Available online at www.sciencedirect.com
ScienceDirect
J. Parallel Distrib. Comput. 68 (2008) 456–470
Journal of Parallel and Distributed Computing
www.elsevier.com/locate/jpdc

An autonomous defense against SYN flooding attacks: Detect and throttle attacks at the victim side independently☆

Bin Xiao[a,*], Wei Chen[b], Yanxiang He[c]

[a]*Department of Computing, Hong Kong Polytechnic University, Hong Kong*
[b]*Computer College, Nanjing University of Posts and Telecommunications, PR China*
[c]*Computer School, Wuhan University, PR China*

Received 25 July 2006; received in revised form 9 January 2007; accepted 19 June 2007
Available online 24 July 2007

Abstract

Distributed denial of service (DDoS) attacks seriously threaten Internet services yet there is currently no defence against such attacks that provides both early detection, allowing time for counteraction, and an accurate response. Traditional detection methods rely on passively sniffing an attacking signature and are inaccurate in the early stages of an attack. Current counteractions such as traffic filter or rate-limit methods do not accurately distinguish between legitimate and illegitimate traffic and are difficult to deploy. This work seeks to provide a method that detects SYN flooding attacks in a timely fashion and that responds accurately and independently on the victim side. We use the knowledge of network traffic delay distribution and apply an active probing technique (*DARB*) to identify half-open connections that, suspiciously, may not arise from normal network congestion. This method is suitable for large network areas and is capable of handling bursts of traffic flowing into a victim server. Accurate filtering is ensured by a counteraction method using IP address and time-to-live(TTL) fields. Simulation results show that our active detection method can detect SYN flooding attacks accurately and promptly and that the proposed rate-limit counteraction scheme can efficiently minimize the damage caused by DDoS attacks and guarantee constant services to legitimate users.

Keywords: DDoS attacks; SYN flooding; TTL; Early detection; Rate-limit counteraction

1. Introduction

Distributed denial of service (DDoS) [3] attacks, large-scale cooperative attacks launched from a large number of compromised hosts, are a major threat to Internet services. Popular web sites such as Yahoo, CNN, and Amazon, are among the well-known victims of DDoS attacks, yet an even larger number of online companies than these depend on the stability and availability of the Internet and face considerable losses should they be the object of a DDoS attack.

More than 90% of DDoS attacks exploit a system's transmission control protocol (TCP) [18,26]. The most efficient and commonly-used SYN flooding attacks [20,26] exploit the standard TCP three-way handshake in which the server receives a client's SYN (synchronization) request, replies with a SYN/ACK (synchronization/acknowledge) packet and then waits for the client to send the ACK (acknowledge) to complete the handshake. While waiting for the final ACK, the server maintains a half-open connection. Since the SYN flooding attacker always chooses unreachable addresses as the spoofed source addresses of the attacking packets [20], the server will not receive the anticipated final ACK from the client. As more and more half-open connections are maintained on a victim server, such DDoS attacks deplete the server's resources and the server will as a result refuse services even to legitimate customer requests.

Current DDoS defence methods can be classified as either cooperative or autonomous. A cooperative method requires cooperation between sub-systems distributed over a large scale network. Because DDoS attacks are often distributed, that is, the DDoS attack traffic is generated from numerous hosts spread all over Internet, a cooperative method is likely to detect and respond more effectively. Unfortunately, cooperative methods call for the wide deployment of defence systems

☆ This work is supported by HK RGC CERG PolyU 5311/06E, by NSF of Jiangsu High-Education 07KJB520079, and NSFC 60642006 China.
* Corresponding author. Fax: +852 2774 0842.
E-mail address: csbxiao@comp.polyu.edu.hk (B. Xiao).

0743-7315/$ - see front matter © 2007 Elsevier Inc. All rights reserved.
doi:10.1016/j.jpdc.2007.06.013

which may be dispersed across different domains supported by different ISPs, all potentially having distinct administrative strategies and security policies. This makes it very difficult to design a distributed defence system. Furthermore, since not every ISP will directly benefit from the deployment of a cooperative defence system, some ISPs are not very strongly motivated to cooperate. All of difficulties discourage researchers from seeking an efficient distributed cooperative defence method.

An autonomous method detects and responds to a DDoS attack at a single-point, whether it be on the victim's side, on the source side, or on an intermediate network. Research into victim-side defences is encouraged by the fact that victims are more willing to deploy the resources to defend system against DDoS attacks. Indeed, the majority of autonomous defences are set up on the victim side. A victim-side defence system should be timely, accurate, and robust, defined as follows:

(1) *Timely*: The system should detect a DDoS attack at an early stage before large amounts of attack traffic have already aggregated on the victim side.
(2) *Accurate*: Counteraction must be able to accurately distinguish between attack traffic and normal traffic. When filtering out attacking packets, the counteraction scheme should not drop valid requests.
(3) *Robust*: The system itself should be immune to DDoS attacks and able to operate during an attack. This means not only that a counteraction scheme should be accurate, but it should also be simple, and its execution should be inexpensive.

This paper provides an effective defence against SYN flooding. It is deployed entirely on the victim side and is autonomous, having no need for the cooperation or support of other infrastructure elements. This makes deployment much simpler. A server seeking immunity from DDoS attacks needs only to deploy our method independently on the server side.

Our approach consists of two components: an active probing detection component and a time-to-live (TTL)-based, rate-limit counteraction component. The detection component, instead of merely passively sniffing signatures, uses an active probing mechanism to capture attack signatures. An active mechanism thereby provides alerts that are both more accurate and more prompt, which makes it more suitable to detect the gradual pulse attack (the maximum rate is achieved gradually, maintained and gradually decreased) [12]. With the help of alerts provided by the detection component, the counteraction component limits the rate of malicious traffic by using the TTL field in an IP header. This TTL-based, rate-limit method is both more accurate and more reliable than DDoS filters or rate-limit methods which use IP and port information to establish filter rules. This is because in a SYN flooding attack it is possible for attackers to fake the source IP information in attacking packets. The routing path of a packet usually remains unchanged during the packet delivery from the attack source to the destination. Consequently, the TTL field, which indicates the remaining router hops before a packet is deleted, is a more reliable basis for a rate-limit.

The following summarizes the novel contributions made in this paper.

(1) An active probing scheme is used to diagnose the network traffic congestion status. We can quickly classify a half-open connection as either normal or abnormal from the knowledge of network traffic distribution. It is possible to obtain the network traffic distribution either using our active delay probing method *DARB* (DelAy pRoBing), or using the traffic delay history.
(2) To counteract malicious packets, this paper presents an efficient and reliable TTL-based scheme which produces little collateral damage. To minimize collateral damage, additional restrained rules are enforced that can distinguish normal packet from the spoofed.
(3) Little overhead is introduced for both the detecting and counteracting methods. We can even skip over the detecting method to jump directly to the counteracting method when there is heavy traffic jam on the neighbor links of a protected server or heavy operation load on a protected server. Thus, we keep our defence system away of being an attacking target.
(4) Our autonomous defence is easy to be deployed independently on the victim side and does not require the cooperation of other network infrastructures. Thus deployment is more independent and less difficult than other cooperative methods.

This paper is organized as follows. Section 2 introduces related work. Section 3 presents the first of the two proposed sequentially implemented components of our system, an active detection scheme. Section 4 presents the second component, a TTL-based counteraction scheme. Section 5 offers simulation results that show that our approach can accurately detect a SYN flooding attack at an early stage and limit the potential damage from a DDoS attack. We also compare our method with a previous passive detection method FDS [26]. Section 6 shows some discussions and Section 7 offers our conclusion.

2. Related work

Current countermeasures against DDoS attacks take a two-step, sequential, detect-and-respond approach. Detection usually involves trying to capture intrusion signatures and giving out accurate alarms when an attack starts. Wang et al. [26] detected SYN flooding attacks at leaf routers that connect end hosts to the Internet. This method was based on the fact that in normal network traffic the SYN and FIN pair should appear symmetrically. To accumulate these pairs, they used a non-parameter CUSUM method that needed to estimate the average duration time of TCP connections, which varied according to a network's status. This variance might adversely influence the accuracy of final results. Peng et al. [17] compiled an IP address database of previous successful connections. When a network was suffering from traffic congestion, an IP address that did not appear in the database was construed as more suspicious. Ref. [9] presented a more proactive method which employed an aggressive drop policy to identify attack traffic. The drop policy

dropped a small number of packets and monitored the client's transient response. By comparing the dropping response with that predicted by a formula, it was possible to detect malicious traffic. Jin's approach [8] utilized the TTL value in the IP header to estimate the hop-count of each packet. Spoofed packets could be distinguished by the fact that their hop-count differed from normal ones. Passive methods [17,26] such as these could produce accurate detection results only in the later stages of an attack, when attack signatures had become evident. Because passive methods are reliant on passively sniffing an attacking signature, such as traffic flow shift [9], periodic change of packet arrivals [4], and abnormal IP address appearance [17,21], they are inaccurate in the early stages of an attack. Given this, this paper proposes the use of an active approach to detection that uses a delay probing method, *DARB*, to provide early warning of DDoS attacks.

Response schemes, including filtering [23,24,27], traceback [5,15,19,21,22], and pushback [7] seek to minimize the influence of DDoS attacks and identify attacking sources. The RFC2827 method proposed in [6] can efficiently filter out outgoing packets containing spoofed source IPs on the attacking source side. Unfortunately, the RFC2827 method requires all network Internet service providers (ISPs) that are connected to the Internet to deploy ingress packet filtering. This imposes a non-negligible overhead on router performance, with the result that the RFC2827 method is not widely used. Wang et al. protected a server against DDoS attacks by using a finely grained quality-of-service (QoS) classifier and an adaptive weight-based resource management mechanism [25]. This mechanism can limit the suspicious traffic rate. Keromytis et al. [10] introduced a proactive approach which used aggressive packet filtering, an overlay network, and a scalable access control mechanism. Traceback techniques attempt to identify the real locations of attackers launching a large volume of useless traffic packets. Because these packets will use a spoofed source IP, most traceback schemes will seek to identify the genuine source location of an attack by marking packets along their routing paths or by sending special packets [1] and monitoring traffic flow. The real routing path can then be reconstructed and the true source IP located. In [19], the authors described a series of marking algorithms, both simple and sophisticated, including node appending, node sampling, and edge sampling. Their paper also discussed encoding and compression issues relevant to economical storage and improved performance. Generally, the more routers participate in marking, the more effective the traceback will be. Once the real path of the spoofed packets has been identified, it becomes possible to deploy pushback techniques and advanced filtering. Pushback is always performed at the last few routers before traffic reaches the target, treating the DDoS problem as a special congestion-control problem. When spoofed packets are detected and the real path of the malicious traffic is identified, the router requests adjacent upstream routers to reduce the amount of traffic from the specified aggregates [7]. The TTL-based counteraction scheme to protect victim servers that we present here offers another approach to limiting the attacking traffic. Syn cache and cookies method was evaluated in Lemon's work [11], the basic idea was to use cache or cookies to evaluate security of connection before establishing the real connection with protected server. Both of these two methods have deployed implementations and can efficiently tolerate the impact of SYN flooding attacks. But they do not attempt to detect SYN flooding attacks and prevent attacks from the source.

3. Active detection scheme and *DARB* technique

This section presents the first component of our approach, a detection scheme which ensures the efficient, early detection of DDoS attacks with the assistance of our active probing method, *DARB*, to capture attacking signatures. The first step in this process is to classify a half-open connection as either normal or abnormal, that is, to determine whether a half-open connection is simply a sign of normal network traffic congestion or is the product of a DDoS attack. This is done using the historical traffic delay to a specific destination (either an IP address or a domain) which can be obtained from Internet traffic history and knowledge of network traffic distribution or by using *DARB* to obtain the router delay by sending packets containing special TTL set at the IP headers. The probability of a half-open connection being the result of a malicious attack is then calculated and if necessary an alert is announced based on the accumulated probability results. The following describes and discusses this process in detail. Section 3.1 describes the distinction between normal and abnormal half-open connections and how they may be classified using *DARB*. In Section 3.2 we describe *DARB* in detail to obtain the traffic delay of a route indicated in a half-open connection. Section 3.3 presents functions to calculate the probability of a half-open connection belonging to abnormal connections. Section 3.4 shows the early detection scheme to launch alerts when DDoS attacks happen.

3.1. Normal and abnormal half-open connections

Central to our approach is the classification of half-open connections as either normal or abnormal. When TCPs establish a connection between a source and a destination prior to the transfer of data packets, they may use a normal three-way handshake as shown in Fig. 1(a). This involves the client first sending a $Syn(k)$ request to the server to initiate a connection. The server replies with a packet, which contains both an acknowledgement $Ack(k+1)$ and a synchronization request $Syn(j)$ (denoted hereafter as $Ack(k+1)+Syn(j)$). The client then completes the connection-building process by sending back the acknowledgment $Ack(j+1)$, with k and j being sequence numbers produced randomly by the server and client during the three-way handshake.

A *half-open connection*, then, is a connection state in which the server is waiting for the acknowledgement $Ack(j+1)$ from a client. This state is normally caused by an uncompleted TCP three-way handshake. For example, a half-open connection will be maintained on the server if the second handshake $Ack(k+1)+Syn(j)$ or the third handshake $Ack(j+1)$ is dropped along its routing path (Fig. 1(b)). In such a case, the server will try to

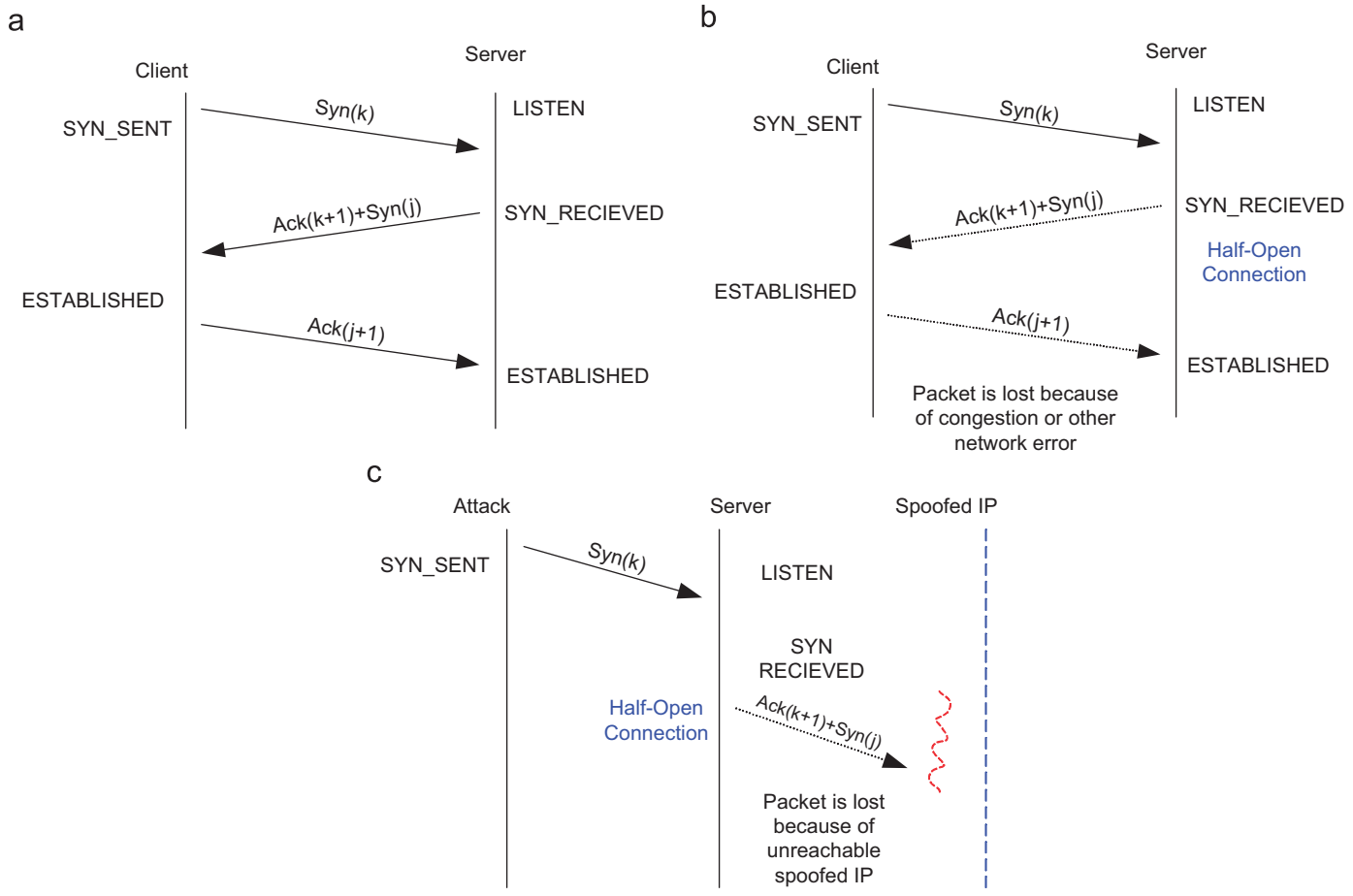


Fig. 1. Three-way handshake in a complete TCP connection and Half-Open connections. (a) Normal three-way handshake. (b) Normal Half-Open connection caused by network congestion. (c) Abnormal Half-Open connection caused by spoofed source IP.

complete the three-way handshake by resending $Ack(k+1)+Syn(j)$ packets. The object of this is to minimize the damage caused by network congestion and to improve the reliability of the three-way handshake.

Normal half-open connections are half-open connections caused by network congestion or other network errors. *Abnormal half-open* connections are those which can be observed on a victim server during DDoS attacks (e.g., a SYN flooding attack). In such cases, the attacker has sent packets with spoofed IP addresses to a victim server (Fig. 1(c)). After receiving a $Syn(i)$ request, the server transfers to the 'syn-received' state and sends an $Ack(k+1)+Syn(j)$ packet back to the destination according to the source IP. Since the source IP has been spoofed by the attacker, most of these $Ack(k+1)+Syn(j)$ packets cannot reach the specified destination. While this type of half-open connection generated by a SYN flooding is different from that caused by a network congestion, it is hard for the server itself to distinguish between them. As a result, the victim server maintains half-open connections and continues resending $Ack(k+1)+Syn(j)$ packets.

The key problem is to distinguish the *abnormal half-open* connection from the *normal half-open* connection so that the abnormal connection can immediately be released and ceases to consume server resources. As noted, most normal half-open connections arise from network congestion whereas abnormal half-open connections have no relevance to the short traffic delay that is seen between network routers in a normal environment. Network traffic congestion can be inferred from features such as increased packet delay, a high packet loss ratio, and a near-capacity queue at a congested router. If these signs can be detected, a given half-open connection is most probably caused by a traffic congestion and is therefore a normal half-open connection. If these signs are not present, a given half-open connection is regarded as an abnormal half-open.

One way to determine whether a half-open connection is normal or abnormal is to use *DARB* (for details, see Section 3.2) to probe the path delay in a half-open connection between a server and a client. A very long delay value is regarded as a sign of congestion, which in turn suggests that the tested half-open connection is normal. The absence of a long delay (inferred network congestion) will lead to the assumption that the half-open state is abnormal.

To test whether the network delay can be used as the sign of traffic congestion, we probed approximately 100 web sites from Hong Kong and Wuhan hop-by-hop and obtained the network delay. The average delay for a healthy network is about 100 ms. Fig. 2(a) shows the average hop-by-hop delay values of a healthy network. The delay values of the first several hops

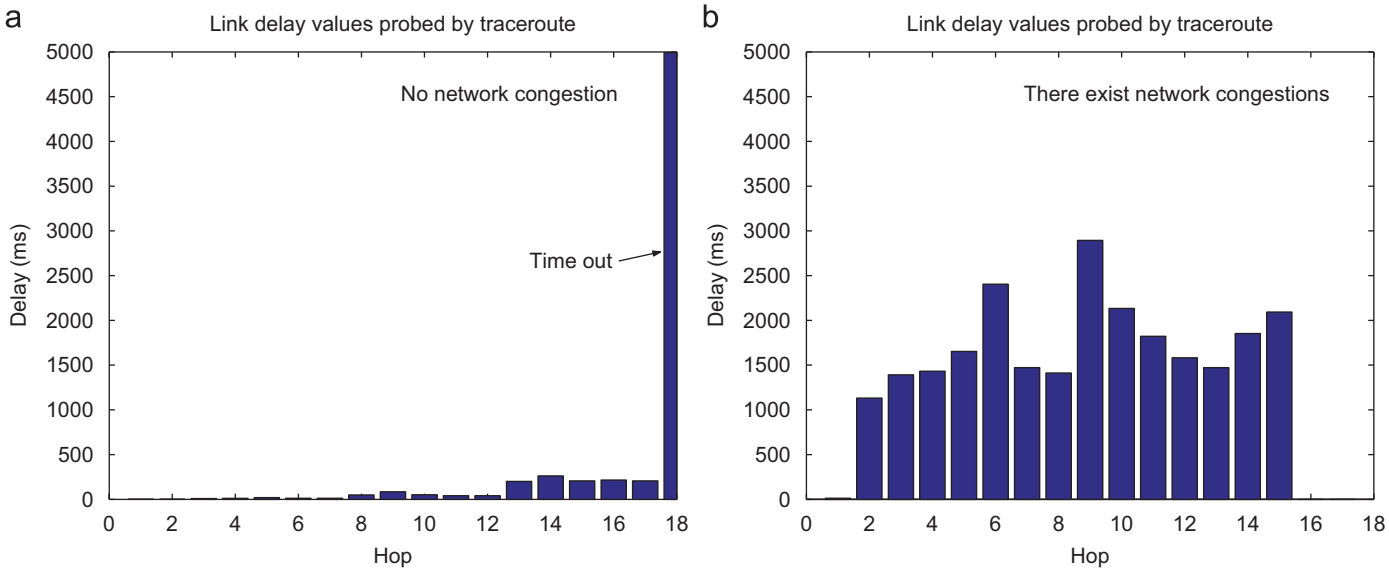


Fig. 2. Delay values probed hop-by-hop. (a) A healthy network. (b) A congested network.

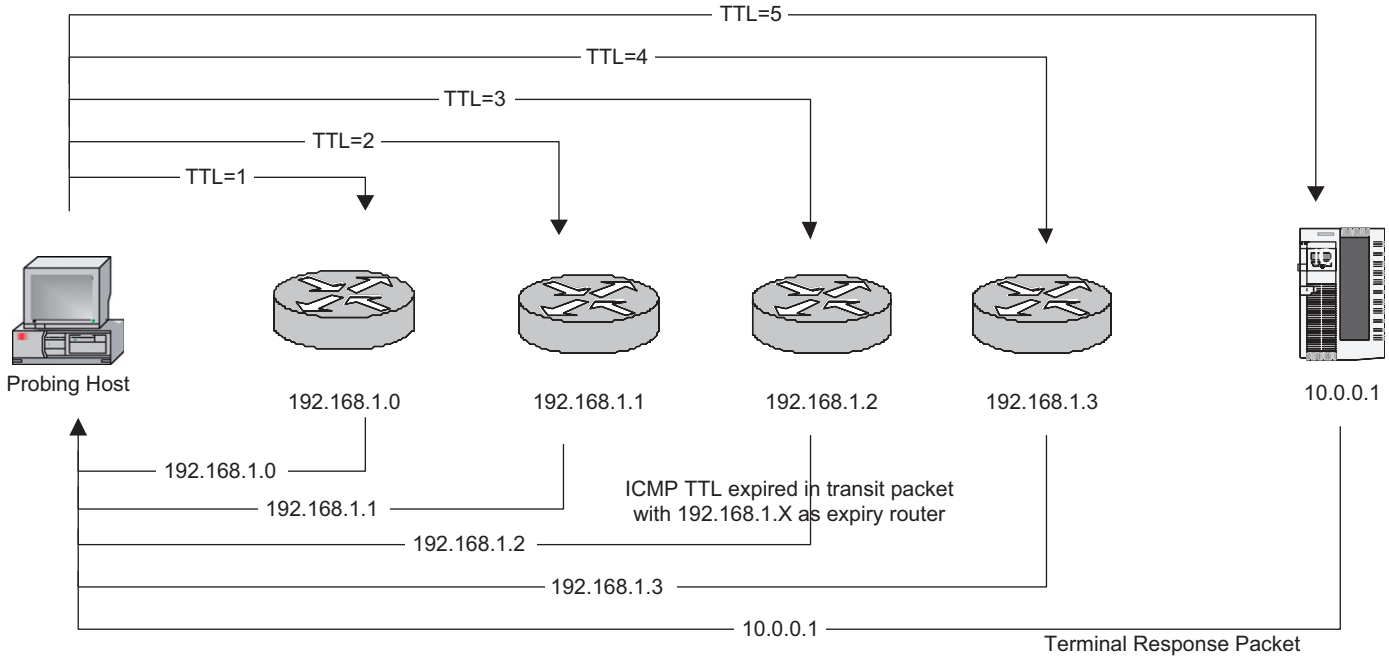


Fig. 3. *DARB* probes the network delay by setting special TTL values.

are always below 20 ms, which seems to be zero in the figure, and the delay values of middle hops is about 100–200 ms. If the probing result in times-out, we use an infinite value to substitute for the delay value. Timeout results are sometimes found at the last hops of delay probing because the probing packets may be filtered out by the firewall on the server side. Fig. 2(b) depicts the delay value of a congested network, which was infected by a worm virus. The average delay is far above 1000 ms. We were notified that the server at hop 2 was infected by a worm virus from ISPs. Then we tested the delay of paths that included hop 2 and found that the delay after hop 2 was far above 1000 ms. These experiments would indicate that the delay value can be used to distinguish a healthy network from a congested one.

3.2. The delay probing method: DARB

It is possible to obtain a path delay from a traffic history table or using a method called *DARB*. *DARB* traces outgoing paths toward network destinations by sending packets with special TTL fields in the IP layer and then recording their times of death, as shown in Fig. 3. The TTL value field in the IP packet

```
Probe(Input: A Half-Open Connection; Output: Delay Value)
Create Probing Packet P
Set P.TTL=128 //It is equal to a 'ping' operation
Send(P)
if Receive ICMP echo reply message M_e then
  Return 0 // The probed destination is a living host and
  // it can be safely regarded as the legitimate user
end if
//If cannot get a direct 'ping' result, do probing initialization
Set far_hop = 32
Set near_hop =the hops to victim's gateways router
Set current_hop = (far_hop + near_hop) / 2
Set delay = ∞
while far_hop > near_hop do
  Set P.TTL= current_hop
  Send(P)
  //If successfully get response from a selective router
  if Receive ICMP TTL response message M_t then
    delay = M_t response time
    near_hop = current_hop + 1
    current_hop = (far_hop + near_hop) / 2
    //If fail to get response
  else
    far_hop = current_hop -1
    current_hop = (far_hop + near_hop) / 2
  end if
end while
if delay = ∞ then
  Return -1 // Fail to probe any router along the path
else
  Return delay
end if
```

Fig. 4. The selective probing scheme.

header indicates the remaining router hops before the packet gets deleted. The IP TTL field limits the lifetime of packets transmitted across the Internet and is decreased by each forwarding device (router). If the TTL field reaches zero before arriving at the destination host, the router drops the offending packets and transmits an ICMP (Internet control message protocol) [1] 'TTL exceeded in transit' error message to the original host, informing the original host of the packet's timeout. If the packet has been created appropriately, the destination host should return a final packet to the original host when the packet reaches its destination. The time-stamps of both the sent-out packets and ICMP-replied packets are recorded to calculate the delay between the original host and each router. For example, if the TTL value of a packet is 3, the packet can be forwarded 3 times before it reaches the destination. Each time it is forwarded, the TTL value will decrease by 1. *DARB* is similar to `traceroute` [2], which works by sending packets with progressively longer TTL values. The `traceroute` technique sends arbitrary Layer 4 packets to a destination host with an IP TTL of 1 and then monotonically increments the TTL field after each response. Our proposed *DARB* method sets TTL values using a selective algorithm, which will be presented in the following paragraph.

DARB selects special routers along a routing path to probe delays rather than, like `traceroute`, probing all the routers hop-by-hop. The selection scheme shown in Fig. 4 is a binary search algorithm. Since this delay probing process will not guarantee the successful probing of all routers along the path, the scheme seeks to probe a router as far as possible. The **far_hop** defines the largest TTL value. The packet with this TTL value will hop to the farthest router in the delay probing process. The **near_hop** defines the nearest router. The **current_hop** indicates the current hop that the packet will traverse. Initially, the TTL is set large enough to ensure that the packet is able to reach the destination host that is obtained from the input half-open connection. If an ICMP echo reply message returns, it means the destination is a live host, rather than a non-existent or offline host whose IP a SYN flooding attacker has used as a spoofed address. In a SYN flooding attack, to ensure the attack efficiency, attackers prefer to use unreachable IPs as the spoofed IP. Otherwise, when a living host, whose IP is used as spoofed source IP, receives unexpected 'ACK/SYN' from a server, it will discard it or send a 'RST' back. Therefore, a live destination can be safely regarded as a legitimate user. If, on the other hand, there is no immediate reply from the tested destination host, a long delay will be probed on routers along the path. The probing packet with the TTL value **current_hop** will first seek a successful reply from a selected router. Upon receiving that, it will probe a farther router. If, however, it fails to get a reply, i.e., the request times out, it will try to probe a nearer router. This probing process will continue until it is possible to probe the available router. If the *DARB* fails to probe any delay along the routing path, the algorithm will return a value of -1 indicating that the probing process has failed. Otherwise the algorithm will return the delay value of the farthest router which can be successfully probed with *DARB*.

The advantages of using such a delay probing scheme is that it avoids the negative influences of firewalls or gateway filters. The simplest method is simply to send an ICMP echo request message to the destination as the `ping` request. However, firewalls and gateways set at some end hosts may filter out an ICMP echo message, so this makes the simple `ping` request unreliable. Alternatively, we can use the proposed delay probing method, *DARB*.

A delay value table T_{delay} is built to store the network delay values according to Internet autonomous systems(ASes). After every time interval T, the T_{delay} extracts network delay values from the successive TCP three-way handshakes. It can also actively probe network delay when system is idle and there is a lack of successive handshakes for certain ASes. These delay values are collected in T_{delay} and refreshed at every time interval T. The delay value collection and congestion estimations are performed according to the ASes because the congestion status does not change much within the same AS. With the support from T_{delay}, *DARB* first searches the table for a specified IP before it probes the network. If a matched delay value is found, which is from the same AS as this IP, the probing process can be omitted to save probing time.

A table that has fixed capacity is used to implement T_{delay}. A table that has flexible capacity could become a potential target of a DDoS attack. When the table is full and a new record comes, the new record will replace the oldest one. Suppose that a table has 10k entries and each entry occupies 32 bits. The 24

bits are used to record the first three octets of an IP address, which contains enough information to identify an AS. The left 8 bits in an entry are used to store the probed delay value of a half-open connection. The total size of the table is about 40k bytes, which is affordable in current operating systems.

3.3. Classifying half-open connections using delay values

Half-open connections are classified as either normal or abnormal using the delay values obtained either from a history table or by the *DARB* method. As can be seen in Fig. 3, if the *DARB* method returns a result showing that a half-open connection has a large delay value, it is highly probable that the half-open connection is normal. A small delay value indicates that the network is in a benign traffic state, which does not produce half-open connections. Thus a small delay value of a path inside a half-open connection denotes that the tested half-open connection is abnormal and may be caused by a SYN flooding attack.

The first delay value that we must obtain is the average network traffic delay when a network runs smoothly. In a time period t, we select a sample from a server containing half-open connections. The sample is denoted as S. Let x_i be the delay returned from a history record or by the *DARB* method for the ith half-open connection in S. The average delay value $\overline{x}$ is calculated by

$$\overline{x} = \frac{\sum_{i \in S} x_i}{|S|}.$$

We now provide a function $f(x)$ to evaluate probed delays. Let $\beta = \overline{x}$ and x be a random variable to denote the delay value of a half-open connection either from a history table or probed by the *DARB* method. Thus, $f(x)$ indicates the probability of a half-open connection belonging to *abnormal half-open*.

$$f(x) = \begin{cases} \frac{1}{\beta} e^{-\frac{x}{\beta}} & x > 0, \\ 0 & x \leqslant 0. \end{cases}$$

The probability density function $f(x)$ is modeled by an exponential distribution because it reflects the relationship between the network traffic delay and the chance that a half-open connection is abnormal. When the observed delay of a half-open connection is short, which means x is nearly 0, it is highly probable that the connection is an abnormal half-open. However, if the probed delay shows network traffic congestion in a routing path, the returned value of x is much larger than β. The corresponding $f(x)$ is small and this would indicate a lower probability of a half-open connection generated by a SYN flooding attack.

3.4. Early detection scheme against DDoS attacks

We can launch an alert of DDoS attacks very early by examining half-open connections stored on a server. Note that it is not necessary to examine every half-open connection maintained on a protected server, only suspicious half-open connections, those which are maintained on the server longer than a predefined, adjustable time period t need to be tested. We obtain those connections by capturing a snapshot of the server's half-open connections at every interval t. We denote S_T as the snapshot set of half-open connections at time T and we denote S_{T-t} as the set at time $T-t$. It is possible to compare two adjacent snapshots, sets S_T and S_{T-t}. If a half-open appears in both S_{T-t} and S_T, we will carry out a delay probe, i.e., the elements in the intersection $S_{T-t} \cap S_T$ will be probed to get the delay value x. If there are too many half-open connections in intersection $S_{T-t} \cap S_T$, a sample will be selected from $S_{T-t} \cap S_T$ and this will be probed.

A threshold T is predefined and can be used to classify a half-open connection as normal or abnormal. Given a half-open connection, we can calculate its $f(x)$ that is defined in Section 3.3. The half-open connection that has $f(x) > T$ is suspicious and is classified as *abnormal half-open*. Otherwise, the half-open connection is legitimate and is regarded as a *normal half-open* connection. The value of T can be adjusted according to other network errors rather than DDoS attacks and network traffic congestions that generate half-open connections.

When the number of *abnormal half-opens* exceeds a predefined threshold N, a DDoS alarm will be sent out. At what level the threshold N is set depends on how many half-open connections a server can tolerate. If the server reserves more resources for half-open connections, the threshold N can be set accordingly larger.

4. Design of rate-limiting counteraction scheme

This section presents the design of the second component of our approach, a TTL-based rate-limit counteraction scheme. The purpose of this scheme is to minimize the damage caused by DDoS attacks. A good counteraction scheme must be *effective*, dropping or blocking most of the malicious packets; *accurate*, being able to distinguish between attack traffic and normal traffic, thereby guaranteeing the uninterrupted provision of services to legitimate users; and it must be *robust*; that is it must be able to execute its defence techniques at a low cost, which will allow it to efficiently handle high volumes of packet traffic.

These three design principles to be applied in our counteraction scheme will be discussed in detail below. Broadly, to efficiently block further attacking traffic, we restrain suspicious traffic by using a rate-limited strategy. To accurately distinguish malicious from legitimate traffic, our counteraction scheme constructs efficient rate-limited rules which use a TTL field in an IP header and the scheme uses two tables T_{TTL} and T_{TTL2IP} to accurately distinguish between malicious and legitimate users. We show the architecture of the counteraction system that protects the victim server with several functional components. Those components can provide rate limitation on suspicious traffic while requires a low cost of communication and table updates.

4.1. Rate-limiting using the TTL values

Because SYN flooding attack packets can use easily forged and randomly chosen IP and port information, attacking packets

cannot be effectively filtered using IP-based rate-limit rules. We propose instead to build the rate-limit rule using the IP packet header TTL value field, which indicates the remaining router hops before a packet gets deleted. As shown in the study of end-to-end routing stability [16], Internet paths are strongly dominated by a single route, and the routes of about two-thirds of them persist for days or weeks. This obviates the problem of forged IP addresses because attacking packets from the same attack source can be identified by the fact that they will have hopped across the same number of routers along their routing path. Attacking packets from one source maintain a static initial TTL value, with each intermediate router decreasing the TTL value by one before forwarding it to the next router. The result is that their TTL value will be the same upon arriving at a victim server. The central assumption of this is that attackers do not change the initial TTL value for each attacking packet. After we conduct an investigation into popular SYN flooding tools, including TFN2k and Stacheldraht, we find most current DDoS attack tools adhere to this assumption. The initial TTL value depends on different operation systems (OSs). Current OSs use only a few selected initial TTL values, i.e., 30, 32, 60, 64, 128 and 255 [8]. As Jin has noted, this set of initial TTL values covers most of the popular OSs, such as Microsoft Windows, Linux, variants of BSD, and many commercial Unix systems.

When abnormal half-open connections are detected, their TTL values are recorded in a table T_{TTL}. If the count of a special TTL value TTL_i exceeds a threshold M, this TTL value TTL_i is suspicious. The count of TTL_i is denoted as $Count(TTL_i)$. The packets having value TTL_i are suspected to come from the same attacking source. During SYN flooding attacks, numerous attacking packets are sent out from the same attacking source. Consequently, a lot of attacking packets resulting in abnormal half-open connections are found to have the same TTL value. The higher $Count(TTL_i)$ is, the more suspicious the packets with TTL_i are.

To limit attack packets rate, we propose here a soft rate-limiting scheme. Specifically, the percentage of traffic to go through is equal to

$$R_{\mathtt{pass}} = \gamma^{-\varepsilon Count(TTL_i)},$$

where $Count(TTL_i)$ is the count of a special TTL value ($TTL_i \in T_{TTL}$) and γ, ε are parameters. A legitimate user does not cause abnormal half-open connection. Its TTL value will not appear in T_{TTL} and $Count(TTL_i)$ is 0. Thus if $Count(TTL_i)$ is close to 0, we allow almost all traffic to pass through. A higher value of $Count(TTL_i)$ means that the traffic has a higher chance of being packets from the same attacking source. The larger the value of $Count(TTL_i)$ is, the more traffic will be blocked.

4.2. Minimizing collateral damage

Filtering all packets having a certain TTL value would result in the filtering of legitimate as well as attack packets. Hence, our TTL-based rate-limit scheme includes rules for distinguishing normal from spoofed packets. It does this by observing TCP three-way handshake behaviors (Fig. 1). During a normal three-way handshake procedure, $Syn(k)$, $Ack(k+1) + Syn(j)$ and $Ack(j+1)$ can be captured at the victim side. However, during a spoofed TCP connection, whereas the first and second round handshakes can be identified while the third round handshake, $Ack(j+1)$, cannot (Fig. 1(c)). On this basis, we conclude that a connection is legitimate if it is possible to capture its third round handshake $Ack(j+1)$. Traffic from this IP will not be confined within our rate-limit scheme.

The use of two tables, T_{TTL} and T_{TTL2IP}, is integral to our scheme. When the *DARB* method detects suspicious half-open connections and there is a warning of a DDoS attack, the TTL value of a suspicious packet is extracted and stored in the table T_{TTL} while on the victim side an *Ack*-Detector monitors the third round handshake $Ack(j+1)$. The source IP address of packet P is denoted IP_P and its TTL value is denoted T_P. When the $Ack(j+1)$ packet P, whose TTL value T_P is in T_{TTL} ($T_P \in T_{TTL}$), is captured, the record (IP_P, T_P) will be inserted into T_{TTL2IP}. As a result, packets from IP_P will not be rate-limited even if they have a suspicious TTL value IP_P ($T_P \in T_{TTL} \wedge Count(T_P) > M$). Table T_{TTL2IP} , used in conjunction with table T_{TTL}, rate-limits only packets in the set $\{P|(IP_P, T_P) \notin T_{TTL2IP} \wedge T_P \in T_{TTL} \wedge Count(T_P) > M\}$.

4.3. Counteraction scheme architecture

Fig. 5 depicts the architecture of the counteraction scheme. The half-open connection snapshot component makes a snapshot of half-open connections on the protected server at predefined intervals. After suspicious half-open connections are extracted from snapshots, they can be probed using *DARB* to verify whether there are signs of network congestion. If there are none, the half-open connection is treated as *abnormal half-open*. Its TTL value is extracted and is stored in table T_{TTL}.

The *Ack*-detector is in charge of capturing the third round handshake $Ack(j+1)$ packets from the incoming traffic. If the $Ack(j+1)$ packet P with a TTL value T_P and T_P belongs to T_{TTL}, the source host with the IP address IP_P has completed its three-way handshake with the server. The traffic is normal and should not be rate-limited. The record (IP_P, T_P) will be inserted into table T_{TTL2IP}. According to the rate-limit rules, the traffic from IP_P will not be rate-limited even if it has a suspicious TTL with ($T_P \in T_{TTL}$).

The counteraction scheme works smoothly from cooperative function units. The rate-limited controller, suspicious half-open detector, and *Ack*-detector work together to carry out the rate-limiting scheme. When the suspicious half-open detector finds a suspicious half-open, it extracts its TTL value and stores it in the table T_{TTL}. The *Ack*-Detector captures the third round handshake, $Ack(j+1)$, from the incoming traffic. If the TTL value T_P of the $Ack(j+1)$ packet P appears in the table T_{TTL}, the IP address IP_P of P will be inserted into table T_{TTL2IP}. If a packet appears in the set $\{P|(IP_P, T_P) \notin T_{TTL2IP} \wedge T_P \in T_{TTL} \wedge Count(T_P) > M\}$, the rate-limited controller will limit its traffic rate. These three components cooperate with each

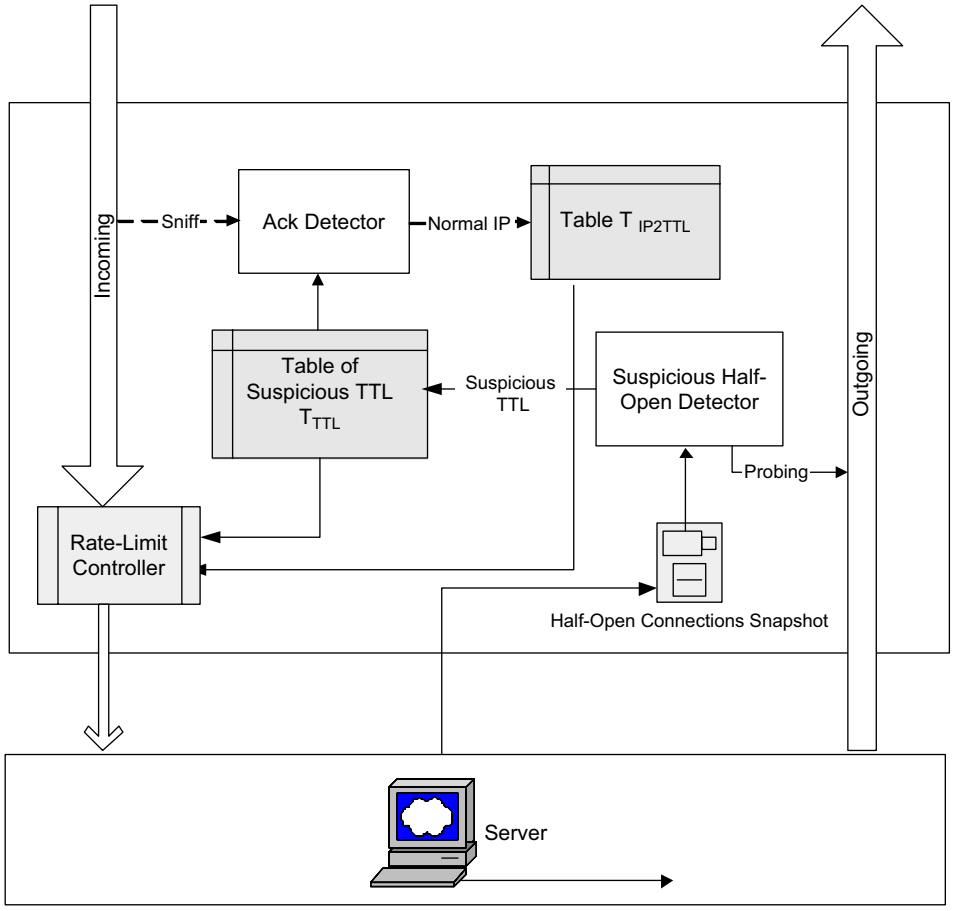


Fig. 5. The architecture of detection and response scheme.

other to counteract malicious traffic. This counteraction scheme has the further advantage of not depending on special hardware as it explores the TTL value and the source IP field inside a packet and, because the rate-limit rules are simple to enforce, the scheme can continue to efficiently handle large volumes of traffic.

5. Simulation results

In this section, we will show the simulation results to apply the proposed active detection scheme in Section 3 and counteraction scheme in Section 4. To evaluate our detection and counteraction performance, the network simulator NS2 [14] is used to simulate DDoS attack scenarios. The network traffic delay and congestion are tested in the NS2 software environment. The detection scheme is evaluated first to determine how efficiently, at an early stage, it detects a SYN flooding attack. If a DDoS attack alarm is launched, the counteract scheme is then applied. Using the suspicious TTL table T_{TTL} and legitimate IP table T_{TTL2IP}, only suspicious traffic will be rate-limited.

In our simulations, the network topology is designed as a tree topology. In the real DDoS attack scenario, a typical network topology is, from the perspective of victim, a tree structure. The tree depth is set from 2 to 16, which stands for different numbers of hops that packets must traverse along the routing path. Fig. 6

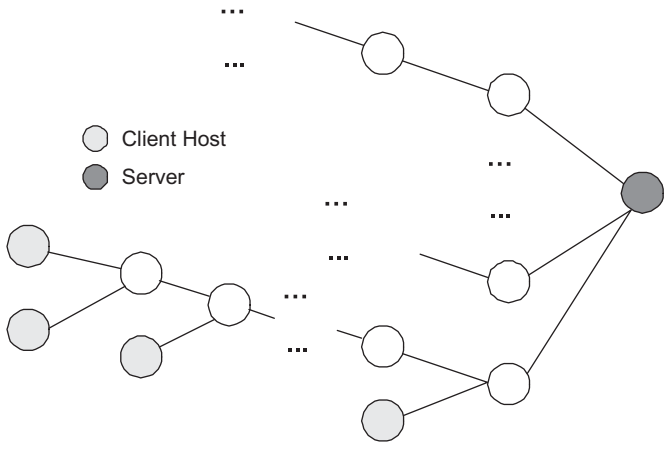


Fig. 6. The viewed network topology at the victim server side.

shows this network topology. The simulations employ a total of 115 nodes. The normal link bandwidth is set at 10 Mb and the delay of link is 10 ms. Around 10 links are randomly selected as the congested links and their traffic delay is set at a random value between 800 and 1500 ms. Once a path includes any congested links, the path delay value is assumed to be large. The more congested links that a path includes, the longer delay it will be. The network simulator lasts 100 s and the SYN flooding

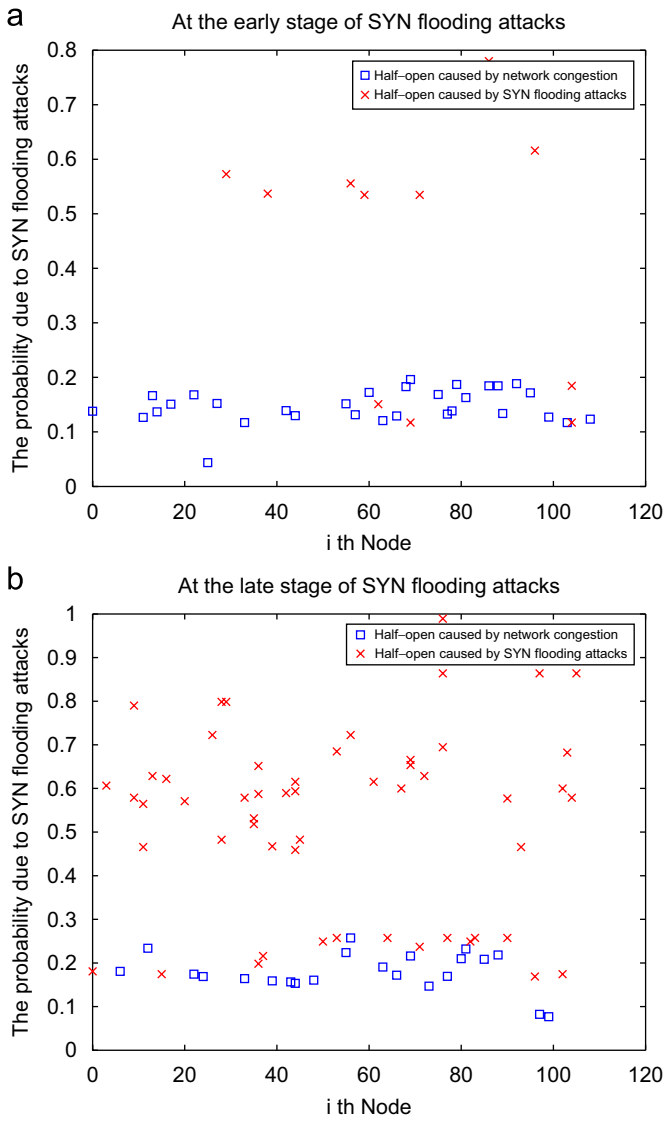


Fig. 7. The probability of a half-open connection caused by SYN flooding attacks. (a) DDoS attacks at the early stage. (b) More attack traffic observed.

attack begins at the 30th second. When SYN flooding attack begins, 40 nodes are chosen as the SYN flooding attack sources, sending numerous SYN control packets to a protected server.

5.1. Detection performance

We verify the efficiency of proposed active detection scheme in Section 3. We try to show it can detect SYN flooding attacks accurately and immediately after some suspicious packets sniffed. When some half-open connections are captured in the protected server, we simply use the *DARB* technique to probe the delay. The return probing delay is used to calculate the probability, which indicates whether a half-open connection is the result of a SYN flooding attack. Fig. 7 shows that abnormal half-opens have much higher scores than normal half-opens. This makes it easy to classify the half-open either into the normal group or into the abnormal group. Yet our method might accidently treat a packet with a spoofed address as a normal request if the path of $Ack(k+1)+Syn(j)$ packet comprises a congested router, which is shown by a few '×'s staying lowly in Fig. 7(a), in most cases we can correctly identify those half-open connections caused by SYN flooding attacks because the majority of them presents a probability over a half. When more attacking traffic observed at the server side, a large number of connections that are detected with a noticeable probability caused by a SYN flooding become obvious as shown in Fig. 7(b). Note that our method works efficiently no matter how much abnormal half-open connections are maintained by the victim. Therefore the proposed method can detect abnormal half-open connections even at the early stage of attack when there is not much of abnormal half-open connections. Adjusting the value of N (defined in Section 3.4), we can accurately and immediately announce a DDoS attack after the probability analysis of some half-open connections kept on a protected server.

5.2. Counteraction performance

We conducted simulations to test the performance of counteraction scheme presented in Section 4. In order to get a comprehensive performance evaluation, we designed two DDoS attacking scenarios: scenario *A* and scenario *B*. After the demonstration of the simulation results in these two scenarios, we compared the impacts on legitimate traffic when a server is either equipped with a defence scheme or not.

Two attacking scenarios (i.e., scenario *A* and scenario *B*) were designed to apply the counteraction scheme. In the scenario *A*, the server had sufficient resource to provide service to legitimate user when there is no DDoS attack. This scenario was designed to simulate large web sites which have abundant resources. The bandwidth of server was set to 10 Mbit/s to handle all traffic requests. In the scenario *B*, the server was busy to provide service to legitimate users even when there were no DDoS attacks. This scenario simulated small web sites which do not have much redundant resources. The bandwidth of server was set to 2 Mbit/s. Assume that in both scenarios, the bandwidth requested by legitimate users is 2 Mbit/s.

Fig. 8 shows simulation results in scenario *A* when the normal traffic started at 0 s, in which Fig. 8(a) demonstrates the performance of a system without any defence system while Fig. 8(b) demonstrates the performance of a system equipped with our detection scheme. When the studied server system had sufficient resources without defence systems against DDoS attacks, it suffered from attack but still can provide low quality service to legitimate users as shown in Fig. 8(a). The server had sufficient bandwidth to provide service to legitimate users at the first 30 s. After 30 s, the attack traffic began to flow toward victim and the bandwidth of the victim was exhausted by the attacking traffic. The victim was busy to handle numerous attacking traffics and could not provide stable high-quality service to legitimate users. Because the attacking traffic occupied most resources, the bandwidth to handle legitimate users' request decreased from 2.8 to 1 Mbit/s as shown in Fig. 8(a). After 80 s the attack began to stop. Both the total bandwidth

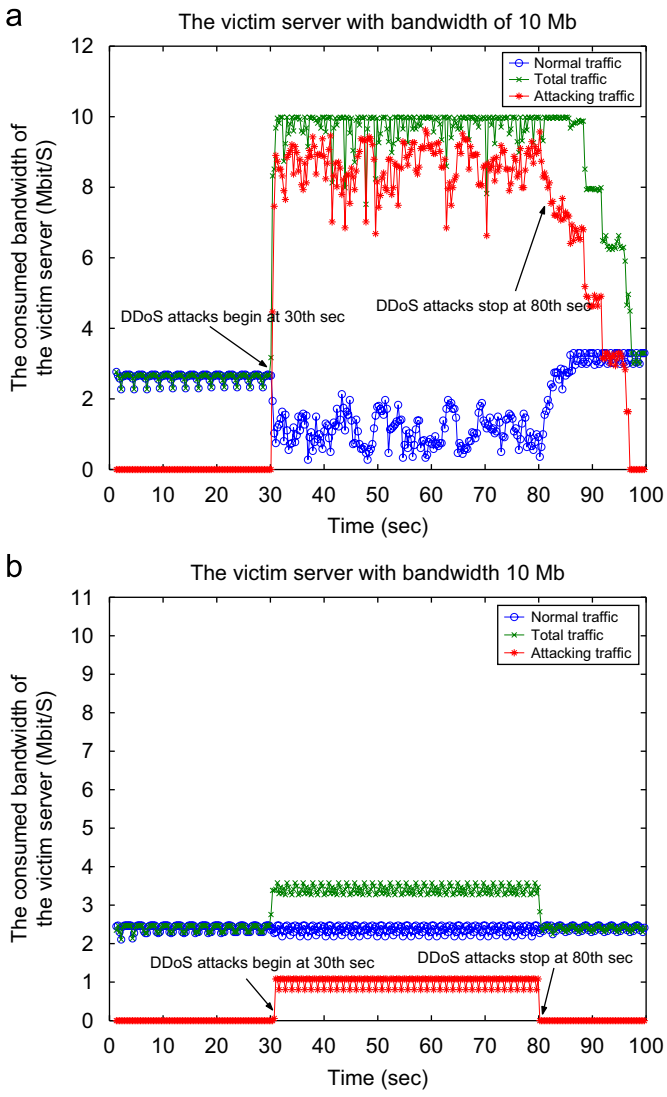


Fig. 8. Performance results, respectively, for a defenceless system and a protected system. (a) The victim without defense suffers from attacks. (b) Counteract scheme mitigates the damage caused by attacks.

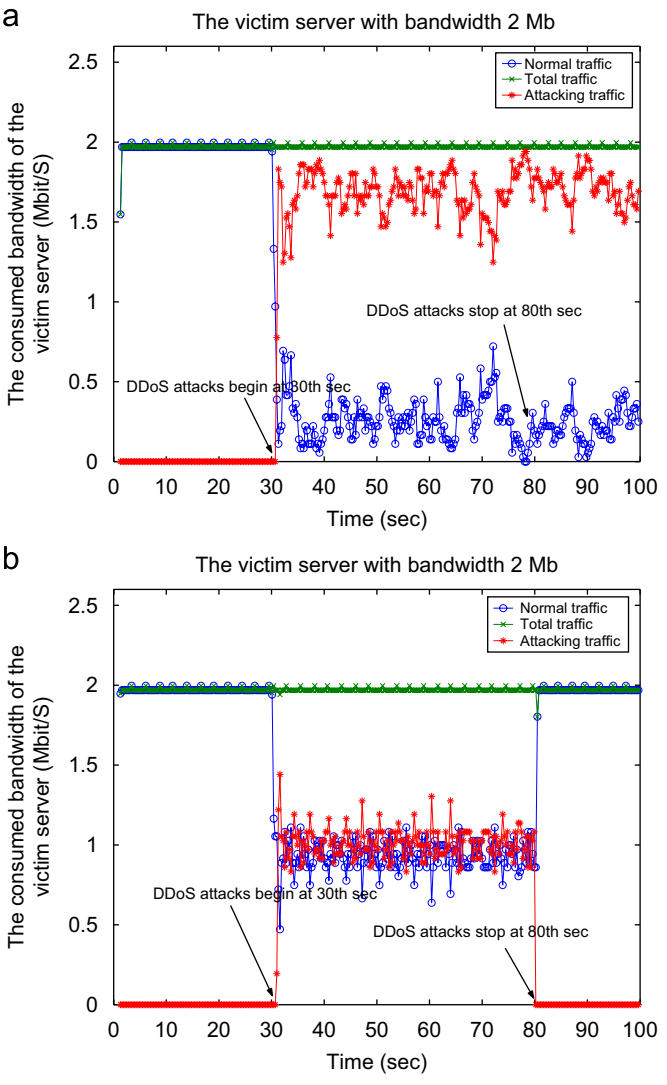


Fig. 9. Performance results, respectively, for a defenceless system and a protected system. (a) The victim without defense suffers from attacks. (b) Counteract scheme mitigates the damage caused by attacks.

and the legitimate users' bandwidth recovered to a normal level slowly, which means the victim without any defence cannot recover to a healthy state unless the attack stops.

Our counteract scheme allowed the normal traffic did not suffer from attack as shown in Fig. 8(b). When our defence method was applied, we noticed that the victim server minimized the damage caused by attack. The occupied bandwidth from attack traffic was around 1 Mbit/s that was much less than almost 9 Mbit/s in the situation as shown in Fig. 8(a). Meanwhile, the TTL-based counteract scheme to limit the rate of attacking traffic bandwidth maintained the legitimate users' bandwidth during the attacking period from 30 to 80 s. Though the total used bandwidth in a server was increased a little, the system allowed such increment without any damage on responding the traffic requests from legitimate users.

Fig. 9 shows simulation results in scenario *B* when the server only had 2 Mbit/s- bandwidth to meet the requirements from legitimate users, in which Figs. 9(a) and (b), respectively, demonstrate the performance of a defenceless system and a system equipped with our counteract scheme. Fig. 9(a) shows the reaction of a defenceless system under DDoS attacks. When the attack began at 30 s, the legitimate users' bandwidth greatly decreased from 2 to 0.3 Mbit/s. The victim server could hardly provide any continuous service to legitimate users. Even after 80 s when the attack began to cease, the legitimate users' bandwidth did not recover back from attack immediately. It means the DDoS attack can cause more fatal damages to a server that has small capacities to tolerate evil attacks. Fig. 9(b) shows the results that a victim server adopted the TTL-based counteract scheme to limit the rate of suspicious traffic. Although the bandwidth required by legitimate users could not be fully attained because a portion of 2 Mbit/s bandwidth was used to defend against the attack traffic, the service to legitimate users was much better than that in a defenceless system. The available bandwidth to provide service to normal requests

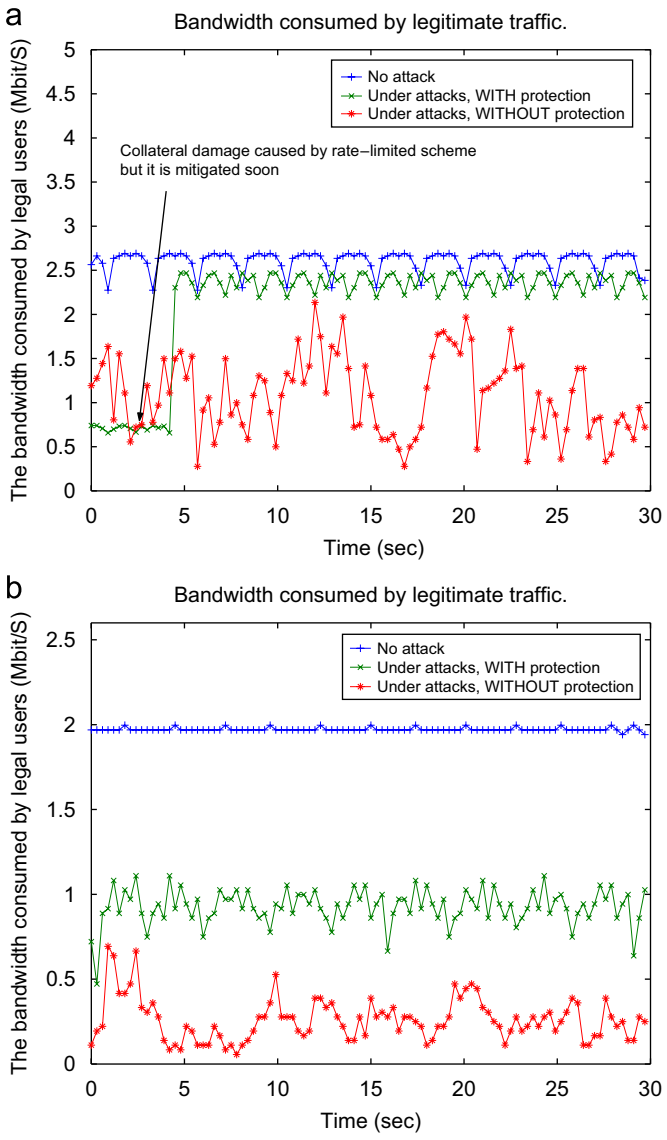


Fig. 10. Legitimate user's traffic suffered seriously if our defence scheme was not adopted. (a) The server bandwidth is 10 Mbit/s. (b) The server bandwidth is 2 Mbit/s.

was around 1 Mbit/s. This simulation result indicates that our counteract scheme is effective to minimize the damage from DDoS attacks by providing 3.3 (1/0.3) times more service to legitimate users.

We compare the available bandwidths to support the traffic requests from legitimate users in three situations: legitimate traffic free of attack, legitimate traffic under attack but without protection, and legitimate traffic under attack but with protection from our counteract scheme. Fig. 10 shows the comparison results where Fig. 10(a) depicts the results from scenario *A* and Fig. 10(b) depicts the results from Scenario *B*. The upper curve represents the bandwidth occupied by legitimate users when there are no attacks. In most of time, the curve representing the results from our counteraction scheme is highly above the curve representing the results from a system without any defences. The figure shows that our defence scheme can efficiently throttle the DDoS attack and assure the server to provide satisfying service during DDoS attacks.

5.3. Comparison with FDS

We compare our active detecting method with FDS (flooding detection system) [26] using NS2. FDS is an effective defending method against SYN flooding attacks. Currently many SYN flooding detection methods have been proposed but most of them use passive detection mechanisms, which rely on passively sniffing an attacking signature. Since there is a lack of active methods to make comparisons, our active method is compared with FDS. FDS is based on the protocol behavior of TCP SYN-FIN(RST) pairs, and is an instance of the sequential change point detection. FDS uses a non-parametric cumulative sum(CUSUM) method to make detection more accurate and insensitive. Compared with other passive detecting methods, FDS has the advantages of accurate detection rate and low computation overhead. However, it does not provide any response scheme. Therefore, we only compare it with our detection method in terms of the detection efficiency. We first briefly introduce the FDS change-point detection method in [26]. Then we compare the detection efficiency of FDS with our active detection method using NS2.

5.3.1. FDS (flooding detection system)

FDS detection is based on the observation under a normal condition: 1. the SYNs and RSTs have a strong positive correlations; 2. the difference between the number of SYNs and that of FINs is close to the number of RSTs. In the normal operation, a FIN(RST) is paired with a SYN at the end of data transmission; but under SYN flooding attacks, this SYN-FIN(RST) pair's behavior will be violated, deviating from the normal operation. In the FDS method, let $\{\Delta_n, n = 0, 1, \ldots\}$ be the number of SYN $-$ (FIN $+$ RST) pairs within one sampling period t_0. $\{\Delta_n\}$ is normalized by the average number $\bar{F}$ of FINs(RSTs) during the sampling period. $\bar{F}$ can be estimated in the real time and updated periodically. An example of recursive estimation and update of $\bar{F}$ is

$$\bar{F}(n) = \alpha\bar{F}(n-1) + (1-\alpha)FIN(RST)(n),$$

where n is the discrete time index and α is a constant lying strictly between 0 and 1 that represents the memory in the estimation.

Define $X_n = \Delta_n/\bar{F}$. The mean of X_n, denoted as c, is much less than 1 and close to 0. FDS uses CUSUM to analyze the sequential change of X_n and defines $y_n = (y_{n-1} + X_n)^+$, $y_0 = 0$. y_n is the CUSUM result of X_n. When the SYN flooding attack occurs, the number of SYNs is much greater than that of FINs(RSTs). X_n and y_n grow abnormally large. When y_n exceeds a predefined threshold, an alarm will be triggered.

FDS is deployed at leaf routers that connect end hosts to the Internet. It can be deployed at the first-mile or last-mile leaf routers. When the FDS is deployed at the last-mile router, it performs detection near the victim side, which is similar to the deployment of our method.

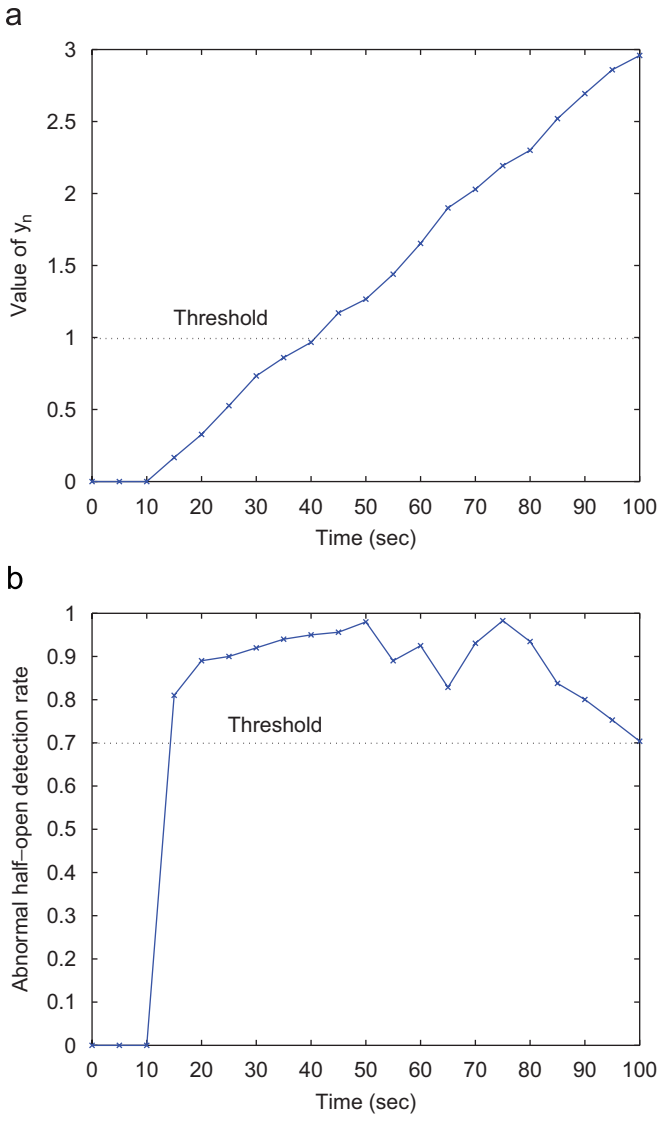


Fig. 11. An alarm is generated when a curve goes beyond the threshold. (a) Detection results of FDS. (b) Detection results of our active method.

5.3.2. Simulation comparisons

Fig. 6 shows the simulated network topology. The whole simulation lasts for 100 s. There are 40 attacking nodes and 75 normal nodes. The bandwidth of a normal link is set at 10 Mb and the delay is 10 ms. Around 10 links are randomly selected as the congested links and their traffic delay is set at a random value between 800 and 1500 ms. SYN flooding attacks begin at the 10th second and end at the 100th second. FDS and our active detection method refresh detection results every 5 s.

Fig. 11(a) shows detection results of FDS. We can see that FDS can gradually identify abnormal SYN packets and accurately detect SYN flooding attacks as time passed by. According to the suggestion in [26], the threshold is set to 1. After the SYN flooding attack begins from the 10th second, FDS launches the alarm at the 40th second. The whole detection process requires 30 s.

Fig. 11(b) shows detection results of our active detection method. The ratio of detected abnormal half-open to total number of half-open connections grows rapidly and exceeds the threshold right after the SYN flooding attack is launched. When the SYN flooding attack begins at the 10th second, our method can detect numerous abnormal half-open connections at the 15th second. The prompt attack detection is the result of the equipped active probing mechanism, *DARB*, which does not passively wait for significant number of incoming attacking packets. When more suspicious half-open connections appear at the late simulation stage, the detection rate fluctuates because *DARB* only probes a sample from all half-open connections to reduce overhead. Since *DARB* can give an alarm at very early stage of attacks, this fluctuation does not affect the detection efficiency.

From the comparisons, we show that *DARB* can detect DDoS attacking signatures more quickly and accurately. Our method does not need to passively wait for attacking signatures and is able to trigger alarms in a timely manner. FDS requires longer detection time because FDS has to wait for collecting enough SYN-FIN(RST)-pair information. A SYN packet is received at the setup stage of a TCP connection while a FIN packet is received when the connection is built up. Since the TCP connection buildup time could last from several seconds to tens of minutes, it will greatly affect the detection efficiency of FDS.

6. Discussion

Our defence approach against DDoS attacks comprises two cooperating components: the detection scheme and the counteraction scheme. The detection scheme seeks an early warning of DDoS attacks through the active probing method and the counteraction scheme is capable to minimize the damage caused by attack traffic. Although there are several factors that may impact the accuracy of detection, the active detection scheme is able to launch alerts correctly in time. The detection scheme is suitable to be applied in a large network area and it is immune to be attacking target.

Although several factors could possibly confuse our decisions on half-open connections, the use of parameter N (introduced in Section 3.4) assures the correct detection of DDoS attacks. Our detection scheme allows the existing of errors on the decision of a small part half-open connections. It is possible that the path to a spoofed IP address of an attacking packet incidentally includes a congested router. The half-connection to this spoofed IP address could be wrongly classified and a false negative is introduced. The errors in routers could also cause potential false positives. However, such error decisions on half-open connections are very few. As shown in Fig. 7, we observe the majority of half-open connections are classified without mistakes and therefore the detection scheme can issue alarms on time. In addition, such errors can be amended if we incorporate the state information of link congestions and routers to the history table.

Our active detection scheme is scalable and robust in the sense that it obtains a path delay either from a history table or

using the *DARB* probing technique. The early detection scheme is scalable to be applied in a large network area. Generally speaking, in most cases it will not be necessary to apply *DARB* technique because it will be possible to obtain the delay value of a path from a history table. This is especially the case if a server collects network traffic information periodically (to assist the network traffic congestion management) and we assume that over a short period the network traffic delay does not change dramatically. Ultimately, only a small proportion of destinations will have to be probed using *DARB*.

The active detection scheme is also robust enough to handle some overwhelming traffic attacks to deplete the limited capacities on a server reserved to contain half-open connections. If most half-open connections from a burst of traffic must be probed using the *DARB* method, our active detection scheme is able to yield a warning of DDoS attacks before the victim becomes congested. As denoted in [13], a victim server can be overwhelmed by an attack rate of 500 SYN packets per second. The WIN server has a more tolerance of DDoS attacks by dropping off a half-open connection at a timeout of 9 s [3]. As shown in Fig. 2, the *DARB* method is effective enough to acknowledge a delay value within 200 ms in a network without traffic congestion, or within 2 s for most cases of a congested network. That is a response time far before 9 s. Hence, the proposed active detection scheme is able to generate an alarm of DDoS attacks accurately and quickly. After the number of abnormal half-open connections reach a threshold N, we can launch the warning of DDoS attacks and stop the detection scheme to initiate the counteraction scheme. Thus, we assure the detection scheme itself is immune to the target of DDoS attacks.

7. Conclusion

This paper presents a system for defending against SYN flooding attacks. The system consists of a detection and counteraction component and uses congestion measures to classify half-open connections as either abnormal or normal, thereby determining whether a server is the victim of a DDoS attack. The detection component uses returned delay values from a history record or a network traffic delay-probing technique (*DARB*). This kind of active detection results in earlier and more accurate detection and can be easily deployed. Counteraction does not rely on source IP addresses as these are easily spoofed. Rather, the TTL field is extracted from suspicious packets and packets are filtered according to rate-limit plan that minimizes collateral damage by using a record of recent legitimate traffic. Our rate-limit counteraction scheme only uses two tables T_{TTL2IP} and T_{TTL} to minimize the damage caused by a DDoS attack. Both the storage size and execution cost are small. The experimental results show that *DARB* can accurately distinguish normal and abnormal half-open connections and that the TTL-based counteraction scheme does reduce the damage caused by DDoS attacks.

References

[1] S.M. Bellovin, ICMP traceback messages, Technical Report March 2000.

[2] S. Branigan, H. Burch, B. Cheswick, F. Wojcik, What can you do with Traceroute?, IEEE Internet Comput. 5 (5) (2001) 96.

[3] R.K. Chang, Defending against flooding-based Distributed Denial-of-Service attacks: a tutorial, IEEE Commun. Mag. 40 (10) (2002) 42–51.

[4] C.-M. Cheng, H. Kung, K.-S. Tan, Use of spectral analysis in defense against DoS attacks, in: Proceedings of IEEE GLOBECOM 2002, Taibei, China, vol. 3, 2002, pp. 2143– 2148.

[5] D. Dean, M. Franklin, A. Stubblefield, An algebraic approach to IP traceback, ACM Trans. Inform. System Security 5 (2) (2002) 119–137.

[6] P. Ferguson, D. Senie, Network ingress filtering: defeating denial of service attacks which employ IP source address spoofing, RFC2827, 2000 URL: ⟨http://www.ietf.org/rfc/rfc2827.txt⟩.

[7] J. Ioannidis, S.M. Bellovin, Implementing pushback: router-based defense against DDoS attacks, in: Proceedings of Network and Distributed System Security Symposium, San Diego, California, The Internet Society, 2002.

[8] C. Jin, H.N. Wang, K.G. Shin, Hop-count filtering: an effective defense against spoofed DDoS traffic, in: Proceedings of the 10th ACM Conference on Computer and Communication Security (CCS), ACM Press, New York, 2003, pp. 30–41.

[9] M. Kalantari, K. Gallicchio, M. Shayman, Using transient behavior of tcp in mitigation of distributed denial of service attacks, in: Proceedings of the 41st IEEE Conference on Decision and Control 2002, vol. 2, 2002, pp. 1422–1427.

[10] A. Keromytis, V. Misra, D. Rubenstein, SOS: an architecture for mitigating DDoS attacks, IEEE J. Sel. Areas Commun. 22 (2004) 176–188.

[11] J. Lemon, Resisting SYN flood DoS attacks with a SYN cache, in: Proceedings of the BSDCon Conference, 2002, pp. 89–97.

[12] J. Mirkovic, G. Prier, P. Reiher, Attacking DDoS at the source, in: Proceedings of the 10th IEEE International Conference on Network Protocols (ICNP2002), Paris, France, IEEE Computer Society, 2002, pp. 312–321.

[13] D. Moore, G. Voelker, S. Savage, Inferring Internet denial-of-service activity, in: Proceedings of the 10th USENIX Security Symposium, 2001, pp. 9–22.

[14] Network simulator NS2 ⟨http://www.isi.edu/nsnam/ns/⟩.

[15] K. Park, H. Lee, On the effectiveness of probabilistic packet marking for IP traceback under denial of service attack, in: Proceedings of the IEEE INFOCOM, 2001, pp. 338–347.

[16] V. Paxson, End-to-end routing behavior in the Internet, IEEE/ACM Trans. Networking 5 (5) (1997) 601–615.

[17] T. Peng, C. Leckie, R. Kotagiri, Protection from distributed denial of service attack using history-based IP filtering, in: Proceedings of the IEEE International Conference on Communications, Anchorage, Alaska, USA, vol. 1, 2003, pp. 482–486.

[18] J. Postel, Transmission control protocol: DARPA internet program protocol specification, RFC 793, September 1981.

[19] S. Savage, D. Wetherall, A. Karlin, T. Anderson, Practical network support for IP traceback, in: Proceedings of the ACM SIGCOMM Conference, ACM Press, New York, 2000, pp. 295–306.

[20] C.L. Schuba, I.V. Krsul, M.G. Kuhn, E.H. Spafford, A. Sundaram, D. Zamboni, Analysis of a denial of service attack on TCP, in: Proceedings of the IEEE Symposium on Security and Privacy, IEEE Computer Society, IEEE Computer Society Press, Silver Spring, MD, 1997, pp. 208–223.

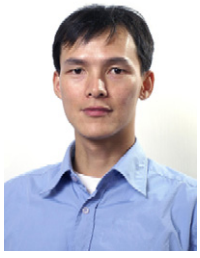
[21] A.C. Snoeren, Hash-based IP traceback, in: Proceedings of the ACM SIGCOMM Conference, ACM Press, New York, 2001, pp. 3–14.

[22] D.X. Song, A. Perrig, Advanced and authenticated marking schemes for IP traceback, in: Proceedings of the IEEE INFOCOM, 2001, pp. 878–886.

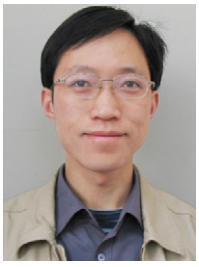
[23] M. Sung, J. Xu, IP traceback-based intelligent packet filtering: a novel technique for defending against internet DDoS attacks, IEEE Trans. Parallel Distrib. Systems 14 (9) (2003) 861–872.

[24] U. Tupakula, V. Varadharajan, Counteracting DDoS attacks in multiple ISP domains using routing arbiter architecture, in: Proceedings of the 11th IEEE International Conference on Networks(ICON2003), 2003, pp. 455–460.

[25] H. Wang, K. Shin, Transport-aware IP routers: a built-in protection mechanism to counter DDoS attacks, IEEE Trans. Parallel Distrib. Systems 14 (2003) 873–884.
[26] H. Wang, D. Zhang, K.G. Shin, Detecting SYN flooding attacks, in: Proceedings of IEEE INFOCOM, vol. 3, 2002, pp. 1530–1539.
[27] A. Yaar, A. Perrig, D. Song, SIFF: A stateless internet flow filter to mitigate DDoS flooding attacks, in: Proceedings of the IEEE Symposium on Security and Privacy, IEEE Computer Society, IEEE Computer Society Press, Silver Spring, MD, 2004, pp. 130–143.



Bin Xiao received the B.Sc and M.Sc degrees in Electronics Engineering from Fudan University, China. in 1997 and 2000, respectively, and Ph.D. degree from University of Texas at Dallas, USA, in 2003 in Computer Science. Now he is an Assistant Professor in the Department of Computing of Hong Kong Polytechnic University, Hong Kong. His research interests include communication and security in computer networks, peer-to-peer networks, wireless mobile ad hoc and sensor networks.



Wei Chen received the B.Sc. and Ph.D. degrees in Computer Science from Wuhan University, China, in 2000 and 2005, respectively. During 2004.6–2004.12 and 2006.4–2006.7, he worked as a research assistant in Hong Kong Polytechnic University. Now he is a lecturer in Nanjing University of Posts and Telecommunication, China. His main research interests include network security and network performance analysis.

Professor **HE Yanxiang** was born in 1952. He received his M.S. and Ph.D. degrees in computer science from the Oregon University, U.S. and Wuhan University, respectively. He is the dean of Computer School at Wuhan University and Vice Director of State key Lab of Software Engineering. He has published over 150 papers in related journals and conference proceedings and 15 academic works and teaching books. His research interests are distributed parallel processing, agent technology, grid computing, data mining and software engineering.