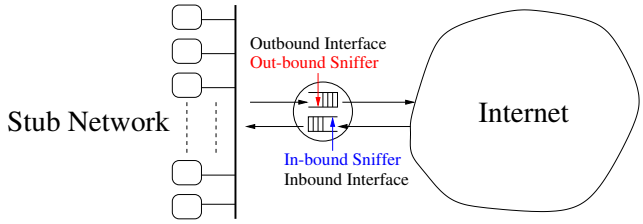


Figure 1: The structure of CPM placed at a leaf router

Both sniffers are software-based agents. Each sniffer consists of three components: a packet classifier, a packet counter, and a CUSUM detector. The packet classifier is used to distinguish the targeted packets such as TCP SYNs from the raw IP traffic. Large-scale packet classification mechanisms [18, 30, 53] have been proposed and implemented, making it possible for routers to differentiate the targeted packets from others at a very high speed. Therefore, the CPM's capability to withstand any flooding attacks depends on the ability of a leaf router in classifying and forwarding packets, typically at the rate of a million packets per second [30]. The packet counter includes a few additional variables that are introduced to record the number of targeted packets at each interface of a leaf router. No per-connection state or state computation is involved in CPM. Unlike the other NIDSes that maintain state for *each* TCP connection, CPM does not have the cold-start problem[3] mentioned in [19]. The CUSUM detector takes the variables of packet counter as its input, and executes the CUSUM detection algorithm given in Section 2.2.

3.3 Dual Role of CPM

Each leaf (ISP edge) router can be either the first-mile (egress) or the last-mile (ingress) router, depending on the direction of traffic between the stub (customer) network and the Internet. The CPM at a leaf (edge) router, therefore, plays a dual role in detecting flooding attacks:

[3] A "cold start" refers to the situation when a network intrusion detection system begins to run, or after it is restarted, it doesn't know how to deal with the incoming TCP traffic that belongs to the connections established earlier.

- as the first-mile (egress) CPM, it detects the flooding attacks originated from its local stub (customer) network and traces the flooding sources inside the local stub (customer) network; and
- as the last-mile (ingress) CPM, it detects flooding attacks on a server inside the local stub (customer) network, and issues a warning signal upon detection of an attack.

The first-mile (egress) CPM plays the primary role in sniffing a flooding attack, due mainly to its proximity to the flooding sources. Once an ongoing SYN flooding attack is detected, the first-mile (egress) CPM's warning signal automatically indicates the flooding sources to be inside the stub (customer) network to which the CPM is connected. However, the detection sensitivity of the first-mile (egress) CPM may diminish as more flooding sources participate and locate in different sites. In a large-scale DDoS attack, the flooding sources can be orchestrated so that each flooding source may cause only an insignificant deviation from the normal traffic pattern.

In contrast, the last-mile (ingress) CPM can quickly detect the flooding attacks as all of the flooding traffic streams are merged at the last-mile (ingress) router. Although it cannot provide any hint about the flooding sources, upon receipt of the last-mile (ingress) CPM's warning signal for an attack, the defense system like SynDefender [34] can be triggered to protect the victim. To bring down the victim under protection, the flooding sources have to increase their flooding rates significantly, but this will make it easier for the first-mile (egress) CPM to detect the flooding attack and locate its source(s). So, the last-mile (ingress) CPM plays an important complementary role in detecting DoS attacks.

For ease of presentation, in the rest of the paper we only use the terms of leaf router, first-mile CPM and last-mile CPM, and stub networks. However, they are exchangeable with ISP edge router, egress CPM and ingress CPM, and customer networks, respectively.

4 Detecting SYN Flooding Attacks

As a case study, we evaluate the efficacy of CPM by detecting SYN flooding attacks. The recent research results have shown that more than 90% of the DoS attacks use TCP [37], and TCP SYN flooding dominates in the available attacking tools and the number of DoS attacks known to date [37]. TCP SYN flooding consists of a stream of spoofed TCP SYN packets directed to a listening TCP port of the victim. Not only the Web servers but also any system connected to the Internet providing TCP-based network services, such as FTP or mail servers, are susceptible to TCP SYN flooding attacks.

4.1 SYN and Reflected SYN/ACK Flooding

SYN flooding attacks exploit the TCP's three-way handshake mechanism and its limitation in maintaining half-open connections. When a server receives a SYN packet, it returns a SYN/ACK packet to the client. Until the SYN/ACK packet is acknowledged by the client, the connection remains in half-open state for a period of up to the TCP connection timeout, which is typically set to 75 seconds. The server has built in its system memory a backlog queue to maintain all half-open connections. Since this backlog queue is of finite size, once the backlog queue limit is reached, all connection requests will be dropped. If a SYN packet is spoofed, the victim server will never receive the final ACK packet to complete the three-way handshake. Flooding spoofed SYN packets can easily exhaust the victim server's backlog queue, causing all the incoming SYN packets to be dropped.

While the conventional SYN flooding is an attack of "system resource consumption," the recent reflected SYN/ACK flooding attacks [16] virtually "disconnect" a victim server from the Internet by hogging the link bandwidth between the victim and its ISP with an excessive number of SYN/ACK packets (a.k.a. bandwidth consumption attack). It is a kind of Distributed Reflection DoS (DRDoS) attacks [42]. In reflected SYN/ACK flooding attacks, a large number of innocent BGP routers (service port 179) and well-known TCP servers are exploited as the reflectors. The attacker "sprays" the spoofed SYN packets, whose source IP addresses are falsified as the victim's IP address, across a large number of reflectors. Each reflector alone only receives a moderate flux of spoofed SYN packets so that it can easily sustain the availability of its normal service. However, these innocent reflectors involuntarily reflect and amplify the malicious SYN packets. Their SYN/ACK responses, which are aggregated and flooded to the victim, are excessive, exhausting the link bandwidth between the victim and its ISP.

Note that in reflected SYN/ACK flooding attacks, all malicious SYN packets from the attacker must traverse the leaf router that connects the attacker to the Internet, in order to reach the Internet and then get sprayed across the numerous reflectors. The CPM installed at this leaf router can detect the flow of these malicious SYNs, since no SYN/ACKs return to the attacker and the total number of malicious SYNs is still very large. So, the same method for detecting SYN flooding attacks can be applied to detect reflected SYN/ACK flooding attacks. To CPM, the reflected SYN/ACK flooding attack is just a variation of the conventional SYN flooding attack.

4.2 Detection Methods

Based on the inherent protocol behavior of TCP connection establishment and teardown, we utilize two types of packet pairs — SYN vs. FIN and SYN vs. SYN/ACK pairs — to detect SYN flooding attacks. According to the type of packet pairs used, we devise two different methods for SYN flooding detection.

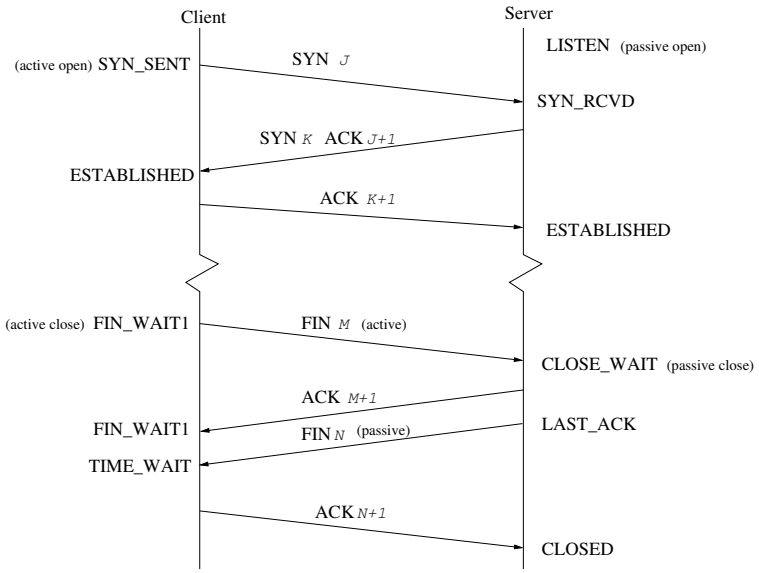


Figure 2: TCP states corresponding to normal connection establishment and teardown (from [54])

As shown in Figure 2 which is borrowed from [54], SYN and FIN packets delimit the beginning (SYN) and end (FIN) of each TCP connection in the same direction. In contrast, SYN and SYN/ACK packets signal the start of a TCP connection establishment in two opposing directions. Under the normal condition, one appearance of a SYN packet results in: (1) the eventual return of a FIN packet in the same direction; and (2) the corresponding transmission of a SYN/ACK packet in the reverse direction within one round-trip time (RTT). Thus, the difference between the number of SYN and FIN (or SYN/ACK) packets can be utilized to detect SYN flooding attacks.

4.2.1 SYN vs. FIN pairs

The first detection method utilizes the SYN vs. FIN pairs. Because a SYN packet and the corresponding FIN pass through a leaf router in the same direction (i.e., the same interface as shown in Figures 1 and 2), the SYN vs. FIN pair can be monitored by the same sniffer. No coordination and communication between these two sniffers are required. The first-mile CPM employs only the out-bound sniffer, while the last-mile CPM uses the in-bound sniffer only. Although SYN packets can be distinguished from

SYN/ACK packets, there is no way to discriminate active FINs from passive FINs, since each end-host behind a leaf router may be either a client or a server. So, the SYN vs. FIN pairs refer to both (SYN, FIN) and (SYN/ACK, FIN). In this detection method, the SYN packets are "generalized" to include the pure SYN and SYN/ACK packets.

Under a long-running normal condition, the TCP semantics has the one-to-one correspondence between SYNs and FINs. However, in reality there can always be a discrepancy between the number of SYNs and FINs. Besides the small number of long-lived TCP sessions, the other major cause of this discrepancy lies in the occurrence of RST packets. A single RST packet can terminate a TCP session without generating any FIN packet, which violates the SYN vs. FIN pair's protocol behavior. RSTs are generated for two reasons: (1) *passive*, or the RST is transmitted upon arrival of a packet at a closed port; and (2) *active*, or the RST is initiated by a client to abort a TCP connection.[4] Each active RST is associated with the SYN from the same session, and both of them can be seen by the same sniffer. In contrast, a passive RST cannot be associated with any SYN seen by the same sniffer as the passive RST and its corresponding SYN must go through different sniffers. Furthermore, passive RSTs may even have nothing to do with SYNs. For instance, late arrival of a data packet at the port that has already been closed, will trigger the transmission of an RST. We treat passive RSTs as a background noise.

In general, three types of SYN pairs are considered as the normal behavior of TCP in the first detection method: (SYN, FIN), (SYN/ACK, FIN) and (SYN, $\text{RST}_{\text{active}}$). Unfortunately, CPM cannot distinguish active RSTs from passive ones. There are two simple but extreme ways to resolve this problem: one is to treat all RSTs as active, and the other is to treat all RSTs as passive. The first approach degrades the CPM detection sensitivity, while the second raises the CPM false alarm rate. To make a trade-off between detection sensitivity and false alarm rate, it is necessary to set an appropriate threshold to filter most of the background noise. Based on our observation, under the normal condition: (1) SYNs and RSTs have a strong positive correlation; and (2) the difference between the number of SYNs and that of FINs is close to the number of RSTs. These imply that passive RSTs constitute only a small percentage of the entire RSTs. So, we set the threshold to 75%, i.e., three out of four RSTs are treated as active. Moreover, for the following reason, CPM can withstand the negative impact of passive RSTs that are incorrectly classified as active ones: at the end of each observation period, the CUSUM algorithm resets any negative difference between the number of SYNs and that of FINs (RSTs) to zero, so the spike of background noise is confined to one observation period only, preventing its cumulative

[4] Active RSTs are issued mostly by clients. In its own best interest, a server rarely sends RST packets to clients once the TCP connection is established.

effects.

The weakness of the SYN vs. FIN pairs scheme lies in its vulnerability to simple counter-measures. Once the attacker is aware of the presence of such a detection mechanism, it can paralyze the mechanism by flooding a mixture of SYNs and FINs (RSTs). Although one can argue that by doubling its flooding traffic, the attacker increases the possibility of being traced back, one may still wonder if there is a better way to overcome this shortcoming.

4.2.2 SYN vs. SYN/ACK pair

Fortunately, there is an alternative that is difficult for an attacker to counter. In the normal TCP three-way handshake, an out-bound SYN induces an in-bound SYN/ACK within a round-trip time. In contrast, for the flooded SYNs, because their spoofed IP source addresses are randomized, most of the corresponding SYN/ACKs will never return to the flooding sources, and hence, cannot go through the same leaf router as those flooding SYNs as shown in Figure 3 (mis-match part).

The second detection method makes use of SYN vs. SYN/ACK pairs to sniff flooding attacks. Since SYN/ACK packets are generated by the victim server, it is much more difficult for the flooding sources to evade the CPM. Moreover, as compared to the SYN vs. FIN pair scheme, the interval between SYN and SYN/ACK is bounded by one RTT, not by the duration of a TCP session that has much larger variations.

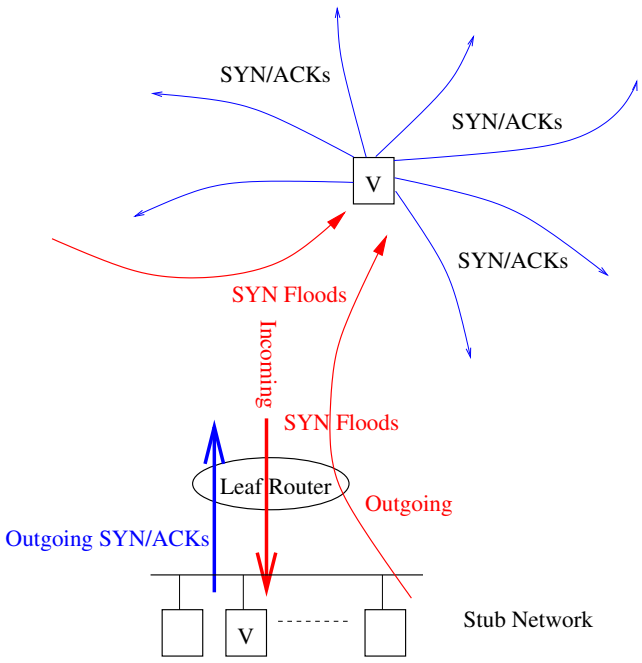


Figure 3: Match and mis-match between SYN vs. SYN/ACK pair at a leaf router

On the other hand, there are two disadvantages of the second detection method. First, unlike the first detection method, the out-bound sniffer and the in-bound sniffer must be coordinated. The out-bound sniffer maintains the count of outgoing SYNs and the in-bound sniffer keeps track of incoming SYN/ACK packets. At the end of each observation period, the count information must be exchanged between the two sniffers. Second, the SYN vs. SYN/ACK pair scheme is restricted to be used by the first-mile CPM only, which sniffs the flooding sources inside the local stub network. It lacks the capability to issue a timely last-mile flooding warning to the network administrator of the local stub network that is under attack. The reason for this is that: (1) each victim server generates a SYN/ACK in response to each SYN it received, regardless whether it is spoofed or not; and (2) the incoming SYN flood and the outgoing SYN/ACKs pass through the same local leaf router. This phenomenon is illustrated in Figure 3 (match part). So, there is no noticeable difference between the number of incoming SYN packets and that of the outgoing SYN/ACKs generated by the victim servers until the victim servers are totally shut down and no more SYN/ACKs are generated. Therefore, the last-mile CPM will still rely on SYN vs. FIN pairs for timely detection of an incoming SYN flooding attack.

4.2.3 CPM in Detecting SYN Flooding

As mentioned earlier, CPM plays a dual role: one as the first-mile CPM for sniffing flooding sources, and the other as the last-mile CPM for issuing attack warnings. To make CPM robust and powerful in SYN flooding detection, both SYN flooding detection methods are included in the CPM. The SYN vs. SYN/ACK pair method is employed by the first-mile CPM to sniff flooding sources inside the local stub network, while the SYN vs. FIN pairs method is used by the last-mile CPM to detect incipient flooding attacks, and issue a warning to the local network administrator.

4.3 Robustness against Network Anomalies

While there is no strict one-to-one match, under the normal condition, a very strong positive correlation between the numbers of SYNs and FINs (RSTs) or SYN/ACKs does exist as shown in Section 5.2. The discrepancy between the numbers of SYN and FIN (RST) or SYN/ACK packets is due to SYN losses and subsequent retransmissions. The SYN losses are caused by various network anomalies, including network congestion, routing loops and link failures. Clearly, these network anomalies reduce the detection sensitivity of CPM.

Fortunately, these network anomalies are triggered by unrelated events; and to date, there exists little evidence indicating that these different network anomalies are closely correlated. The recent Internet

measurement studies have shown that: (1) there is little congestion inside the core of the Internet, where the bandwidth over-provisioning has been widely used and the link utilization varies from 3 to 60% [5]; (2) the majority of routing loops last shorter than 10 seconds [22]; and (3) link failures are fairly well spread even in the time scale of minutes [20]. Therefore, these randomly-occurring network anomalies can be treated as white noise. To offset the effect of white noise, a safety margin can be added when the normal upper bound, a, is set, without jeopardizing the detection sensitivity. Only a severe network congestion, long lasting routing loops, and bursty occurrence of link failures — which rarely happen — can confuse the CPM to issue false alarms.

4.4 Parameter Specification

To use the CUSUM algorithm for SYN flooding detection, we only need to specify three tunable parameters: a, the upper bound of c, which is the mean of $\{X_n\}$ in normal operation; h, the lower bound of the increase in $\{\tilde{X}_n\} = \{X_n - a\}$ during an attack; and, N, the flooding threshold.

The CUSUM algorithm requires $E(\tilde{X}_n) < 0$ before the change point, and $E(\tilde{X}_n) > 0$ after it, i.e., $a > c$ and $h > a$. Based on the discussion in Section 2.2, to ensure a long false alarm time and make it independent of network size and traffic pattern, we set $h = 2a$ in our design. As the last-mile CPM that utilizes SYN vs. FIN pairs for flooding detection monitors the incoming traffic, all the SYN flooding packets converge, and therefore, a large difference between the numbers of SYN and FIN (RST) packets is easily observable with $h \gg c$. In this case, the detection is not sensitive to the choice of a. With a large safe margin, we can simply choose $a = 1$ and $h = 2$.

In contrast, as the first-mile CPM that employs SYN vs. SYN/ACK pairs for flooding detection monitors the outgoing SYN and incoming SYN/ACK traffic, only part of the flooding SYN packets can be seen by each detector because an attack may be initiated from many sites simultaneously. Thus, a proper choice of a is more important. To balance the detection sensitivity and false alarm rate, we set $a = 0.35$ and $h = 0.7$. Note that the choices of a and h are insensitive to network size and traffic pattern. In doing so, a universal false alarm rate can be realized for easy implementability of our detection mechanism. On the other hand, in practice, the network administrator of the involved edge router can incorporate site-specific information so that the algorithm can achieve higher detection sensitivity.

Based on a and h, the flooding threshold N can be specified as follows: (1) assume $c = 0$, and γ can thus be obtained from Eq. (9); and (2) specify a target detection time (i.e., the product of γ and N) such that the flooding threshold N is determined by Eq. (8). We choose t_0 as the designed detection time

for the last-mile CPM, hence $\gamma = 1$ and $N = 1$. In contrast, we choose $3t_0$ as the counterpart for the first-mile CPM, hence $\gamma = 2.86$ and $N = 1.05$.[5] Compared to the last-mile CPM, the short detection time of the first-mile CPM is not so crucial to the victim: the revelation of the flooding sources is more valuable, although it may take longer time. Note that the value of N is partially determined by the designed detection time, so it may not be larger than the value of h.

It is worth noting that our algorithm is to check the cumulative effect of an attack. So, it can detect attacks with the SYN flooding rate less than h at the expense of a longer response time. The actual lower bound of detection sensitivity in terms of SYN flooding rate, f_{min}, can be given as

$$f_{min} = (a - c) \cdot \frac{\bar{R}}{t_0}. \tag{10}$$

Furthermore, the detection capability is not sensitive to the flooding pattern: it can detect the attacks with both constant and bursty flooding rates. The effectiveness of CPM is evaluated by trace-driven simulations.

5 Performance Evaluation

To evaluate and validate the CPM, we have conducted trace-driven simulation experiments. The trace data we used are collected from four different sites at different times. The first trace was gathered at DEC's (now HP) primary Internet access point, which is an Ethernet DMZ network. It contains an hour's worth of all wide-area traffic between DEC Western Research Lab and the Internet on March 9th, 1995. The second trace was taken on March 13th, 1997 on a 10 Mbps Ethernet connecting Harvard's main campus to the Internet, which is a half-hour trace. The third set was obtained by placing network monitors on the high-speed link (OC-12, 622 Mpbs) that connects the University of North Carolina at Chapel Hill (UNC) campus network to the rest of the world. The trace was collected on September 27th, 2000. The fourth set was collected at the Internet access link that connects the University of Auckland at New Zealand to the rest of the world. The tracing ran from 14:36 to 17:47 on Thursday, December 5th, 2000.

The traces used in our experiments are summarized in Table 1. Note that the DEC and Harvard traces are mixed traffic collections in both directions, but the UNC and Auckland traces are uni-directional: UNC-in and Auckland-in collected the traffic data from the Internet to the UNC and Auckland campus networks, respectively, while UNC-out and Auckland-out collected the traffic data from

[5] N may not seem to be large on the absolute term, but it is large relative to normal fluctuations.

Table 1: A summary of the trace features

Trace	Starting time	Traffic type
DEC	2:00, Thu Mar 9, 1995	Bi-directional
Harvard	12:39, Thu Mar 13, 1997	Bi-directional
UNC-in	19:30, Wed Sept 27, 2000	Uni-directional
UNC-out	19:30, Wed Sept 27, 2000	Uni-directional
Auckland-in	14:36, Thur, Dec 5, 2000	Uni-directional
Auckland-out	14:36, Thur Dec 5, 2000	Uni-directional

the UNC and Auckland campus networks to the Internet, respectively.

5.1 Data Sampling

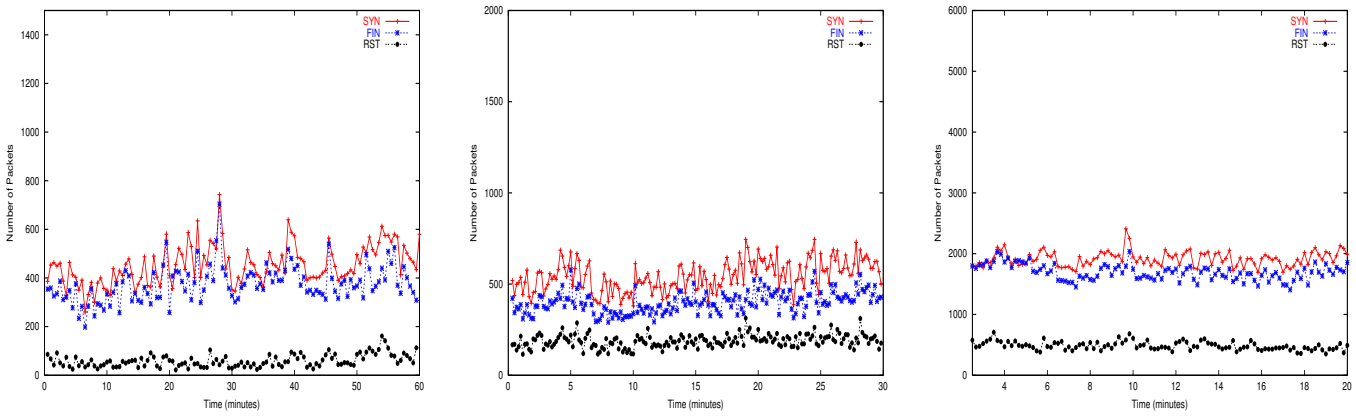
We collect the numbers of SYN, SYN/ACK, and FIN (RST) packets during every observation period t_0, which determines the detection resolution. In order to relate the SYN and FIN (RST) packets of the same connection, the sampling time of FIN (RST) is delayed by t_d after SYN is sampled, where t_d is so chosen that a significant portion of connections requested during the SYN sampling period terminate in the corresponding FIN (RST) sampling period. Internet traffic measurements [56] have shown that most of TCP connections last 12–19 seconds, so we set the sampling delay t_d to 10 seconds. In contrast, since most RTTs are less than 0.5 second, we start the collection of SYNs at the out-bound sniffer and SYN/ACKs at the in-bound sniffer simultaneously. To balance the detection resolution and the algorithm's stability and accuracy, we set t_0 to 10 seconds. Note, however, that both parameters are tunable and our algorithm is not very sensitive to this choice.

5.2 Normal Protocol Behavior

The three sets of traces represent the normal protocol behaviors at the exchange points between different stub networks and the Internet at different times. We parse the traces and extract the TCP SYN, SYN/ACK, FIN and RST packets from the TCP traffic.

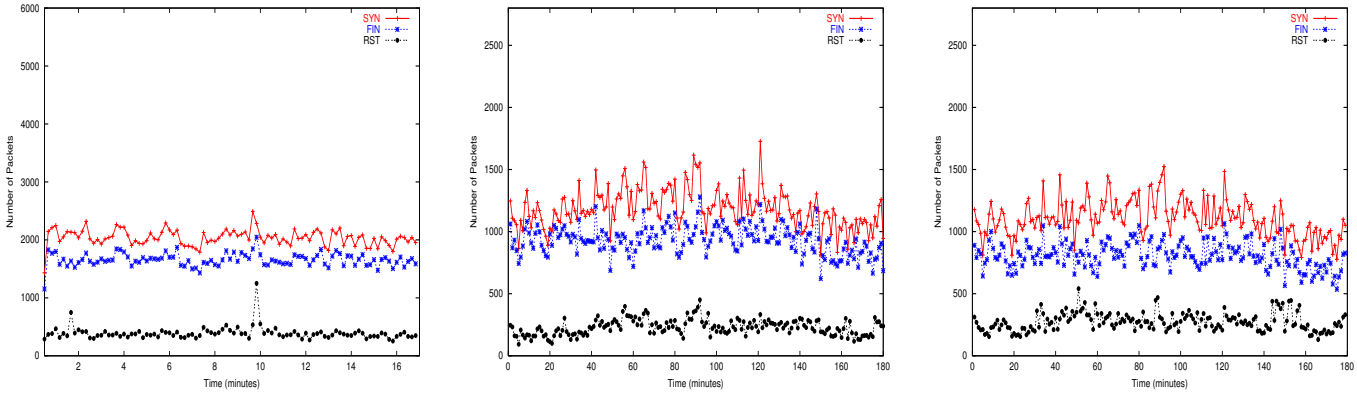
5.2.1 SYN vs. FIN Pairs

The dynamics of "generalized" SYNs that include SYNs and SYN/ACKs, FIN and RST packets at the DEC site are illustrated in Figure 4 (a), and the corresponding result from the Harvard trace is illustrated in Figure 4 (b). Those from UNC-in and UNC-out are plotted in Figures 4 (c) and 5 (a), and Auckland-in and Auckland-out are shown in Figures 5 (b) and (c), respectively. They clearly show the



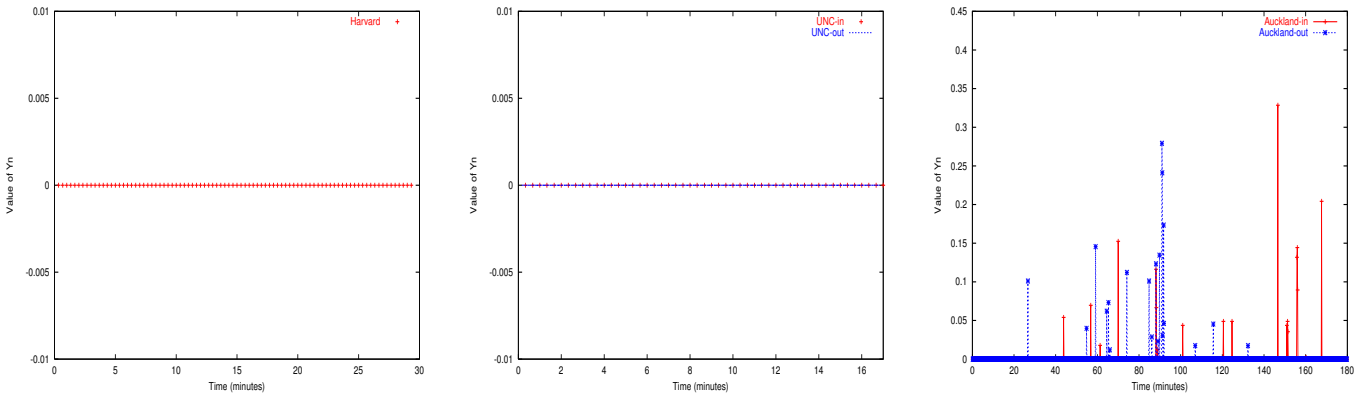
(a) DEC (b) Harvard (c) UNC-in

Figure 4: The dynamics of SYN and FIN (RST) packets (part I)



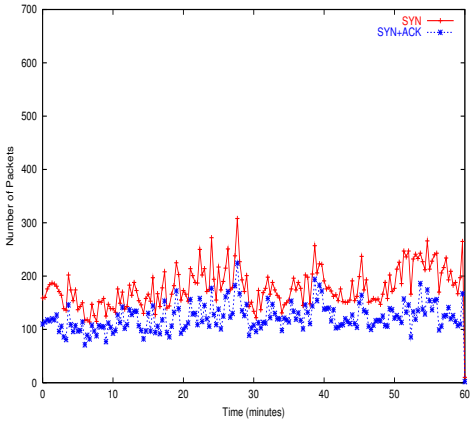
(a) UNC-out (b) Auckland-in (c) Auckland-out

Figure 5: The dynamics of SYN and FIN (RST) packets (part II)

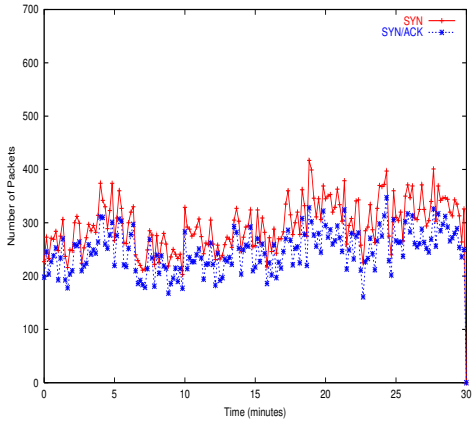


(a) Harvard (b) UNC (c) Auckland

Figure 6: CUSUM test statistics under normal operation

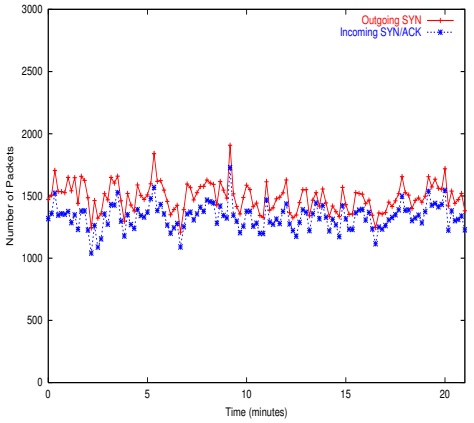


(a) DEC

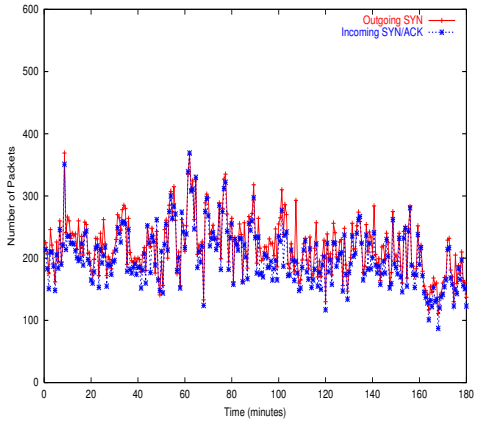


(b) Harvard

Figure 7: The dynamics of SYN and SYN/ACK packets at DEC and Harvard



(a) UNC



(b) Auckland

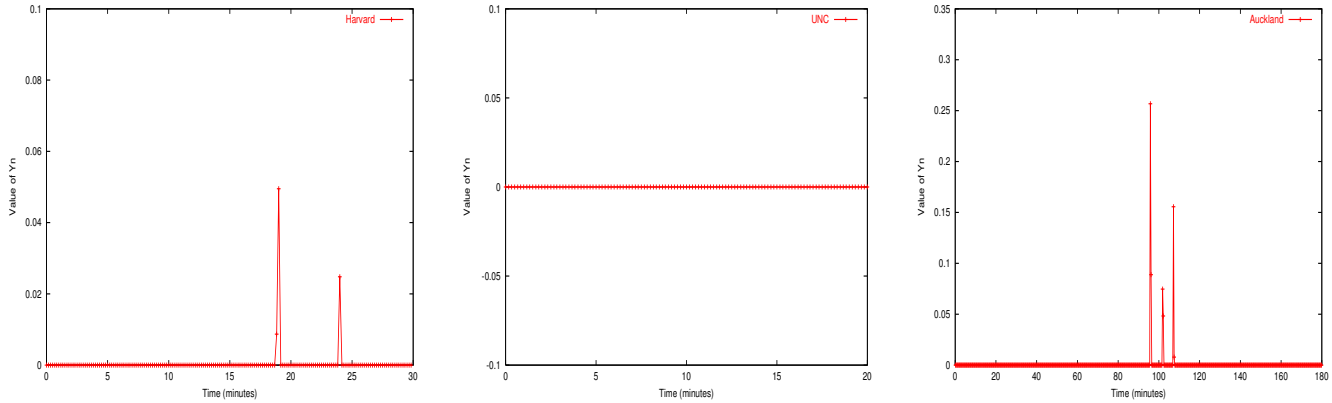
Figure 8: The dynamics of SYN and SYN/ACK packets at UNC and Auckland

consistent synchronization between SYN and FIN (RST) packets. The consistency indicates that the synchronization is an inherent protocol behavior and independent of time and sites.

We have applied the CUSUM algorithm on all the available traces without injecting flooding traffic. The test statistics, $\{y_n\}$, for all traces are plotted in Figure 6. For the Harvard and UNC traces, y_n's are constantly zeros. For the Auckland traces, more than 99% y_n's stay at zero. The isolated bursts in y_n are always much smaller than the threshold $N = 1.05$: the maximal spikes of y_n in Auckland-in and Auckland-out are 0.32 and 0.27, respectively. So, no false alarms are reported.

5.2.2 SYN—SYN/ACK pair

The dynamics of SYN and SYN/ACK packets at the DEC and Harvard sites are illustrated in Figures 7 (a) and (b), respectively. The outgoing SYNs and incoming SYN/ACKs from the UNC and Auckland



(a) Harvard (b) UNC (c) Auckland

Figure 9: CUSUM test statistics under normal operation at: Harvard, UNC and Auckland

traces are shown in Figures 8 (a) and (b), respectively. As with SYN vs. FIN pairs, these figures clearly demonstrate a consistent positive correlation between SYN and SYN/ACK packets. The consistency indicates that the strong positive correlation is also a distinct protocol behavior and independent of time and sites. Note that in the figures of the DEC and Harvard traces, SYNs and SYN/ACKs are collected from both directions, instead of "Outgoing SYN" and "Incoming SYN/ACK" as shown in the UNC and Auckland traces.

Also, we have applied the CUSUM algorithm on the Harvard, UNC and Auckland traces without adding flooding attacks. The test statistics, $\{y_n\}$, for the Harvard and UNC traces are plotted in Figures 9 (a) and (b); that for the Auckland trace is plotted in Figure 9 (c). As expected, for all the traces tested, y_n's are mostly zeros. Among the isolated spikes of y_n in the Harvard trace, the maximum is about 0.05; the maximal spike of y_n in the Auckland trace is about 0.26. Both are much smaller than the flooding threshold $N = 1.05$. So, no false alarms are reported.

In summary, under the normal condition, the difference between the collected number of SYNs and FINs (RSTs) or SYN/ACKs is very small, as compared to the total number of TCP connection requests. This observation holds in spite of the fact that the total number of TCP connection requests may be bursty on a small time scale, and slowly-varying on a large time scale. In other words, the correlation between the numbers of SYNs and FINs (RSTs) or SYN/ACKs is not sensitive to the request arrival process. The consistent synchronization between SYNs and FINs (RSTs) or SYN/ACKs is independent of the sites and time-of-day.

5.3 SYN Flooding Detection

With the appearance of Trinoo, which only implements UDP packet flooding, many tools have been developed to create DDoS attacks. Most of them, such as Tribe Flood Network (TFN), TFN2K, Stacheldraht, Trinity, Plague and Shaft, generate TCP SYN flooding attacks and randomize all 32 bits of the source IP address [12, 13]. Although these DDoS attack tools employ different ways to coordinate attacks with the goal of achieving robust and covert DDoS attacks, their flooding behaviors are similar in that the SYN packets are continuously bombarded to the victim.

Under SYN flooding attacks, the flooding SYN traffic has significant regularity and semantics that can be filtered out. The experiments with SYN attacks on commercial platforms [39] have shown that the minimum flooding rate to overwhelm an unprotected server is 500 SYN packets per second. However, with a specialized firewall designed to resist against SYN flooding, a server can withstand an attack whose flooding rate is up to 22,000 SYN packets per second [39]. To bring down the victim server for 10 minutes, for example, attackers must collectively inject at least 300,000 SYN packets. During the same time period, however, the numbers of counted FINs (RSTs) and SYN/ACKs remain largely unchanged. Therefore, there will be much more SYNs than FINs (RSTs) or SYN/ACKs collected during the flooding period. The difference between the numbers of SYNs and FINs (RSTs) or SYN/ACKs will increase dramatically, and remain large during the whole flooding period, which typically lasts for several minutes [37].

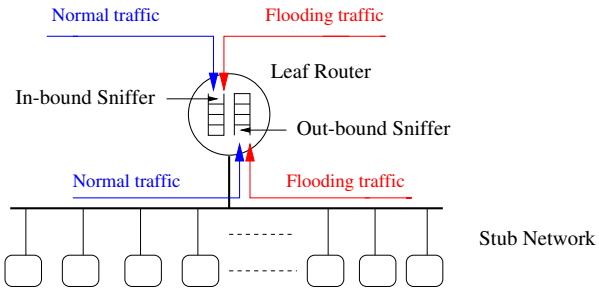
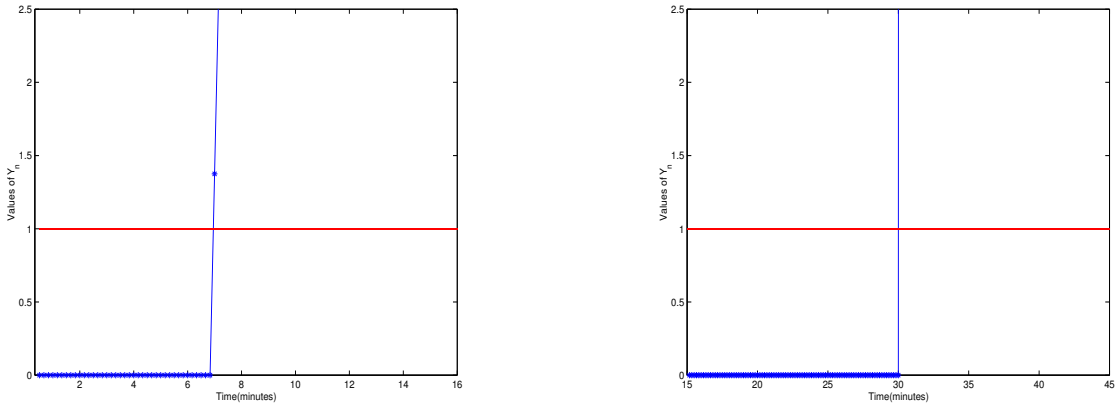


Figure 10: The trace-simulation flooding attack experiment

In the SYN flooding detection experiments, the UNC and Auckland 2000 traces are used as the normal background traffic. Among them, UNC-in or Auckland-in is used for incoming background traffic, and UNC-out or Auckland-out is for outgoing background traffic. The flooding traffic is mixed with the normal traffic, and the CPM at the leaf router is simulated, as shown in Figure 10. Because the non-parametric CUSUM method is used for detection of flooding attacks, the flooding traffic pattern or its transient behavior (bursty or not) does not affect the detection sensitivity. Rather, the detection



(a) UNC case (b) Auckland case

Figure 11: SYN flooding detection sensitivity at the last-mile CPM

sensitivity depends only on the total volume of flooding traffic. So, without loss of generality, we assume that the flooding rate is constant.

In DDoS attacks, the flooding traffic seen by the first-mile and the last-mile CPMs is quite different. The flooding traffic passing through the last-mile CPM is the aggregation of the flooding traffic from all distributed flooding sources, allowing for much easier detection of an attack. However, the flooding detection at the first-mile CPM is much more difficult. In a large-scale DDoS attack, the flooding sources can be so coordinated that the traffic from each flooding source may not be noticeable at all. Suppose the minimum SYN flooding traffic to bring down a TCP server is V packets per second. Then, the flooding rate at the last-mile CPM is V, but the flooding rate seen by the first-mile CPM may be much smaller than this.

We assume that the flooding traffic is evenly distributed among different flooding sources and there is only one flooding source inside each stub network. The flooding rate seen by the first-mile CPM, f_i, equals the individual flooding rate inside the same stub network. Therefore, f_i is determined by $\frac{V}{A_s}$, where A_s is the total number of the stub networks that contain flooding sources. This setting is intended to "hide" the attack from the first-mile CPM. That is, the less the flooding sources inside the stub network, the less flooding traffic seen by the first-mile CPM and the harder to detect the flooding attack. The flooding duration in all experiments is set to 10 minutes, a typical attack duration observed in the Internet [37]. The starting time of flooding attacks in the UNC traces is randomly chosen between 1 and 9 minutes, but the starting time in the Auckland traces lies between 3 and 166 minutes.

We first examine the detection sensitivity at the last-mile CPM, which employs SYN vs. FIN pairs as its detection method. To demonstrate the high sensitivity of last-mile CPM to SYN flooding, the

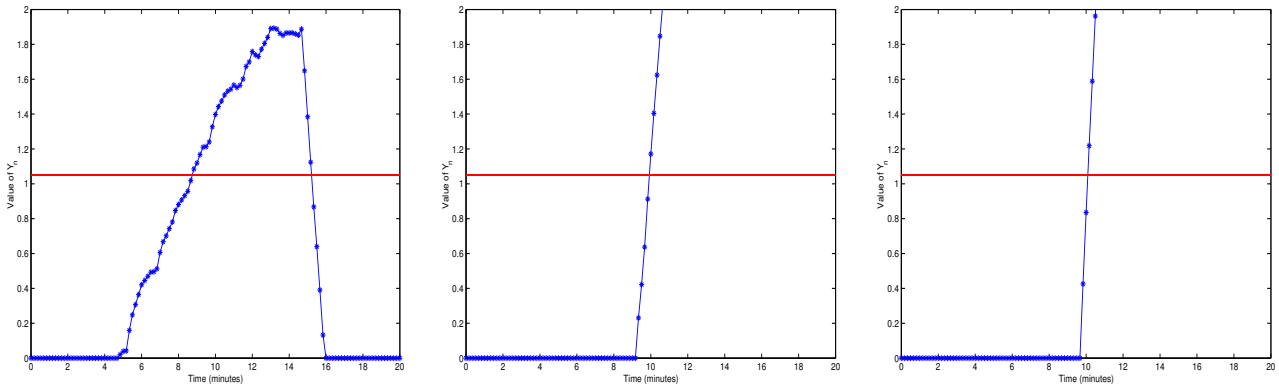
flooding rate V is set to its minimum, 500 SYNs per second. The simulation results are plotted in Figures 11 (a) and (b), showing that the cumulative sum y_n exceeds the flooding threshold "1" in one observation period, i.e., the fastest response can be achieved, in the Auckland and UNC trace cases, respectively. So, the last-mile CPM of the Auckland case can detect the SYN flooding attack less than 10 seconds, and so does the last-mile of the UNC case. Once the flooding attack is detected, a defense system like SynDefender can be triggered to protect the victim from the flooding attack. To paralyze the defense system at the victim, attackers have to increase their flooding rate, and the first-mile CPM will then be more likely to detect and locate the flooding sources inside the stub network.

To examine the detection sensitivity of the first-mile CPM, which employs SYN vs. SYN/ACK pairs to detect attacks, we vary the flooding rate f_i seen by the first-mile CPM, i.e., the individual flooding rate inside the stub network. As the last-mile detection is much easier than the first-mile detection, we only study the detection probability and detection time for the latter. We conduct the SYN flooding detection experiments on the UNC and Auckland traces.

5.3.1 The UNC Case

Using the UNC traces as the background traffic, we observe the dynamics of y_n. Figures 12 (a), (b) and (c) plot the dynamic behaviors of y_n when f_i is set to 35, 60 and 80 SYNs per second, respectively. The accumulative effects of SYN flooding are clearly shown in these figures. In the cases of 60 and 80 SYNs per second, the first-mile CPM can detect the SYN flooding attack in 4 and 2 observation periods, respectively. However, in the case of 35 SYNs per second, the first-mile takes a much longer time (about 24 observation periods, i.e., 4 minutes) to exceed the flooding threshold of 1.05. The detection performance of the first-mile CPM in the context of the UNC traces is summarized in Table 2, which lists the detection probabilities and detection times for different f_i values. Note that the units of detection time are measured in number of the observation period t_0, which is set to 10 seconds.

Clearly, larger flooding rates lead to faster and easier detection of attacks. According to Eq. (10), the lower detection bound is about 37 SYNs per second in this simulation scenario. If we implement the same CPM at a smaller subnet, $\bar{R}$ — the average number of incoming SYN/ACKs — will be smaller, so we can achieve higher detection sensitivity. This is confirmed by the study of the Auckland traces, which is presented in the next section.



(a) 35 SYNs per second (b) 60 SYNs per second (c) 80 SYNs per second

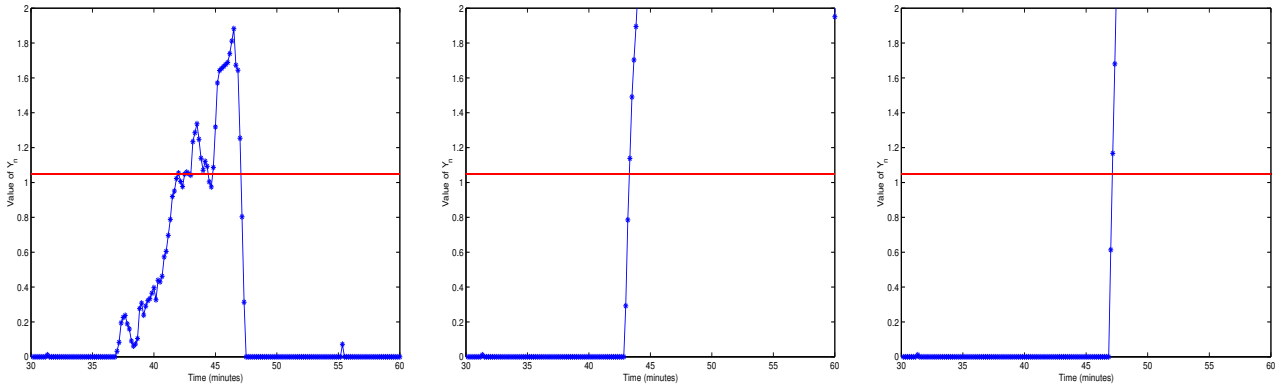
Figure 12: SYN flooding detection sensitivity of the CPM at UNC

Table 2: Detection Performance of the first-mile CPM at UNC

f_i	Detection Prob.	Detection Time
35	0.8	24.43
37	1.0	18.75
40	1.0	12.25
50	1.0	5.95
60	1.0	4.05
70	1.0	2.80
80	1.0	2.05
100	1.0	1.45
120	1.0	1.0

5.3.2 The Auckland Case

In the case of Auckland traces, the dynamic behaviors of y_n are illustrated in Figure 13 when f_i is set to 1.5, 3 and 4 SYNs per second, respectively. In the case of 1.5 SYNs per second, the first-mile CPM can detect the SYN flooding attack in about 27 observation periods. In contrast, at the flooding rate of 3 or 4 SYNs per second, the first-mile CPM takes a much shorter time (3 or 2 observation periods, respectively) to detect the ongoing flooding. The detection performance of the first-mile CPM for the context of Auckland traces is summarized in Table 3. Since $\bar{R}$ of the Auckland trace is much smaller than that of the UNC trace, the lower detection bound is reduced significantly from 35 to 1.5 SYNs per second.



(a) 1.5 SYNs per second (b) 3 SYNs per second (c) 4 SYNs per second

Figure 13: SYN flooding detection sensitivity of the first-mile CPM at Auckland

Table 3: Detection Performance of the first-mile CPM at Auckland

f_i	Detection Prob.	Detection Time
1.5	0.65	27.1
1.75	1.0	13.8
2	1.0	8.0
2.5	1.0	4.1
3	1.0	2.5
4	1.0	1.5
5	1.0	1.0

5.3.3 Discussion

From the detectable flooding rate, we can determine the efficacy of CPM in detecting distributed flooding attacks. To bring a protected server down, the aggregate flooding rate V should be larger than 22,000 requests per second [39]. In the UNC case, the lower detection bound is 35, and A_s can be as large as 628 stub networks like the UNC case. Considering the fact that the UNC stub network consists of over 35,000 users [49], it clearly demonstrates the utility and power of CPM. In the Auckland case, the lower detection bound is 1.5, and hence, A_s has to be as large as 14,666 medium-size stub networks like the Auckland case. Source address spoofing requires that the attacking software open a *raw* network socket, so the attacker must have root access on end hosts. Although the attacker can simultaneously initiate the flooding attacks from (possibly many) machines in several ISPs, it is much harder to launch attacks from hundreds or even tens of thousands of stub networks due to access limit.

We set the parameters independently of network size and traffic pattern, but the network administrator of the involved leaf router can incorporate site-specific information so that the CPM algorithm can achieve a higher detection sensitivity. For instance, in the UNC case, we can reduce a, the upper bound

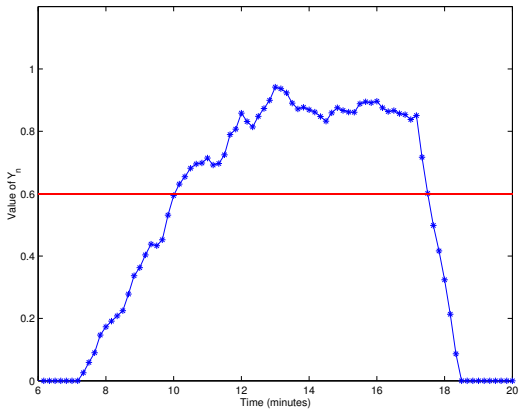


Figure 14: Improvement of flooding detection sensitivity

in case of normal operation, from 0.35 to 0.2 and N, the flooding threshold, from 1.05 to 0.6 without incurring additional false alarms. Then, the lower detection bound f_{min} decreases from 35 to 15 SYNs per second, and the detection sensitivity is greatly improved. The dynamics of y_n for the case $f_i = 15$ is shown in Figure 14.

In summary, CPM not only achieves fast detection and high detection accuracy, but is also easily implementable and broadly applicable.

6 Related Work

Over the past several years, a number of countermeasures have been proposed and implemented to detect, defense and trace-back DoS attacks. Defense mechanisms are deployed either at routers to block the prorogation of DoS traffic, or at victim servers to mitigate flooding attacks. The router-based mitigation systems include Distributed Packet Filtering (DPF) [40], Ingress Filtering [14], Pushback [25, 35], Rate Throttling [61] and SAVE [33]. There are commercial products such as Mazu's Enforcer [38] and Arbor Network's Peakflow [23] to block the DoS traffic at either the enterprise-network perimeter or the ISP edge routers. In parallel with these, many defense mechanisms have been installed at victim servers or their proximate firewalls to withstand DoS attacks, such as Client Puzzle [27, 59], Defensive Programming [44], Escort [52], Hop-count Filtering (HCF) [26], Path identifier (Pi) [60], Syn cache [32], Syn cookies [4] and Synkill [48]. Also, there are commercial products available to defend Internet servers against the flooding attacks, such as CheckPoint's SynDefender [34] and Netscreen's Syn proxying [24].

Since the source addresses of flooding packets are spoofed, it is very difficult to uncover the origin of a flooding source. To identify the compromised end-hosts that directly generate flooding packets and

the network path that these packets take, various traceback techniques [3, 7, 11, 47, 50, 51] have been proposed. By marking packets at intermediate routers, IP traceback [47, 51] and ICMP traceback [3] reconstruct the path to the flooding source. The controlled flooding technique developed by Burch and Cheswick [8] infers the attacking path by observing the impact of selectively exhausting some network links upon the victim. Using noisy polynomial reconstruction, Dean *et. al* [11] proposed an algebraic approach to performing IP traceback. A hash-based IP traceback technique [50] can identify the origin of individual packets with reasonable overhead through the use of Bloom filters. Moreover, an overlay network with selective rerouting has been used to track and prevent DoS traffic [28, 55].

As the first step to thwart DoS attacks, an accurate, fast and lightweight detection mechanism is essential for the defense and traceback systems. Based on traffic behaviors, several DoS detection mechanisms have been developed, including MULTOPS [17] and D-WARD [36]. Being deployed at source-end networks, D-WARD [36] monitors two-way traffic between the network and the rest of the Internet. By comparing the monitored traffic with normal traffic models, D-WARD detects and throttles the ongoing flooding attacks. Its normal traffic models are simply based on flow rates. In the design of MULTOPS [17], a tree of nodes are built to keep packet-rate statistics for subnets at different aggregation levels. Based on the observation of a significant disproportional difference between the traffic flowing in and out of the victim, routers use MULTOPS to detect ongoing bandwidth attacks. However, the burstiness of Internet traffic [31, 43] makes this detection of attacks much harder, since there is no natural length of burst for self-similar traffic. Furthermore, the normal Internet traffic pattern is site- and time-dependent. With the diversity of user behaviors and the emergence of new network applications, it is very difficult to build a robust and general model for describing the normal traffic flow rates.

Within the scope of more general intrusion detection, many different approaches have been proposed to detect anomalies, such as machine learning, neural networks, state machine model and Markov-chain model. In this paper, we detect the occurrence of a DoS attack as an instance of sequential change-point, and apply non-parametric CUSUM for detecting the change-point. Other change detection methods, which directly apply to raw IP traffic without deriving protocol behaviors for detecting network anomalies, have also been introduced, e.g., wavelet-based [1], spectrum-based [21], sketch-based [29] and signal processing approaches [57].

Unlike the above-mentioned schemes and commercial products, CPM extracts the inherent protocol behaviors from the raw IP traffic and detects DoS attacks based on the protocol behaviors. Since the protocol behaviors are much more stable than those of Internet traffic, CPM is much less sensitive to

site and traffic patterns than the other schemes. Moreover, no per-flow state is required by the CPM. It only keeps track of a few packet counts. By applying the non-parametric CUSUM method, CPM can detect the flooding attacks in a timely manner with low computational overhead as shown in our trace-driven simulations. Overall, the robustness of CPM results from its simplicity and reliance on the protocol behaviors.

Note that all of the other detection, defense and traceback mechanisms deployed at routers or firewalls were used solely for countering DoS attacks, and do not improve (some may even degrade) the end-to-end performance of clients behind routers, thus giving little incentive for wide deployment. In contrast, CPM is, in some sense, a by-product of the router infrastructure that can provide fine-grain packet classification and service differentiation [58]. As CPM differentiates control packets (such as TCP SYNs) from data packets, end-to-end performance can be improved significantly as shown in [58]. Therefore, CPM benefits not only victim servers but also the clients inside the stub network, making it attractive for wide deployment.

7 Conclusion and Future Work

We developed and evaluated a simple and robust mechanism, CPM, to detect DoS flooding attacks. CPM utilizes the inherent network protocol behaviors that are invariant under various arrival models and independent of sites and time-of-day. The distinct features of CPM include: (1) it is stateless and requires low computation overhead, making itself immune to any flooding attacks; (2) the non-parametric CUSUM method is employed, making the detection robust; and (3) it is insensitive to sites and traffic patterns. CPM can be installed at either firewalls or leaf (ISP edge) routers.

As a case study, the efficacy of CPM is evaluated and validated by detecting SYN flooding attacks. Traces from different sites and collected at different times have clearly demonstrated the SYN pairs' behaviors. Then, we conducted trace-driven simulations. The experimental results show that the CPM achieves high detection accuracy and short detection time. Moreover, once the first-mile CPM detects the ongoing flooding traffic, this information can help reveal the origin of flooding sources.

Recently, multi-homed ASs become attractive to improve availability, reliability and load-balancing. In such a case, a customer network is connected to the Internet by multiple ISPs. As long as the packets that belong to the same session go through the same leaf (ISP edge) router, CPM still works. However, if the packets of the same session go through different leaf routers, we need a loose synchronization mechanism between the CPMs in these leaf (ISP edge) routers, which is the subject of our future work.

Acknowledgment

We would like to thank Dong Lin for Harvard traces, Kevin Jeffay for UNC traces and Klaus Mochalski for Auckland traces. Also, comments from Cheng Jin at CalTech are gratefully acknowledged. Finally, we thank the anonymous reviewers for their constructive suggestions. The work reported in this paper was supported in part by the Office of Naval Research under Grant No. N00014-04-10726 and the NSF under Grant No. CCR-0329629.

References

[1] P. Barford, J. Kline, D. Plonka, and A. Ron. A signal analysis of network traffic anomalies. In *Proceedings of ACM Internet Measurement Workshop'2002*, Marseille, France, November 2002.

[2] M. Basseville and I. V. Nikiforov. *Detection of Abrupt Changes : Theory and Application*. Prentice Hall, 1993.

[3] S. M. Bellovin. ICMP traceback messages. In *Internet Draft: draft-bellovin-itrace-00.txt (work in progress)*, March 2000.

[4] D. J. Bernstein and Eric Schenk. Linux kernel SYN cookies firewall project. http://www.bronzesoft.org/projects/scfw.

[5] S. Bhattacharyya, C. Diot, J. Jetcheva, and N. Taft. Pop-level and access-link-level traffic dynamic in a tier-1 POP. In *Proceedings of ACM Internet Measurement Workshop'2001*, San Francisco, CA, November 2001.

[6] B.E. Brodsky and B.S. Darkhovsky. *Nonparametric Methods in Change-point Problems*. Kluwer Academic Publishers, 1993.

[7] H. Burch and B. Cheswick. Mapping the internet. *IEEE Computer*, 32(4), 1999.

[8] H. Burch and B. Cheswick. Tracing anonymous packets to their approximate source. In *Proceedings of USENIX LISA'2000*, New Orleans, LA, December 2000.

[9] R. Caceres, P. B. Danzig, S. Jamin, and D. J. Mitzel. Characteristics of wide-area TCP/IP conversations. In *Proceedings of ACM SIGCOMM '91*, Zurich, Switzerland, September 1991.

[10] W. S. Cleveland, D. Lin, and D. Sun. IP packet generation: statistical models for TCP start times based on connection-rate superposition. In *Proceedings of ACM SIGMETRICS '2000*, Santa Clara, CA, June 2000.

[11] D. Dean, M. Franklin, and A. Stubblefield. An algebraic approach to IP traceback. *ACM Transactions on Information and System Security*, 5(2), May 2002.

[12] S. Dietrich, N. Long, and D. Dittrich. Analyzing distributed denial of service tools: The shaft case. In *Proceedings of USENIX LISA'2000*, New Orleans, LA, December 2000.

[13] D. Dittrich. Distributed denial of service (DDoS) attacks/tools page. http://staff.washington.edu/dittrich/misc/ddos/.

[14] P. Ferguson and D. Senie. Network ingress filtering: Defeating denial of service attacks which employ ip source address spoofing. In *RFC 2267*, January 1998.

[15] L. Garber. Denial-of-service attack rip the internet. *Computer*, April 2000.

[16] S. Gibson. Distributed reflection denial of service. In *Technical Report, Gibson Research Corporation*, February 2002. http://grc.com/dos/drdos.htm.

[17] T. M. Gil and M. Poletter. MULTOPS: a data-structure for bandwidth attack detection. In *Proceedings of USENIX Security Symposium'2001*, Washington D.C, August 2001.

[18] P. Gupta and N. McKeown. Packet classification on multiple fields. In *Proceedings of ACM SIGCOMM '99*, Cambridge, MA, September 1999.

[19] M. Handley, V. Paxson, and C. Kreibich. Network intrusion detection: evasion, traffic normalization, and end-to-end protocol semantics. In *Proceedings of USENIX Security Symposium'2001*, Washington D.C, August 2001.

[20] U. Hengartner, S. Moon, R. Mortier, and C. Diot. Detection and analysis of routing loops in packet traces. In *Proceedings of ACM Internet Measurement Workshop'2002*, Marseille, France, November 2002.

[21] A. Hussain, J. Heidemann, and C. Papadopoulos. A framework for classifying denial of service attacks. In *Proceedings of ACM SIGCOMM '2003*, Karlsruhe, Germany, August 2003.

[22] G. Iannaccone, C.-N. Chuah, R. Mortier, S. Bhattacharyya, and C. Diot. Analysis of link failures in an IP backbone. In *Proceedings of ACM Internet Measurement Workshop'2002*, Marseille, France, November 2002.

[23] Arbor Networks Inc. Peakflow. Available: http://arbornetworks.com/standard?tid=34&cid=14.

[24] Netscreen Inc. Netscreen 100 firewall appliance. Available: http://www.netscreen.com/.

[25] J. Ioannidis and S. M. Bellovin. Implementing pushback: Router-based defense against DDoS attacks. In *Proceedings of NDSS'2002*, San Diego, CA, February 2002.

[26] C. Jin, H. Wang, and K. G. Shin. Hop-count filtering: An effective defense against spoofed DDoS traffic. In *Proceedings of ACM CCS '2003*, October 2003.

[27] A. Juels and J. Brainard. Client puzzle: A cryptographic defense against connection depletion attacks. In *Proceedings of NDSS'99*, February 1999.

[28] A. D. Keromytis, V. Misra, and D. Rubenstein. SOS: Secure overlay services. In *Proceedings of ACM SIGCOMM '2002*, Pittsburgh, PA, August 2002.

[29] B. Krishnamurthy, S. Sen, Y. Zhang, and Y. Chen. Sketch-based change detection: Methods, evaluation, and applications,. In *Proceedings of ACM Internet Measurement Conference'2003*, Miami, FL, October 2002.

[30] T.V. Lakshman and D. Stiliadis. High speed policy-based packet forwarding using efficient multi-dimensional range matching. In *Proceedings of ACM SIGCOMM '98*, Vancouver, Canada, September 1998.

[31] W. Leland, M. Taqqu, W. Willinger, and D. Wilson. On the self-similar nature of ethernet traffic. *IEEE/ACM Trans. on Networking*, 2(1), February 1994.

[32] J. Lemon. Resisting SYN flooding DoS attacks with a SYN cache. In *Proceedings of USENIX BSD-Con'2002*, San Francisco, CA, February 2002.

[33] J. Li, J. Mirkovic, M. Wang, P. Reiher, and L. Zhang. SAVE: Source address validity enforcement protocol. In *Proceedings of IEEE INFOCOM '2002*, New York City, NY, June 2002.

[34] Check Point Software Technologies Ltd. Syndefender. http://www.checkpoint.com/products/firewall-1.

[35] R. Mahajan, S. M. Bellovin, S. Floyd, J. Ioannidis, V. Paxson, and S. Shenker. Controlling high bandwidth aggregates in the network. *ACM Computer Communication Review*, 32(3), July 2002.

[36] J. Mirkovic, G. Prier, and P. Reiher. Attacking DDoS at the source. In *Proceedings of IEEE ICNP'2002*, Paris, France, November 2002.

[37] D. Moore, G. Voelker, and S. Savage. Inferring internet denial of service activity. In *Proceedings of USENIX Security Symposium'2001*, Washington D.C., August 2001.

[38] Mazu Networks. Enforcer. Available: http://www.mazunetworks.com/products/.

[39] R. Oliver. Countering SYN flood denial-of-service attacks. In *Tech Mavens, Inc*, August 2001. http://www.tech-mavens.com/synflood.htm.

[40] K. Park and H. Lee. On the effectiveness of route-based packet filtering for distributed DoS attack prevention in power-law internets. In *Proceedings of ACM SIGCOMM '2001*, San Diego, CA, August 2001.

[41] V. Paxson. Bro: A system for detecting network intruders in real-time. *Computer Networks*, 31(23-24), 1999.

[42] V. Paxson. An analysis of using reflectors for distributed denial-of-service attacks. *ACM Computer Communication Review*, 31(3), July 2001.

[43] V. Paxson and S. Floyd. Wide-area traffic: The failure of poisson modeling. *IEEE/ACM Transactions on Networking*, 3(3), June 1995.

[44] X. Qie, R. Pang, and L. Peterson. Defensive programming: Using an annotation toolkit to build dos-resistant software. In *Proceedings of USENIX OSDI'2002*, Boston, MA, December 2002.

[45] M. Roesch. Snort — lightweight intrusion detection for networks. In *Proceedings of USENIX LISA'99*, Seattle, WA, November 1999.

[46] K. A. Ross. *Elementary Analysis: The Theory of Calculus, 5th Edition*. Springer-Verlag Publishing Company, 1980.

[47] S. Savage, D. Wetherall, A. Karlin, and T. Anderson. Practical network support for IP traceback. In *Proceedings of ACM SIGCOMM '2000*, Stockholm, Sweden, August 2000.

[48] C. L. Schuba, I. V. Krsul, M. G. Kuhn, E. H. Spafford, A. Sundaram, and D. Zamboni. Analysis of a denial of service attack on TCP. In *Proceedings of IEEE Symposium on Security and Privacy*, May 1997.

[49] F. D. Smith, F. H. Campos, K. Jeffay, and D. Ott. What TCP/IP protocol header can tell us about the web. In *Proceedings of ACM SIGMETRICS '2001*, Cambridge, MA, June 2001.

[50] A. C. Snoren, C. Partridge, L. A. Sanchez, C. E. Jones, F. Tchakountio, S. T. Kent, and W. T. Strayer. Hash-based IP traceback. In *Proceedings of ACM SIGCOMM '2001*, San Diego, CA, August 2001.

[51] D. Song and A. Perrig. Advanced and authenticated marking schemes for IP traceback. In *Proceedings of IEEE INFOCOM '2001*, Anchorage, Alaska, March 2001.

[52] O. Spatscheck and L. Peterson. Defending against denial of service attacks in scout. In *Proceedings of USENIX OSDI'99*, New Orleans, LA, February 1999.

[53] V. Srinivasan, G. Varghese, S. Suri, and M. Waldvogel. Fast and scalable layer four switching. In *Proceedings of ACM SIGCOMM '98*, Vancouver, Canada, September 1998.

[54] W. R. Stevens. *TCP/IP Illustrated, Volume 1*. Addison-Wesley Publishing Company, 1994.

[55] R. Stone. CenterTrack: An IP overlay network for tracking DoS floods. In *Proceedings of USENIX Security Symposium'2000*, Denver, CO, August 2000.

[56] K. Thompson, G. J. Miller, and R. Wilder. Wide-area internet traffic patterns and characteristics. *IEEE Network*, 11(6), November/December 1997.

[57] M. Thottan and C. Ji. Anomaly detection in ip networks. *IEEE Transactions on Signal Processing*, 51(8), August 2003.

[58] H. Wang and K. G. Shin. Layer-4 service differentiation and resource isolation. In *Proceedings of IEEE RTAS'2002*, San Jose, CA, September 2002.

[59] X. Wang and M. Reiter. Defending against denial-of-service attacks with puzzle auctions. In *Proceedings of IEEE Symposium on Security and Privacy*, Oakland, CA, May 2003.

[60] A. Yaar, A. Perrig, and D. Song. Pi: A path identification mechanism to defend against DDoS attacks. In *Proceedings of IEEE Symposium on Security and Privacy*, Oakland, CA, May 2003.

[61] D. Yau, J. Lui, and F. Liang. Defending against distributed denial-of-service attacks with max-min fair server-centric router throttles. In *Proceedings of IWQoS'2002*, Miami Beach, FL, May 2002.