Sustaining Availability of Web Services under Distributed Denial of Service Attacks*

Jun Xu (contact author) and Wooyong Lee
College of Computing
Georgia Institute of Technology
Atlanta, GA 30332-0280
{jx,wooylee}@cc.gatech.edu

November 7, 2002

Abstract: The recent tide of Distributed Denial of Service (DDoS) attacks against high-profile web sites demonstrate how devastating DDoS attacks are, and how defenseless the Internet is under such attacks. We design a practical DDoS defense system that can protect the availability of web services during severe DDoS attacks. The basic idea behind our system is to isolate and protect legitimate traffic from huge volume of DDoS traffic when an attack occurs. Traffic that needs to be protected can be recognized and protected using efficient cryptographic techniques. Therefore, by provisioning adequate resource (e.g., bandwidth) to legitimate traffic separated by this process, we are able to provide adequate service to a large percentage of clients during DDoS attacks. The worst-case performance of the system is evaluated based on a novel game theoretical framework, which characterizes the natural adversarial relationship between a DDoS adversary and the proposed system. We also conduct a simulation study to verify a key assumption used in the game-theoretical analysis and to study the system dynamics during an attack.

Keywords: Availability, Survivability, Game Theory, Distributed Denial of Service (DDoS), World-Wide Web.

1 Introduction

The recent tide of Distributed Denial of Service (DDoS) attacks against high-profile web sites such as Yahoo, CNN, Amazon and E*Trade in early 2000 [13] demonstrate how damaging the DDoS attacks are, and how defenseless the Internet is under such attacks. The services

*This work is supported in part by the National Science Foundation under grant ITR/SY 0113933.

of these web sites were unavailable for hours or even days as a result of the attacks.

In a DDoS attack, a human *adversary* first compromises a large number of Internet-connected hosts by exploiting network software vulnerabilities such as buffer overrun. Then DDoS software such as TFN (Tribe Flood Network) will be installed on them. These hosts will later be commanded by the adversary to simultaneously send a large volume of traffic to a victim host or network. The victim is overwhelmed by so much traffic that it can provide little or no service to its legitimate clients. We refer to such compromised hosts as *attackers* in the sequel.

Most of the DDoS research [4, 5, 10, 35, 39, 11, 38] currently being proposed deals with *IP traceback*, that is, to trace the origins of the attackers[1]. Once the true identity of an attacker is established through traceback, it will be "taken out" through administrative means (e.g., to be shut down manually by a network manager). This is in general a slow process which may take hours or even days. During this period of time, the web site can do little to restore its service to legitimate clients. Therefore, although IP traceback is useful in identifying attackers post-mortem, they are not able to mitigate the effect of an attack while it is raging on.

1.1 Overview of the proposed work

The objective of this work is to design an effective and practical countermeasure that allows a victim system or network to sustain high availability during such attacks. In particular, we propose a DDoS defense system for sustaining the availability of web services. Protecting web services is of paramount importance, because the web is the core technology underlying E-commerce and the primary target for DDoS attacks.

When a DDoS attack occurs, the proposed defense system ensures that in a web transac-

[1] This is not trivial since the source IP address contained in DDoS packets can be spoofed.

tion, which typically consists of hundreds or even thousands of packets from client to server (shown later in Table 1), only the very first SYN packet may get delayed due to packet losses and retransmissions. Once this packet gets through, all later packets will receive service that is close to normal level. This clearly will lead to significant performance improvement.

The basic idea behind the proposed system is to isolate and protect legitimate traffic from huge volumes of DDoS traffic when an attack occurs. *Our first step is to distinguish packets that contain genuine source IP addresses from those that contain spoofed addresses.* This is achieved by redirecting a client to a new IP address and port number (to receive web service) through a standard HTTP redirect message. Part of the new IP address and port number will serve as a Message Authentication Code (MAC) for the client's source IP address. Packets from an attacker who uses spoofed IP addresses will not have the correct MAC, since the attacker will not be able to receive the HTTP redirect message.

However, attackers may also use their genuine IP addresses to send a large volume of traffic to the victim. *Our second step is to prevent such attackers from consuming too much system resource.* The strategy is to perform fair bandwidth allocation among all clients and attackers that are using legitimate IP addresses. However, even with the fair bandwidth allocation, the attackers may still outnumber the legitimate clients and "steal" a large portion of the system bandwidth. To address this problem, we introduce a "no loitering" law to enforce a quota on the amount of high-priority traffic each client may send. When a client has exceeded this quota, it is suspected as a possible attacker, and will be given only a fraction of its fair share[2]. In this way, we guarantee that eventually most of the system resource will be given to legitimate clients.

The proposed system is designed for practical implementation. It does not require the modification of either the web server or web client software. The proposed system only re-

[2] It is, however, allowed to use the excess bandwidth if there is any.

quires some lightweight (e.g, no per-flow state) support from a small number of intermediate ISP routers. In contrast, IP traceback schemes would require most of the Internet routers to participate.

1.2 Performance modeling using a game-theoretical framework

Another important part of this work is to employ a novel game-theoretical framework to model the effectiveness of the proposed system, and to guide its design and performance tuning accordingly. In DDoS attacks, performance of a system becomes a security issue because it is exactly what the adversary aims to destroy. The effectiveness (performance) of the proposed system is modeled under the following conservative assumption. We assume that the adversary tries to minimize the overall *utility* (e.g., total client satisfaction) of the proposed system by choosing the most effective strategy at its disposal. The proposed system, on the other hand, tries to choose a strategy that maximizes this utility. The adversarial relationship between the adversary and the system suggests that the system performance should be analyzed using a constrained minimax model in the context of game theory. The minimax utility under this model represents the worst-case performance of the system under *all possible attacks* within the attackers' capability. Our goal in designing the defense system is to achieve reasonable level of *minimax utility*. We refer to this goal as our *minimax sound* design principle.

The minimax soundness principle is based on a conservative assumption that an adversary will use all strategies at his/her disposal to reduce the system performance. We believe this is a valid assumption even though most real-world attacks use strategies much less sophisticated than this worst-case. For example, we show in [43] that a defense technique proposed by Internet Security System (ISS) [3] is very effective in countering current DDoS software. However, it becomes powerless when such software is slightly modified [43]. This shows that

Site visited	Action	#Packets the client sends
cnn.com	browse three pieces of headline news	1387
delta.com	search, reserve, and purchase a flight ticket	513
etrade.com	look up 5 stock quotes and access account balance	523

Table 1: Number of packets (HTTP and HTTPS) sent by a client during typical web transactions

a defense technique which is not minimax sound can at best be a short-term solution.

We apply the minimax soundness principle to the design of the proposed DDoS defense system. In Sec. 2.3, we analyzed various ways in which the proposed system can be attacked, through which we identify the system's and the adversary's best strategies. Performance results based on the game-theoretical analysis (Sec. 3) indicate that the proposed system is very effective in protecting web services. For example, during an attack where the incoming traffic rate is 5 times as high as the total link rate, a system with medium load (50%) can continue to provide service to about 55% of legitimate clients. Without such protection, no client is able to receive any service.

1.3 Organization of the rest of the paper

The rest of the paper is organized as follows. Section 2 presents the design of the proposed system. Section 3 analyzes its performance using a game-theoretical framework. Simulation results are shown in Section 4. Section 5 discusses the implementation issues of the system. Section 6 surveys the related work on DoS and DDoS. Section 7 summarizes the contributions of this work.

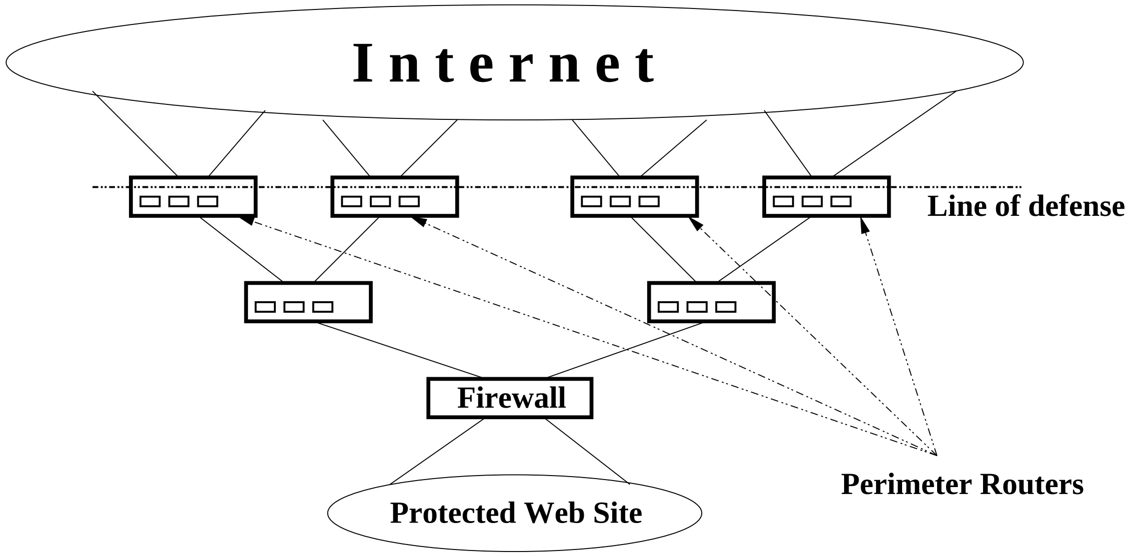

Figure 1: The system model for the proposed defense system

2 Design of the Proposed System

We propose a practical DDoS defense system that aim to sustain the availability of web services under DDoS attacks. We observe that a web transaction typically consists of hundreds or even thousands of packets sent from a client to a server. This is confirmed by our measurement results shown in Table 1. During a DDoS attack, since the packets will be randomly dropped at high probability, each of these packets will go through a long delay due to TCP timeouts and retransmissions. Consequently, the total page download time in a transaction can take hours[3]. Such service quality is of little or no use to clients. In contrast, our defense system ensures that throughout a web transaction, only the very first packet from a client may be delayed. All later packets will be protected and served. We show that this allows a decent percentage of legitimate clients to receive reasonable level of service.

2.1 System model

The proposed protection system adopts a similar system model as used in [25], shown in Fig. 1. The protected web site is connected to the Internet through a firewall. A set of upstream routers, typically belonging to a local ISP, will help protect the web site by dropping DDoS packets going through them. We refer to them as *perimeter routers* in the sequel. Instructions for carrying out this filtering operation will be issued to the perimeter routers by the firewall. We will show in Sec. 5.1 that the filtering operation is lightweight in terms of both space and time complexity: the amount of state it needs to keep is small (e.g., less than 100K bytes), and the amount of computation involved is reasonable (e.g., 1.1 μs per packet).

It is necessary to obtain help from the perimeter routers because, during a DDoS attack, most of the packets are dropped at the upstream routers before reaching the victim [25]. However, the proposed defense system is different from [25] in that it allows the perimeter routers to distinguish between DDoS and legitimate traffic, thereby making a better-informed filtering decision than [25] (discussed in detail in Section 6).

2.2 System assumptions

In the following, we state and justify the assumptions used in designing the proposed system and modeling its performance:

- We assume that the firewall, rather than the web server farm, is the performance bottleneck of the whole system. This is usually true in high-volume web sites where hundreds or even thousands of servers handle client requests in parallel. How to design a web server that is robust against bandwidth DDoS attacks is an interesting research

[3]This estimate takes into consideration the fact that several concurrent TCP connections are allowed.

topic, but is outside the scope of this paper. As to TCP SYN flood attack [7] that targets TCP/IP socket data structure inside web server OS, the proposed system employs the standard technique of TCP connection interception to counter it (discussed in Section 5).

- We assume that there can be a large number of attackers. Scenarios with several thousand attackers will be used in our performance modeling study. Our analysis remains correct when the number of attackers is small.

- Each attacker may send any type of DDoS packets, using spoofed or genuine IP addresses. However, its attacking bandwidth is limited by its local link speed.

- We assume that DDoS attacks in general will not significantly impact *unidirectional* packet forwarding speed at intermediate routers from web servers to web clients, although performance in the other direction can be severely degraded. This is generally true in today's routers that use full-duplex links and switched architectures[4] [31].

- For performance modeling purpose, we assume that 8 seconds are as long as a human user's patience can last. In other words, if there is no response from a web site for 8 seconds, a client gives up. This assumption is backed up by a careful study performed by Zona Research Inc. [2].

- We assume that the perimeter routers will share a secret key (for performing MAC verification) with the perimeter routers. This requires a secure key distribution protocol. One of the existing protocols such as [27, 12, 33] may be used or adapted for this purpose.

[4] In older switch/router designs, where shared-bus architecture [8] were used instead, inbound and outbound traffic may affect each other.

2.3 Making the system minimax sound

The design of the system considers and counters various possible ways attackers can inflict damage on the performance of the system, to be discussed in Section 2.3.1 and 2.3.2.

2.3.1 Defending against attacks using spoofed IP addresses

An attacker has two options where it sends DDoS packets with spoofed IP addresses. The first option is to send a large volume of TCP SYN packets to the victim. The second option is to send a large volume of other types of packets (e.g., TCP ACK). We will show that the proposed HTTP redirection technique renders the second option useless.

There are two incentives for an adversary to send a large volume of TCP SYN packets (the first option) to the victim:

- First, such TCP SYN flooding may deplete the web server data structures for half-open TCP connections [7], if the server is not properly protected. The proposed system eliminates this problem by adopting standard countering techniques (discussed in Section 5). So in the following, we focus on the second incentive.

- Second, the large volume of TCP SYN packets from attackers with spoofed IP addresses are indistinguishable from the very first TCP SYN packet from a legitimate client. So the perimeter routers have to indiscriminately drop a high percentage of TCP SYN packets. When this happens, the first TCP SYN packets from legitimate clients will also suffer heavy loss and therefore experience a noticeable delay due to TCP retransmissions. Some clients may quit after they have waited for more than a certain amount of time (e.g., 8 seconds as assumed above). The strategy to counter this is to allocate adequate amount of bandwidth to such packets so that legitimate TCP SYN packets will have a decent probability to go through.

Once the very first TCP SYN packet of a client gets through, the proposed system immediately redirects the client to a pseudo IP address (still belonging to the web site) and port number pair, through a standard HTTP URL redirect message. Certain bits of this IP address and the port number pair will serve as the Message Authentication Code (MAC) for the client's IP address. MAC is a symmetric authentication scheme that allows a party A, which shares a secret key k with another party B, to authenticate a message M sent to B with a signature $MAC(M, k)$. The signature $MAC(M, k)$ has the property that, with overwhelming probability, no one can forge it without knowing the secret key k. By the above assumption, the perimeter routers will share a secret MAC key with the firewall. So the perimeter routers will be able to check the validity of MAC's and allow packets with correct MAC's to pass. Note that a client should not know k. It also does not need to know k, since $MAC(A, k)$ is computed by the system and sent to its claimed address A. The details of this protocol is discussed in Sec. 5.1.

Since a legitimate client uses its real IP address to communicate with the server, it will receive the HTTP redirect message (hence the MAC). So all its future packets will have the correct MAC's inside their destination IP addresses and thus be protected. The DDoS traffic with spoofed IP addresses, on the other hand, will be filtered because the attackers will not receive the MAC sent to them. So this technique effectively separates legitimate traffic from DDoS traffic with spoofed IP addresses.

The proposed system may potentially be vulnerable to a "replay" attack if proper countermeasures are not taken. In this attack, an adversary may first obtain valid MAC's for the IP addresses of some legitimate clients (e.g., by using a university or library host to access the victim) during a "pre-play" stage, which triggers the proposed DDoS defense system (hence the valid MAC's), before launching a major DDoS attack. Then these (valid) MAC's will be "replayed" during the major attack to (1) pose as these legitimate clients to consume network bandwidth, and (2) frame these clients by sending a huge volume of traffic using their IP addresses (with valid MAC's collected during the "pre-play"). An effective countermeasure is to have the MAC key evolve over time[5] and use a small "timestamp" (e.g., 2 bits) in the packet header to indicate which (recent) MAC key is in use. When the expiration time is set to a reasonably small value (e.g., 30s), an adversary will not be able to collect a large number of "fresh" MAC's, without compromising these clients.

2.3.2 Defending against attacks using genuine IP addresses

Attackers may also pose as legitimate clients and send legitimate HTTP requests to consume the bandwidth of the proposed system. Our URL redirection technique does not prevent this type of attack because these attackers are using their genuine IP addresses and will be able to receive the MAC. To address this problem, the firewall will perform fair bandwidth allocation among all clients and attackers that use genuine IP addresses. Deficit Round Robin (DRR) [37] is chosen as the packet scheduling algorithm since it has low implementation complexity ($O(1)$) and provides tight fairness guarantee. If an attacker sends packets much faster than its fair share, the scheduling policy will drop its excess traffic. Moreover, for each genuine IP address, the firewall will perform accounting on the number of packets that reach the firewall but are dropped by the scheduler. Once a host is found to have more packets dropped than a threshold H, its IP address will be blacklisted. The perimeter routers will be informed to drop all packets from that IP address. In general, a legitimate client will not have too much traffic dropped because it will adjust its sending rate around its fair share due to the TCP congestion control mechanism [42].

Determining the aforementioned threshold H involves a compromise between two con-

[5]There is no need for frequent key distribution to perimeter routers since later keys can be derived from the first key using a secure hash function.

flicting security issues. On the one hand, it is desirable for this H to be as small as possible because the system should not allow an attacker to send a large volume of traffic over its fair share without being punished. On the other hand, H has to be large enough to prevent legitimate clients from being accidentally blacklisted and to prevent the attackers from "framing" innocent IP addresses by guessing (brute-force) their corresponding MAC's. Typically, setting H to be a few thousand packets achieves a nice compromise. A detailed discussion can be found in the extended version of this paper [43].

In response to this fair queuing strategy, an attacker has two counter-strategies. One is to "bomb" the web site with huge volumes of traffic (using genuine IP addresses) and eventually get itself (its IP address) blacklisted, acting like "Kamikaze." We found that even when all the attackers perform "Kamikaze" together, it will typically take no more than several minutes for all of them to get blacklisted. So it is not a rational strategy for the adversary in the game theoretical sense [20].

The final strategy of the attack is to simply "keep low profile" and steal a fair share of bandwidth. We found when there are a large number of attackers, this may cause considerable degradation of service in the form of much longer web page download time for legitimate clients. One possible way to counter this type of "nonviolent" attack is to enforce the following "no loitering" law. The idea is that the total number of packets a client will send during a web transaction is not very large (several hundred to a few thousand as shown in Table 1). The firewall can identify and punish suspicious users by checking whether a user "loiters" in the system after its business with the system should be over. The system can set a quota Q such that the probability for a legitimate transaction to send more than Q packets is very small. After an IP address has sent more than Q packets, it will be given only a tiny fraction (say 1/10) of its fair share. This effectively limits the amount of bandwidth attackers can consume. In our performance modeling and simulation study, we will set Q to be three times the average number of packets in a web transaction. Note that these "punished" users are allowed to use excess bandwidth if there is any (i.e., they just have to yield). In practice, this quota Q should be set according to the normal transaction behavior profiled at the protected web site. We recognize that sometimes collateral damage is unavoidable: legitimate users may be accidentally suspected of "loitering" and have their services degraded. However, the benefit that the proposed system offers during an attack clearly outweighs such damage.

2.4 Summary

From the above discussion, we can see that it is most effective for an adversary to use a combination of the following two strategies:

- Command the attackers to send a large volume of TCP SYN packets using spoofed IP addresses. This makes it harder for a legitimate client to get its first packet through the perimeter routers.
- Command the attackers to consume a fair share of firewall bandwidth using their genuine IP addresses.

We have shown that other strategies such as "framing innocent IP addresses" and "Kamikaze" does not work as effectively as the above two. We acknowledge, however, that this does not constitute a rigorous proof that the system design is minimax sound, although every effort is made to identify all possible ways an adversary can attack our system. In general, it is very hard to take into consideration all possible attack scenarios given the complexity of a system. In the next section, we use a game theoretical approach to study the worst-case performance of the system when the adversary is using a combination of the aforementioned two strategies.

3 The Performance of the Proposed System

In this section, we study the minimax performance of the system using a novel game-theoretical framework. In this game, the proposed system and the adversary are fully aware of the set of possible strategies the other party has. The goal of the proposed system is to maximize a *system utility function* while the adversary's goal is to minimize it. The system utility function in this context is the total client satisfaction rate, defined as the number of new clients (per second) that eventually make their way to the system, multiplied by the average satisfaction of each client. Two utility functions will be introduced to model the average satisfaction of each client as a function of the average bandwidth it has received.

Notations used in the analysis:
A : arrival rate of legitimate clients
B : total bandwidth of the firewall
Y : bandwidth given to unprivileged traffic
$B - Y$: bandwidth given to privileged traffic
N : total number of attackers
X : number of attackers sending unprivileged traffic
Z : number of attackers sending privileged traffic
α: the average sending rate of an attacker
W : average amount of traffic a client sends during the whole web transaction
b : effective per-client bandwidth when there is no DoS
R : effective per-client bandwidth when there is DoS
p : percentage of unprivileged traffic that passes through the perimeter routers
$f(p)$: given p, the percentage of the clients that eventually get into the system
$d(p)$: given p, the average initial delay (to the very first SYN packet) a client has experienced before getting into the system
T : total page download time during a web transaction in average

As explained before, an adversary's rational strategy set consists of combinations of two substrategies: (a) command the attackers to send TCP SYN packets with spoofed IP addresses, and (b) command the attackers to consume a fair share of bandwidth using their genuine IP address. We refer to the former type of traffic as *unprivileged traffic* and the latter type as *privileged traffic*. The parameters under the adversary's control is X, the number of attackers that send unprivileged traffic, and Z, the number of attackers that send privileged traffic. In this paper, we assume that both can be as large as the total number of attackers N. However, there can be situations where each attacker can only send one type of traffic (i.e., $X + Y \leq N$). For example, if an effective IP traceback scheme is deployed, it may be able to identify and blacklist an attacker that sends both types of traffic.

The parameter that is under the control of the proposed system is Y, the amount of bandwidth allocated to allow unprivileged traffic to go through. The remaining $B - Y$ is allocated to privileged traffic, where B is the total bandwidth of the firewall. Note that if Y is set to 0, no legitimate client can get its first SYN packet through. On the other hand, it also should not be set too high. Otherwise $B - Y$ will be too small for the privileged traffic. So Y needs to be set in a way that allows just the right number of legitimate clients to go through without consuming too much of the firewall bandwidth.

We will show that the overall system utility can be written as $g(X, Y, Z)$. Then according to the game theory [20], the minimax (worst-case for the proposed system) utility of the proposed system is:

$$\underset{Y}{max} \; \underset{X,Z}{min} \quad g(X, Y, Z) \tag{1}$$

For both parties, the parameters X, Y, Z should be set to the values with which this minimax utility is achieved. Neither party has the incentive to unilaterally deviate from the minimax solution because if one does, the other party can gain more by choosing a strategy that takes advantage of this deviation. In the following, we explicitly derive the function g in (1).

We denote as A and W the arrival rate of new clients and the average amount of traffic each client will send during the whole web transaction, respectively. When there is no attack, each client still has an upper limit on its effective bandwidth (denoted as b). So when there is no attack, $\frac{A*W}{B}$ is the load of the system and $\frac{W}{b}$ is the total page download time in a web transaction. Let α be the average rate an attacker can generate unprivileged traffic and p be the percentage of the unprivileged traffic that the firewall will allow to pass. We calculate p

as $p = \frac{Y}{X*\alpha}$ because among the $X * \alpha$ unprivileged packets (per second) that arrive, only Y will be allowed to pass. The arrival rate of the first SYN packets of legitimate new clients are not considered here because they are negligible compared to $X * \alpha$.

Let $f(p)$ denote the percentage of new clients that eventually have their first SYN packet get through and receive web service afterwards. Since a human user is willing to wait for 8 seconds for the response to his first TCP SYN packet, this means that four consecutive packet losses and retransmissions of the first SYN packet can be tolerated. In the default TCP setting, the timeout value for these four retransmissions are 0.5, 1, 2, and 4 seconds, respectively [41]. They (1+2+4+0.5) add up to 7.5 seconds. So $f(p) = \sum_{i=0}^{4} p * (1-p)^i = 1-(1-p)^5$. Let d(p) be the average delay of the very first SYN packets of new clients which eventually reach the victim. Then $d(p) = \frac{\sum_{i=0}^{4} 0.5*2^i*p*(1-p)^i}{f(p)} = \frac{1-(2-2p)^5}{2*p*(2p-1)*f(p)}$ according to the aforementioned default timeout setting.

Let R and T be the average bandwidth and total download time of a web transaction during the DoS attack. T and R are related by $T = \frac{W}{R}$. If a web transaction lasts T seconds, there are $A * T * f(p)$ concurrent legitimate clients according to Little's Law. Since there are also Z attackers who will take a fair share of the bandwidth, each client will receive up to $\frac{B-Y}{A*T*f(p)+Z}$ bandwidth thanks to the fair bandwidth allocation performed at the firewall ($B - Y$ is reserved for the privileged traffic). Since a client's bandwidth is limited by b, we get $R = min\{\frac{B-Y}{A*T*f(p)+Z}, b\}$. So

$$W = \frac{W}{R} * R = T * min\{\frac{B-Y}{A*T*f(p)+Z}, b\} \tag{2}$$

Solving for T, we get

$$T = \begin{cases} \frac{W*Z}{B-Y-W*f(p)*A} & : \quad b \geq \tau \\ \frac{W}{b} & : \quad otherwise \end{cases} \tag{3}$$

where $\tau = \frac{B-Y}{Z+f(p)*A*\frac{W*Z}{B-Y-W*f(p)*A}}$.

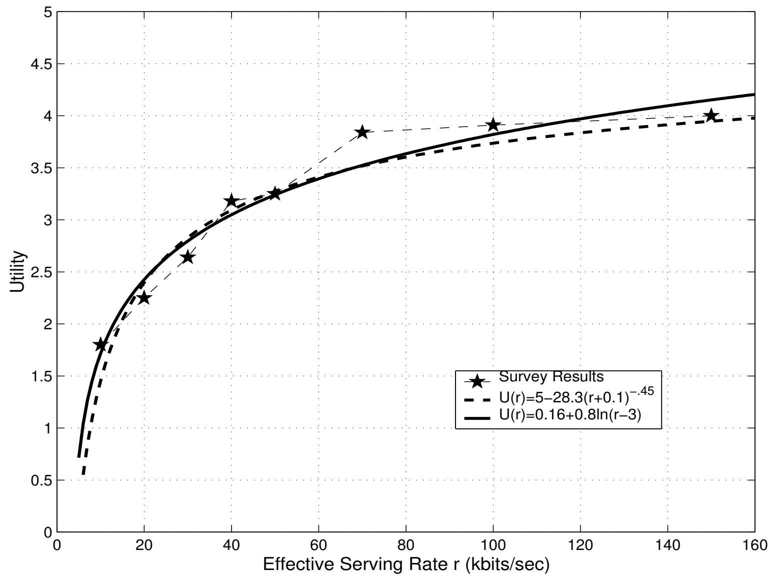

Figure 2: Survey and curve fitting results (adapted from [19])

The total client satisfaction rate, which is the metric to optimize, is $g(X, Y, Z) = f(p) * A * U(r)$. It can be verified that the right hand side is indeed a function of X, Y and Z. Here U is the user-perceived utility as a function of the average web page download rate $r = \frac{W}{d(p)+T}$. We will use two different utility functions in the following study. The first and folklore utility function (c is a constant) we will use is

$$U_1(r) = c * r, \qquad c > 0 \tag{4}$$

The second function we consider is an empirical utility curve obtained by a team of researchers in AT&T-labs [19]. They obtained the utility curve for web browsing through subjective surveys, in which users are asked to grade the performance of a web application under a range of network conditions. The testers (mimic users) were asked to give levels of satisfaction (subjective opinions on the quality of service) scaled from 1 to 5. The stars in Fig. 2 show average ratings obtained from the survey for web browsing running locally at various data transmission rates. Several concave curves are fit into the survey results. The exponential and the log curves fit the subjective survey very well when data transmission rates are from 10 to 150 kbps. These curves are

$$U_2(r) = 5 - 28.3(r + 0.1)^{-0.45} \tag{5}$$

$$U_3(r) = 0.16 + 0.8ln(r - 0.3) \tag{6}$$

Since these two curves are close to each other, we will only use U_2 in the following study.

3.1 The numerical results

We present a numerical example of (1) under a real-world scenario where each attacker can send both privileged and unprivileged packets. In this scenario, the constraints that X and Z need to satisfy are $X \leq N$ and $Z \leq N/10$. This $N/10$ comes from the "no loitering" law. We can see that, given any fixed Y, $g(X, Y, Z)$ decreases when either X or Z becomes larger. So the adversary's optimal strategy will always be $X = N$ and $Z = N/10$. The proposed system will then choose Y such that $g(Y, N, N/10)$ is maximized. So in this scenario, the minimax formula degenerates into a single variable optimization problem.

In this example, the system parameters are set as follows. The bandwidth of the firewall is assumed to be 400,000 ($B = 400,000$) inbound packets per second (pps), which is about 128 Mbps when each packet is of the minimum size (40 bytes). Each web transaction consists of 1000 ($W = 1000$) packets and a client's average effective bandwidth is assumed to be 40 pps. Both are reasonable web traffic volume and performance number [24, 9, 28, 15]. The traffic sending rate of an attacker is assumed to be 1000 packets per second, which is translated into 320 kbps with minimum packet size.

Using numerical methods, we obtain two set of minimax system performance results, corresponding to two aforementioned utility functions. Each set contains numerical results for two key metrics: (a) survival percentage of a legitimate client and (b) percentage of increase in total web page download time. Each metric is obtained for three load conditions:

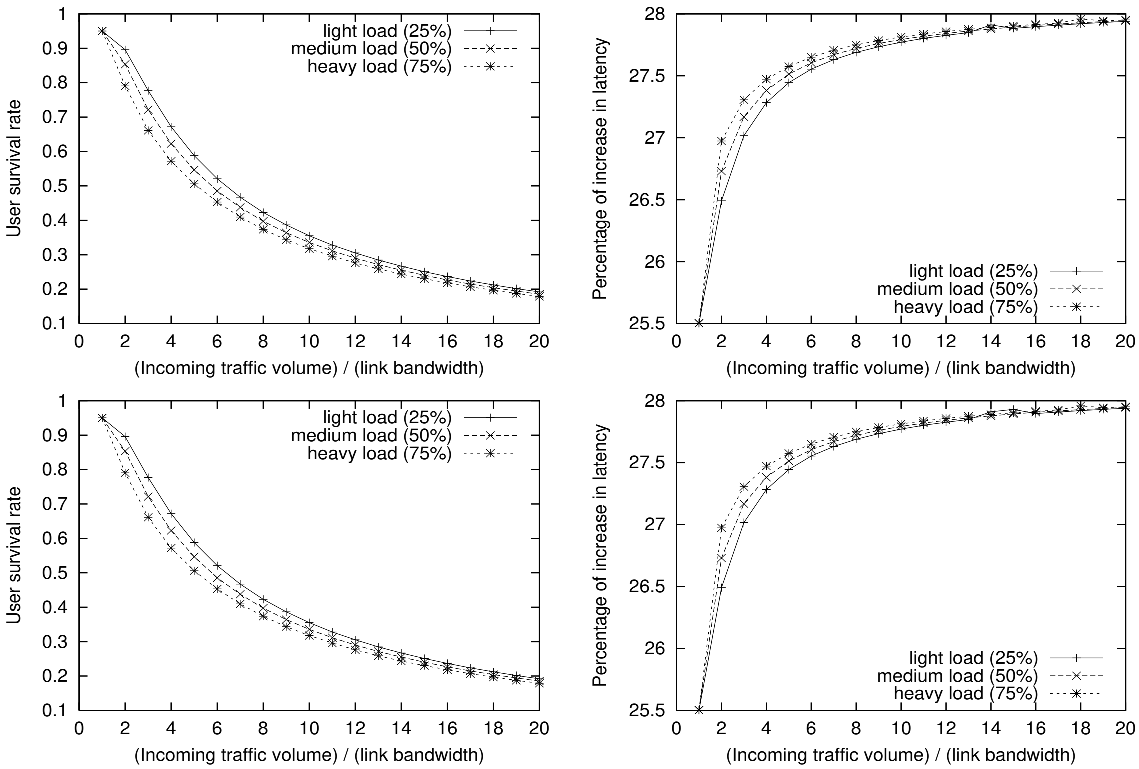

Figure 3: Top: (a)(b) The percentage of survival and the percentage of latency increase using utility function U_1. Bottom:(c)(d) The percent of survival and the percentage of latency increase using utility function U_2.

light (25%) load, medium (50%) load, and heavy (75%) load. The average arrival rate of new clients A is adjusted to generate these three load conditions (load is equal to $\frac{A*W}{B}$).

Fig. 3(a) and Fig. 3(b) show the client survival percentage and percentage of increase in total web download time when utility function U_1 is adopted. Each figure contains three curves corresponds to three different load conditions. Each curve shows how the corresponding metric changes when the amount of incoming traffic is between 1 to 20 times of the link bandwidth, representing from light to very severe DDoS attacks. We can see from Fig. 3(a) and Fig. 3(b) that even during severe DDoS attacks, the proposed system can render service to a decent percentage of clients, with a tolerable increase on the average page download time. For example, under medium load, when the incoming traffic is 5 times the link bandwidth (hence 80% packet loss), the system can continue to serve 55% of legitimate clients, at a tradeoff of 27.5% longer end-to-end page download time. The results shown in Fig. 3(c) and Fig. 3(d) are very similar to those shown in Fig. 3(a) and Fig. 3(b), even though a very different utility function (U_2) is used.

3.2 Measurement of parameters

In reality, we do not know some of the aforementioned parameters exactly. They will be estimated in an adaptive way. The system can measure and store B, W, and b when there are no attacks. When a DDoS attack happens, the system can estimate N and α by measuring the amount of traffic that arrives at the perimeter routers during the attack. These measurements do not need to be accurate at the beginning. The system will adapt to the optimal strategy by trying different $Y's$ in cautious steps.

4 Simulation Study

We conducted a simulation study using Berkeley Network Simulator (ns-2 [1]). The goal of this study is twofold:

- First, we verify a key assumption used in the game-theoretical modeling to make sure that the performance modeling results are close to actual performance of the system in the real-world operation. The assumption is that the scheduling algorithm we use (DRR) indeed achieves fair or weighted (for enforcing no loitering law) fair bandwidth allocation among web clients and attackers, even during a severe attack.
- Second, we would like to study the dynamics of bandwidth sharing under a DDoS attack. We will show how key metrics such as client bandwidth, page retrieval time, and

packet drop probability vary under various system load and attack severity conditions.

We emphasize that we are not verifying the whole game-theoretical analysis, since all assumptions except for the aforementioned fair bandwidth allocation are precisely captured in the analysis. So we will not be simulating the minimax performance of the system, since they have been studied in the analysis. Instead, in the following simulation, we assume that each attacker devotes 100% of its local bandwidth to steal a fair share bandwidth from the victim network, and none of them is sending any TCP SYN packets with spoofed IP addresses (unprivileged traffic). The proposed system, accordingly, devotes almost all system resource to the privileged traffic. Note that here neither the attackers nor the protection system are using their respective minimax strategies. However, simulation results under this condition are sufficient for us to achieve the aforementioned goals.

We choose ns-2 for our simulation because it provides ready-to-use simulation modules for studying the behaviors of HTTP and TCP protocols and various scheduling algorithms such as DRR. However, since it is not very memory-efficient, it limits the number of TCP clients we can create (at most a few thousands) and subsequently limits the size of other parameters. So the parameters used in the simulation will be smaller than used in the game-theoretical modeling. However, since the total link bandwidth is also proportionally smaller, the attack scenarios simulated in the following are actually more severe than analyzed in Sec. 3.

4.1 Simulation set-up

The single-bottleneck topology used in our simulation is shown in Fig. 4. A firewall router connects a large number of legitimate clients or attackers to the web server farm. The bandwidth and propagation delay of each link is assumed to be 1 Mbps and 10ms respectively. The inbound (from client to server) bandwidth of the firewall router is assumed to be 1

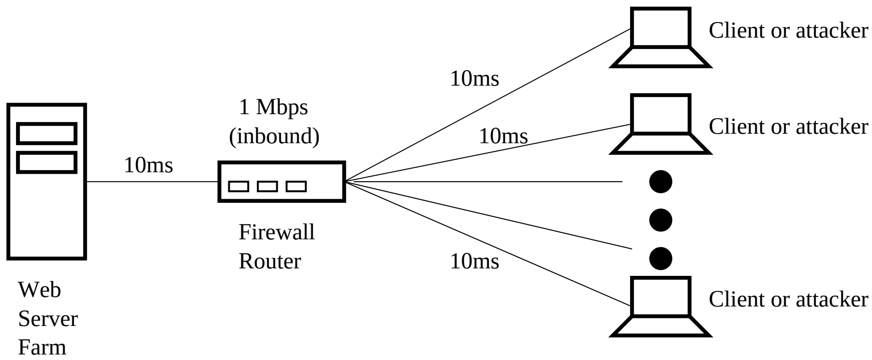

Figure 4: Network topology used in our simulation study

Mbps. Here we intend the firewall to be the performance bottleneck of the system. The outbound bandwidth of the firewall is essentially unlimited (modeled as 50 Mbps). Note that in the actual implementation, fair scheduling may be performed in both directions. Here we only simulate one direction since *in the simulation* for each outbound packet p, there is approximately one inbound packet ($p's$ TCP ACK) corresponding to it.

Total end-to-end propagation delay	10+10 = 20ms
Bandwidth (from client to server)	1Mbps
Scheduling algorithm	Deficit Round Robin (DRR)
DRR total buffer	10K bytes
DRR service quantum size	250 bytes
TCP version	TCP/Reno
HTTP request length	256 bytes
HTTP version	HTTP 1.0
Maximum concurrent TCP connections per client	4
Quota used in enforcing "no loitering" law	3000 packets

Table 2: Parameters used in the simulation

Our simulation parameters are summarized in Table 2. The firewall router will apply DRR to perform bandwidth allocation among all concurrent users. The total buffer size is 10K bytes. The quantum size of DRR is set to 250 bytes. Both attackers and clients are assumed to use HTTP 1.0. The number of concurrent connections per user are limited to 4. The type of TCP client is TCP/Reno.

For web traffic generation, we use a combination of a model introduced in [24] and another one introduced in a more recent study [9]. Each client requests 4 web pages from the server. Each page contains a main HTML file and embedded objects. There is a "think time" of about 15 seconds between two consecutive web requests. Throughout a web transaction, the total number of packets sent by a client (to server) is approximately 1000 packets. A client that has sent more than 3000 packets is suspected of "loitering" and will be given only 1/10 of a fair share.

4.2 Simulation results

Our simulation studies the following four set of metrics as a function of time, which allows us to verify the aforementioned key modeling assumption and to study the system performance dynamics under a DDoS attack.

1. Total throughput of attackers and legitimate clients.
2. A legitimate client's average download time of a web page.
3. Number of concurrent attackers and clients. Note that when an attacker has used up its quota, it will only be counted as 1/10.
4. Packet drop probability for attackers and legitimate clients.

We will study the above metrics under the following three scenarios:

- Severe attack (300 attackers) when the system is lightly loaded (25% load).
- Moderate attack (100 attackers) when the system is heavily loaded (75% load).
- Severe attack (300 attackers) when the system is heavily loaded (75% load).

We omit the case of moderate attack when the system is lightly loaded, since that result will obviously be better than in any of the above scenarios. Careful readers will notice that the number of attackers in our game-theoretical analysis are much larger (a few thousands) than used in the simulation. However, the attack here is actually more severe because the link speed is about one hundred times smaller. In all scenarios, the simulation starts from time 0 and lasts 30 minutes (1800 seconds). Legitimate clients will start in a uniform fashion during this period. All attackers will start between time 290s and 310s. Once they are started, they will continue to attack toward the end. Unlike legitimate clients, there is no "think" time between their HTTP requests. However, they do conform to TCP congestion control since they are "non-violent".

Fig. 5 shows the simulation results for the first aforementioned scenario: severe attack (300 attackers) under the light load condition. Fig. 5(a) shows that the total throughput of attackers suddenly jump to about 750 kbps around the time 300s, when the attack starts. Then it goes down to 600kbps around time 650s. This is exactly the time when most of the attackers have used up their "quota" and will only be given 1/10 of the fair share. This is confirmed in Fig. 5(c). Note that the attackers' bandwidth decreases only a little bit instead of 90% after time 650s. This is because under the light load condition, there are plenty of excess bandwidth (not used by clients) for them to use. Fig. 5(b) shows the average client page retrieval time. It starts at about 3.2s when there is no attack, jumps to about 5.3s between time 300s and 650s due to the arrival of 300 attackers, and drops to about 3.8s once these attackers use up their quota. The page retrieval time is longer after time 650s than before time 300s because 1/10 of the attackers are stilling competing with the clients for bandwidth. We verified using the numbers shown in Fig. 5(a) and Fig. 5(c) that DRR indeed guarantees approximately fair or weighted fair bandwidth allocation among clients and attackers. Fig. 5(d) shows that the packet drop probability for an attacker is much

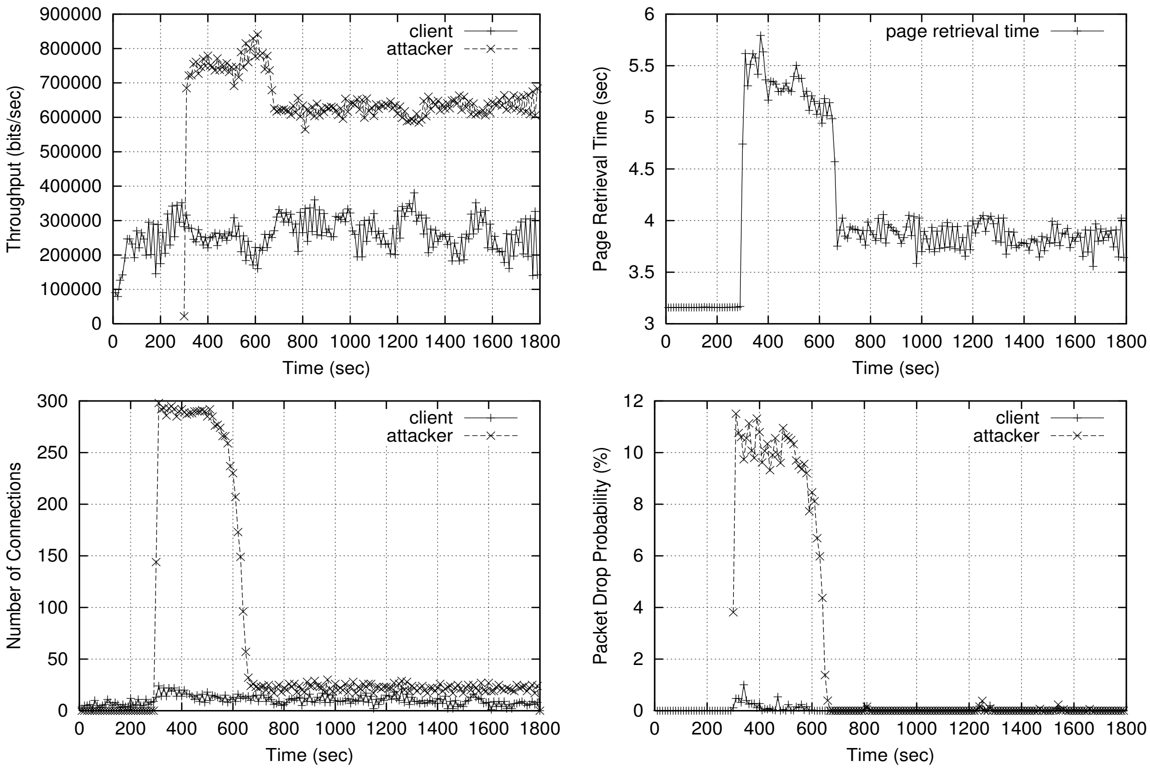

Figure 5: Light load with 300 attackers. (a) Total throughput of clients and attackers, (b) Client page retrieval time (averaged over a 10s time interval), (c) Number of concurrent clients and attackers, (d) Packet drop probability for a client and an attacker

higher (about 10%) than a client (close to 0) during the attack. In summary, the whole attack takes about six minutes (300s to 650s) to "die down".

Fig. 6 shows the simulation results for the second scenario: moderate attack (100 attackers) under heavy load condition. In Fig. 6(a), the total throughput of attackers jumps to about 250 kbps around time 300s and drops to about 200 kbps around time 720s, when their quota are used up. This is confirmed by Fig. 6(c). Fig. 6(b) plots the average client page retrieval time. It starts at about 3.2s when there is no attack, jumps to about 6.5s at time 300s, and gradually drops to about 4s when the "no loitering" law takes effect. We verified using the numbers shown in Fig. 6(a) and Fig. 6(c) that DRR indeed guarantees

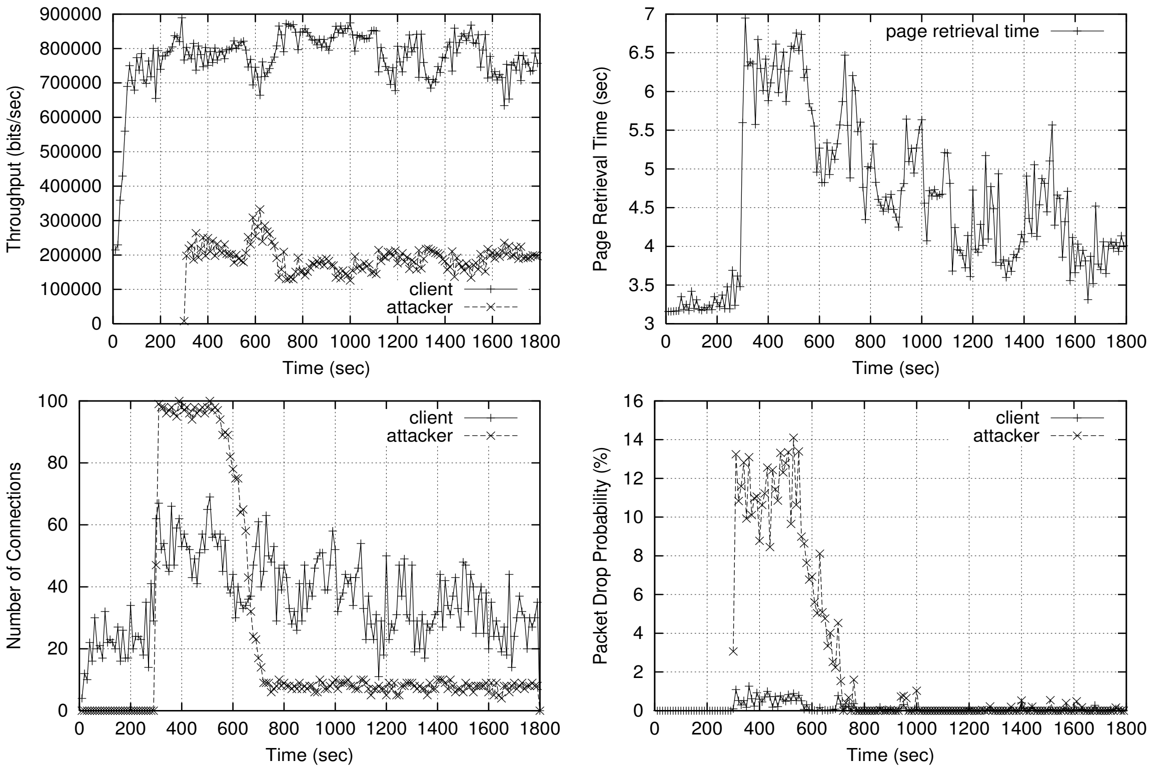

Figure 6: Heavy load with 100 attackers. (a) Total throughput of clients and attackers, (b) Client page retrieval time (averaged over a 10s time interval), (c) Number of concurrent clients and attackers, (d) Packet drop probability for a client and an attacker

approximate fair or weighted fair bandwidth allocation among clients and attackers. Fig. 6(d) shows that during the attack, the packet drop probability for an attacker is much higher (about 12%) than a client (about 0) during an attack. Overall, it takes about seven minutes (300s to 720s) for the attack to “die down”.

Fig. 7 presents the simulation results for the third scenario: severe attack (300 attackers) under the heavy load condition. Fig. 7(a) shows that the total throughput of attackers jump to about 300 kbps around the time 300s and stays around this level later on. Fig. 7(c) shows that the number of attackers goes down from 300 to about 30 around time 1300s, 17 minutes after the attack. This “die down” process is longer than in previous two scenarios because

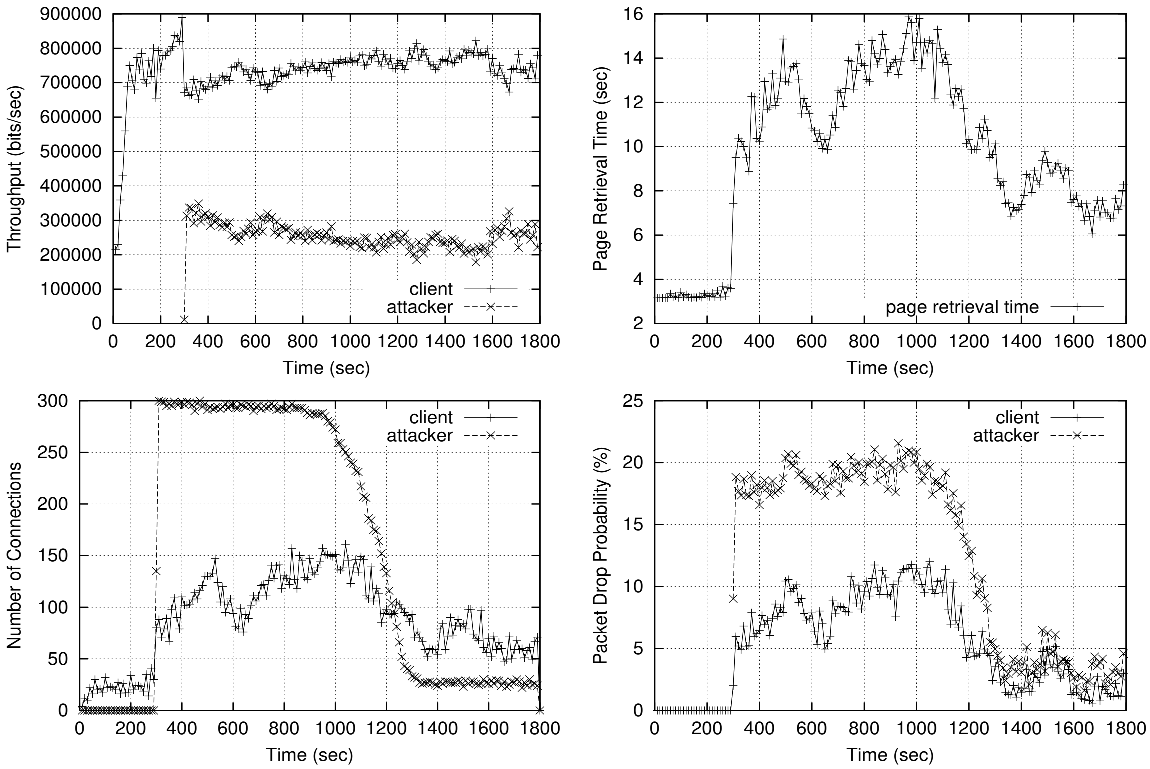

Figure 7: Heavy load with 300 attackers. (a) Total throughput of clients and attackers, (b) Client page retrieval time (averaged over a 10s time interval), (c) Number of concurrent clients and attackers, (d) Packet drop probability for a client and an attacker

under the heavy load, it takes much longer for an attacker to use up its quota. Fig. 7(c) also shows that the number of concurrent clients goes up from about 20 to between 100 and 150. This is because each client stays longer (longer page retrieval time) in the system after the attack begins, as shown in Fig. 7(b). We verified using the numbers in Fig. 7(a) and Fig. 7(c) that DRR again guarantees approximate fair or weighted fair bandwidth allocation among clients and attackers. Fig. 7(d) shows that the packet drop probability for an attacker and for a client is about 18% and 9% respectively during the attack. Both drops to about 4% after time 1300s, when the attack "dies down".

Through above simulations, we have achieved both of our aforementioned goals. We verified that DRR packet scheduling policy indeed guarantees fair bandwidth allocation between clients and attackers. We also show how different system metrics evolve as a function of time during a DDoS attack.

5 Implementation Issues

In this section, we discuss the issues involved in the implementation of the proposed system. The proposed system requires that operations be performed at two components of the system (shown in Fig. 1), namely, the perimeter routers and the firewall.

5.1 Operations performed at the perimeter routers

```
Operation performed at a perimeter router
    Upon arrival of a packet "pkt"
    IF (pkt.DST_IP != victim) THEN forward the packet and exit;
    IF (pkt.SRC_IP blacklisted) THEN drop the packet and exit;
    mac := MAC(pkt.SOURCE_IP, k);
    /* "k" is the MAC key, "||" denotes concatenation */
    IF (mac[1:18] == pkt.DST_IP[29:32] || pkt.DST_PORT[3:16]) THEN
        pass the packet and exit;
    IF (mac[19:40] ⊕ pkt.SRC_PORT == pkt.TCP_SEQ[11:32]) THEN
        pass the packet and exit;
    IF (pkt is SYN packet) THEN pass it with probability p;
    drop the packet;
```

Above is the algorithm of the operation performed at the perimeter routers. When a packet destined for the victim arrives, the algorithm first checks whether or not its source IP address is blacklisted and should be dropped. Then it identifies the traffic that belongs to protected class by verifying the correctness of the following two MACs.

The first MAC appears in the pseudo IP address and port pair that a web client will be redirected to. Here we describe a representative way to encode a MAC into this pair. The actual encoding may vary from system to system. We conservatively assume that the web site owns a network no smaller than a 28-bit IP prefix (consisting of 16 IP addresses). The algorithm, however, can work with smaller IP address space (bigger is better) or even a single IP address, with proper adjustments on other system parameters. Under this representative encoding, the web site uses the last 4 bits (host ID) from the IP address and lower 14 bits from the port number, to hold a 18-bit MAC of the source IP address claimed by the client. The first bit of the port number signals whether the port number is a regular port number or a MAC, and the second bit distinguishes between HTTP and HTTPS.

The second MAC is for the protection of the several packets that need to be sent by the client (TCP ACK packets) in order to receive the HTTP URL redirect message. They are protected using an extended form of SYN cookie technique (adapted from [17]). SYN cookie is a special TCP sequence number contained in a TCP SYN+ACK packet sent from a server to a client. It serves as the MAC for the client's IP address in the packet, which allows the server to verify later that the client has indeed received the packet [17]. It is originally designed to counter TCP SYN flood attack [7]. In our system, SYN cookie will be used for both countering the SYN flood attack and protecting the TCP ACK packets. The extended SYN cookie technique sets the first 22 bits of the TCP sequence number as a MAC and the last 10 bits to zero. Since the HTTP URL redirect message from the server is much shorter than 1024 bytes, the TCP acknowledgment numbers of these packets will share the same 22-bit prefix [32]. Therefore, perimeter routers are able to recognize such packets by checking whether the first 22 bits of the TCP acknowledgment number are the MAC of the port number and the source IP the client claims to be. In the above algorithm, we use "mac[19:40] $\oplus$ pkt.SRC_PORT" instead of computing another MAC (of both "pkt.SRC_IP" and "pkt.SRC_PORT") to save a MAC operation.

The operation performed at the perimeter router is lightweight in both space and CPU requirement. In terms of space, a hash table containing the IP addresses of a few thousand blacklisted attackers will be no more than 100K bytes. This is not comparable to the huge overhead of maintaining per-flow state in IntServ [44]. In terms of CPU cycles, we have shown that only one MAC operation needs to be performed. Such a MAC operation can be finished in about 1.1 microseconds on a commodity CPU processor, as shown in [43].

5.2 Operations performed at the firewall

In our system model (Fig. 1), the firewall is shown as one box and is considered as one abstract entity throughout this paper. In reality, it can be implemented as a number of boxes operating in parallel with same functionalities or as several boxes with different functionalities. The firewall will be enhanced to provide the following three functionalities.

First, it will perform standard connection interception [34] and SYN-cookie operation when an unprotected TCP SYN packet arrives. It should send back a SYN cookie as explained before. When a packet arrives from the client with correct SYN cookie, the firewall will establish a connection between a web server and the client. Also, the firewall should intercept HTTP requests for the default URL of the web site and respond with a URL redirect message containing the pseudo IP+port pair, as explained before.

Second, the firewall will apply fair bandwidth allocation among users, identifiable by their IP addresses, using the DRR packet scheduling policy [37]. Since here a flow is actually an IP flow (instead of TCP flow), the number of concurrent flows will be smaller than in the usual sense (TCP flow). Therefore, the space complexity of this operation is reasonable. Also, as explained before, the firewall will perform accounting on each of such IP address and check whether an IP address has sent too much over its fair share or has used up its quota (to enforce the "no loitering" law).

Third, the firewall will perform network address translation (NAT) so that the pseudo IP+port pair that serves as MAC for protected traffic will be translated into the actual IP address of a web server and actual port number (port 80 for HTTP and port 443 for HTTPS). The system can make this process completely "stateless" by using hash functions (similar techniques are used in [6] for network load-balancing purposes). Here we assume that web servers are identical to each other in terms of the content hosted and functionalities provided. In the other direction, when a web server sends a packet back to a client, the source IP address of the packet will be overwritten by the pseudo IP+port (calculated from its destination IP) before it leaves the web site.

Finally, we assume that there is a protocol that facilitates communication between the firewall and the perimeter routers. The design of this protocol is not complicated but is outside the scope of this paper[6]. The amount of information that needs to be conveyed is moderate, which only includes a secret key for verifying MAC, and a list of IP addresses that need to be blacklisted. Since such information is sensitive, packets carrying them need to be authenticated and encrypted. For example, they may run on top of IPSEC protocol [23].

6 Related Work

Internet Denial of service incidents began to be reported frequently after 1996 [16]. The most popular type of DoS attack is the TCP SYN flood attack [7]. Cryptographic [17, 21] and noncryptographic [36, 18] solutions have been proposed to address it. Recent large-scale distributed DoS attacks have drawn considerable attention [13]. Most of the proposed solutions have so far focused on IP traceback [4, 5, 10, 35, 39, 11, 38], that is, to trace the origin(s) of an attack. While the traceback schemes are valuable in finding the exact location

[6]The same protocol as proposed in [25] may be adopted with packet format modifications.

of the attacker and (hopefully) punishing the hacker after the fact, they are in general not able to mitigate the effect of a DoS attack while it is raging on. Also, lack of authentication in most of these techniques enables attackers to produce false traceback information to confuse the victim, as analyzed by Park [29].

Research has been done in other aspects of the distributed DoS problem. Gil proposes an attack-resistant data structure to enable routers to detect ongoing DoS attacks [14]. Zhou and Schneider propose an on-line certificate authority [45] which is robust against DoS attacks. Techniques to mitigate the effect of distributed DoS attacks have been studied in [22], in which attackers send bogus traffic aggressively only using their real IP addresses. Their technique is to isolate traffic sent by aggressive IP addresses, from other traffic sources. Though effective against this particular form of attack, it is vulnerable to other forms of DoS attacks. For example, it has no effective measure to defend against DoS packets sent using spoofed IP addresses. Also, if the attackers just behave like normal users to take a "fair share" of service, the system has no reliable way to distinguish them. These problems are addressed fully in our proposed system. Spatscheck and Peterson [40] implement mechanisms in the Scout operating system for detecting and mitigating network DoS attacks such as SYN flood [7]. However, these mechanisms require a principal to be properly authenticated. This may not always be possible for all network services. Also, authentication protocols may themselves become a target for DoS attacks [26].

Park [30] proposes to install packet filters at autonomous systems in the Internet to filter packets traveling between them. It is shown in [30] that when 20% of strategically chosen autonomous systems install such filters, most of the packets with randomly generated IP address (usual sense of IP spoofing) can be dropped. However, this requires the cooperation of thousands of autonomous systems, every ingress/egress router of which has to install the filter. Also, the attacker can still spoof IP addresses, albeit within a much smaller domain (e.g., a few autonomous systems).

One technique to mitigate the effect of DoS attacks is proposed in [25]. Recall that our DDoS defense system adopts a system model that is similar to [25]. In [25], each perimeter router is required to perform rate limiting on the amount of traffic destined for the victim network. Each router sets a threshold on the traffic rate destined for the victim. The amount of traffic over the threshold will be *randomly* dropped. It is shown in [25] that the scheme may be able to improve the throughput of legitimate traffic, when DDoS traffic only congests a small subset of the perimeter routers that legitimate traffic goes through. However, the effecitveness of the scheme is limited by the fact that a perimeter router has no way to distinguish between legitimate and DDoS traffic. Therefore it has to drop packets indiscriminately. So it offers little help when the ratio of legitimate traffic to DDoS traffic is similar among the perimeter routers (i.e., equally contaminated).

7 Contributions

The main contributions of this work can be summarized as follows:

- We designed a system that effectively sustains the availability of web services even during severe DDoS attacks. Our system is practical and easily deployable, because it is transparent to both web servers and clients, and is fully compatible with all existing network protocols. Since the web is the core technology underlying E-commerce and a primary target for recent DDoS attacks, this work offers a practical solution to a very important security problem.

- We proposed a novel game theoretical framework that accurately models the performance of our system as the minimax solution between conflicting goals of the adversary and the proposed system. Since all DoS problems contain such an adversarial relationship in nature, we expect this model to be also useful for analyzing the performance of other DoS problems and solutions.

- We performed a simulation study to verify a key assumption used in the game-theoretical analysis. The simulation study also exhibits the system dynamics under various system load and attack severity conditions.

- The design of our system is well engineered to address various security and performance considerations. The design is very amenable to implementation since it uses or customizes standard techniques (e.g., DRR, MAC, NAT, SYN cookie) that have been well developed and validated.

Acknowledgment

We thank guest editors for coordinating an expeditious review of our submission. We also thank anonymous reviewers for their constructive suggestions that help improve the quality and readability of this paper.

References

[1] Ucb network simulator - ns (version 2), 2001.

[2] The economic impacts of unacceptable web site download speeds. Technical report, Zona Research Inc., available at http://www.keynote.com/solutions/assets/applets/wp_downloadspeed.pdf, 1999.

[3] *Distributed Denial of Service Attack Tools*, 2001.

[4] S. Bellovin. Internet draft: Icmp traceback messages. Technical report, Network Working Group, March 2000.

[5] H. Burch and B. Cheswick. Tracing anonymous packets to their approximate source. In *Prof. Usenix LISA 2000*, December 2000.

[6] Z. Cao, Z. Wang, and E. Zegura. Performance of hashing-based schemes for internet load balancing. In *Proc. Infocom'2000*, March 2000.

[7] CERT. Tcp syn flooding and ip spoofing attacks. Advisory CA-96.21, September 1996.

[8] T. Chen and S. Liu. *ATM Switching Systems*. Artech House, Boston, MA, 1995.

[9] H. Choi and J. Limb. A behavior model of a web traffic. In *Proc. ICNP'99*, September 1999.

[10] D. Dean, M. Franklin, and A. Stubblefield. An algebraic approach to ip traceback. In *Proc. NDSS 2001*, pages 3–12, February 2001.

[11] T. Doeppner, P. Klein, and A. Koyfman. Using router stamping to identify the source of ip packets. In *Proc. ACM CCS-7*, pages 184–189, November 2000.

[12] R. Ganesan. Yaksha: Augmenting kerberos with public-key cryptography, 1995.

[13] L. Garber. Denial-of-service attacks rip the internet. *IEEE Computer*, 33(4):12–17, April 2000.

[14] T. Gil and M. Poletto. Multops: a data-structure for bandwidth attack detection. In *Proc. 10th Usenix Security Symposium*, August 2001.

[15] J. Heidemann, K. Obraczka, and J. Touch. Modeling the performance of http over several transport protocols. *IEEE/ACM Transaction on Networking*, 5(5):616–630, October 1997.

[16] J. Howard. *An Analysis of Security Incidents on the Internet*. PhD thesis, Carnegie Mellon University, August 1998.

[17] IETF. *Photuris: Session-Key Management Protocol*, March 1999.

[18] Checkpoint Inc. Tcp syn flooding attack and the firewall-1 syndefender. http://www.checkpoint.com/products/firewall-1/syndefender.html, 1997.

[19] Z. Jiang, Y. Ge, and Y. Li. Max-utility wireless resource management for best effort traffic, January 2002.

[20] A. Jones. *Game Theory: mathematical models of conflict*. John Wiley & Sons, 1980.

[21] A. Juels and J. Brainard. Client puzzles: A cryptographic countermeasure against connection depletion attacks. In *Proc. of NDSS'99*. Internet Society, March 1999.

[22] F. Kargl, J. Maier, S. Schlott, and M. Weber. Protecting web servers from distributed denial of service attacks. In *Proc. WWW-10*, May 2001.

[23] S. Kent and R. Atkinson. *Security Architecture for the Internet Protocol*. IPSEC Working Group, May 1998.

[24] B. Mah. An empirical model of http network traffic. In *Proc. Infocom'97*, April 1997.

[25] R. Mahajan, S. Bellovin, S. Floyd, J. Ioannidis, V. Paxson, and S. Shenker. Controlling high bandwidth aggregates in the network. Technical report, ACIRI and AT&T Labs Research, February 2001.

[26] C. Meadows. A formal framework and evaluation method for network denial of service. In *Proceedings of the 1999 IEEE Computer Security Foundations Workshop*, Mordano, Italy, June 1999.

[27] C. Neuman and T. Ts'o. Kerberos: An authentication service for computer networks, from IEEE communications magazine, september, 1994. In *William Stallings, Practical Cryptography for Data Internetworks, IEEE Computer Society Press, 1996*. 1996.

[28] V. Padmanabhan and J. Mogul. Improving http latency. *Computer Networks and ISDN Systems*, 28(1&2), December 1995.

[29] K. Park and H. Lee. On the effectiveness of probabilistic packet marking for ip traceback under denial of service attack. In *Proc. IEEE Infocom'2001*, April 2000.

[30] K. Park and H. Lee. On the effectiveness of route-based packet filtering for distributed dos attack prevention in power-law internets. In *Proc. ACM Sigcomm'2001*, August 2001.

[31] C. Partridge et al. A 50-gb/s ip router. *IEEE/ACM Trans. on Networking*, 6(3):237–248, June 1998.

[32] J. Postel. Rfc 793: Transmission control protocol. Technical report, Internet Society, September 1980.

[33] Michael K. Reiter, Matthew K. Franklin, John B. Lacy, and Rebecca N. Wright. The omega key management service. In *ACM Conference on Computer and Communications Security*, pages 38–47, 1996.

[34] A. Rice. Defending networks from syn flooding in depth. Technical report, Sans Institute, December 2000.

[35] S. Savage, D. Wetherall, A. Karlin, and T. Anderson. Practical network support for ip traceback. In *Proc. ACM SIGCOMM 2000*, pages 295–306, August 2000.

[36] C. Schuba et al. Analysis of a denial of service attack on tcp. In *Proceedings of the 1997 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, 1997.

[37] M. Shreedhar and G. Varghese. Efficient fair queuing using deficit round robin. In *Proc. of ACM SIGCOMM'95*, pages 231–242, August 1995.

[38] A. Snoeren, C. Partridge, et al. Hash-based ip traceback. In *Proc. ACM Sigcomm'2001*, August 2001.

[39] D. Song and A. Perrig. Advanced and authenticated marking schemes for ip traceback. In *Proc. Infocom 2001*, April 2001.

[40] O. Spatscheck and L. Peterson. Defending against denial of service attacks in scout. In *Proceedings of the 1999 USENIX/ACM Symposium on Operating System Design and Implementation*, pages 59–72, February 1999.

[41] W. Stevens. *TCP/IP Illustrated Volume 1, The Protocols.* Addison-Wesley, 1994.

[42] B. Suter, T. Lakshman, D. Stiliadis, and A. Choudhury. Design considerations for supporting tcp with per-flow queueing. In *Proc. of IEEE INFOCOM'98*, March 1998.

[43] J. Xu. Sustaining availability of web services under severe denial of service attacks. Technical report, Georgia Institute of Technology, May 2001.

[44] Lixia Zhang, Stephen Deering, and Deborah Estrin. RSVP: A new resource ReSerVation protocol. *IEEE network*, 7(5):8–18, September 1993.

[45] L. Zhou, F. Schneider, and R. Renesse. Coca: A secure distributed on-line certification authority. Technical report, Dept. of Computer Science, Cornell University, December 2000.