Distributed Denial of Service Attacks

Felix Lau
Simon Fraser University
Burnaby, BC, Canada
V5A 1S6
fwlau@cs.sfu.ca

Stuart H. Rubin
SPAWAR Systems Center
San Diego, CA, USA
92152-5001
srubin@spawar.navy.mil

Michael H. Smith
University of Calgary
Calgary, AB, Canada
T2N 1N4
mhs@mining.ubc.ca

Ljiljana Trajkovic
Simon Fraser University
Burnaby, BC, Canada
V5A 1S6
ljilja@cs.sfu.ca

Abstract

We discuss distributed denial of service attacks in the Internet. We were motivated by the widely known February 2000 distributed attacks on Yahoo!, Amazon.com, CNN.com, and other major Web sites. A denial of service is characterized by an explicit attempt by an attacker to prevent legitimate users from using resources. An attacker may attempt to: "flood" a network and thus reduce a legitimate user's bandwidth, prevent access to a service, or disrupt service to a specific system or a user. We describe methods and techniques used in denial of service attacks, and we list possible defenses. In our study, we simulate a distributed denial of service attack using ns-2 network simulator. We examine how various queuing algorithms implemented in a network router perform during an attack, and whether legitimate users can obtain desired bandwidth. We find that under persistent denial of service attacks, class based queuing algorithms can guarantee bandwidth for certain classes of input flows.

1. Introduction

We studied distributed denial of service attacks in the Internet such as the widely publicized, distributed attacks on Yahoo!, Amazon.com, CNN.com, and other major Web sites in February 2000. Even though denial of service attacks have existed for some time, their recent distributed formats have made these attacks more difficult to prevent.

In this paper we first summarize the methods involved in denial of service attacks, list possible defenses, and discuss in more depth the attack on Yahoo!. We then use a network simulator to study distributed denial of service attacks. Our simulation study examines how various queuing services in network routers may alleviate the problem of denying bandwidth to legitimate users during the denial of service attack. Finally, we use simulation results to recommend certain queuing algorithms that may protect users in cases of distributed denial of service attacks.

2. Characteristics of Distributed Denial of Service Attacks

A denial of service attack is characterized by an explicit attempt by an attacker to prevent legitimate users of a service from using the desired resources. Examples of denial of service attacks include [6]:

- attempts to "flood" a network, thereby preventing legitimate network traffic
- attempts to disrupt connections between two machines, thereby preventing access to a service
- attempts to prevent a particular individual from accessing a service
- attempts to disrupt service to a specific system or person.

The distributed format adds the "many to one" dimension that makes these attacks more difficult to prevent [17]. A distributed denial of service attack is composed of four elements, as shown in Figure 1 [4]. First, it involves a victim, i.e., the target host that has been chosen to receive the brunt of the attack. Second, it involves the presence of the attack daemon agents. These are agent programs that actually conduct the attack on the target victim. Attack daemons are usually deployed in host computers. These daemons affect both the target and the host computers. The task of deploying these attack daemons requires the attacker to gain access and infiltrate the host computers. The third component of a distributed denial of service attack is the control master program. Its task is to coordinate the attack. Finally, there is the real attacker, the mastermind behind the attack. By using a control master program, the real attacker can stay behind the scenes of the attack. The following steps take place during a distributed attack [7]:

1. The real attacker sends an "execute" message to the control master program.
2. The control master program receives the "execute" message and propagates the command to the attack daemons under its control.
3. Upon receiving the attack command, the attack daemons begin the attack on the victim.

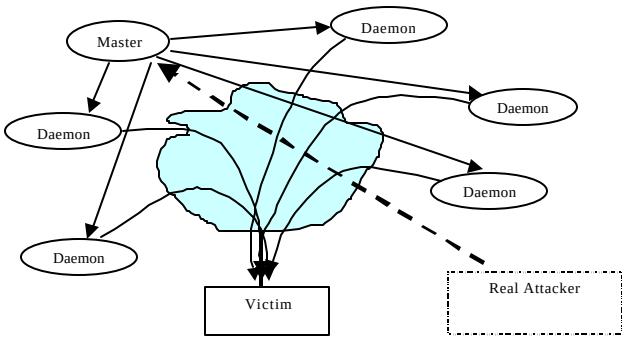

Figure 1: The four components of a distributed denial of service attack: a real attacker, a control master program, attack daemons and the victim [4].

Although it seems that the real attacker has little to do but sends out the "execute" command, he/she actually has to plan the execution of a successful distributed denial of service attack. The attacker must infiltrate all the host computers and networks where the daemon attackers are to be deployed. The attacker must study the target's network topology and search for bottlenecks and vulnerabilities that can be exploited during the attack. Because of the use of attack daemons and control master programs, the real attacker is not directly involved during the attack, which makes it difficult to trace who spawned the attack.

In the following subsections, we review some well-known attack methods (Smurf, SYN Flood, and User Datagram Protocol (UDP) Flood) and the current distributed denial of service methods (Trinoo, Tribe Flood Network, Stacheldraht, Shaft, and TFN2K). We describe defense mechanisms that can be employed by networks, and briefly review the Yahoo! attack.

2.1 Methods of Denial of Service Attacks

We described below some widely known basic denial of service attack methods that are employed by the attack daemons.

Smurf attack involves an attacker sending a large amount of Internet Control Message Protocol (ICMP) echo traffic to a set of Internet Protocol (IP) broadcast addresses. The ICMP echo packets are specified with a source address of the target victim (spoofed address) [9]. Most hosts on an IP network will accept ICMP echo requests [5] and reply to them with an echo reply to the source address, in this case, the target victim. This multiplies the traffic by the number of responding hosts. On a broadcast network, there could potentially be hundreds of machines to reply to each ICMP packet. The process of using a network to elicit many responses to a single packet has been labeled as an "amplifier" [16]. There are two parties who are hurt by this type of attack: the intermediate broadcast devices (amplifiers) and the spoofed source address target (the victim). The victim is the target of a large amount of traffic that the amplifiers generate. This attack has the potential to overload an entire network.

SYN Flood attack is also known as the Transmission Control Protocol (TCP) SYN attack, and is based on exploiting the standard TCP three–way handshake. The TCP three-way handshake requires a three-packet exchange to be performed before a client can officially use the service. A server, upon receiving an initial SYN (synchronize/start) request from a client, sends back a SYN/ACK (synchronize/acknowledge) packet and waits for the client to send the final ACK (acknowledge). However, it is possible to send a barrage of initial SYN's without sending the corresponding ACK's, essentially leaving the server waiting for the non-existent ACK's [3]. Considering that the server only has a limited buffer queue for new connections, SYN Flood results in the server being unable to process other incoming connections as the queue gets overloaded [8].

UDP Flood attack is based on UDP echo and character generator services provided by most computers on a network. The attacker uses forged UDP packets to connect the echo service on one machine to the character generator (chargen) service on another machine. The result is that the two services consume all available network bandwidth between the machines as they exchange characters between themselves. A variation of this attack called ICMP Flood, floods a machine with ICMP packets instead of UDP packets.

2.2 Methods of Distributed Denial of Service Attacks

In this section, we describe the distributed denial of service methods employed by an attacker. These techniques help an attacker coordinate and execute the attack. These types of attacks plagued the Internet in February 2000. However, these distributed attack techniques still rely on the previously described attack methods to carry out the attacks.

The techniques are listed in chronological order. It can be observed that as time has passed, the distributed techniques (Trinoo, TFN, Stacheldraht, Shaft, and TFN2K) have become technically more advanced and, hence, more difficult to detect.

Trinoo uses TCP to communicate between the attacker and the control master program. The master program communicates with the attack daemons using UDP packets.

Trinoo's attack daemons implement UDP Flood attacks against the target victim [10].

Tribe Flood Network (TFN) uses a command line interface to communicate between the attacker and the control master program. Communication between the control master and attack daemons is done via ICMP echo reply packets. TFN's attack daemons implement Smurf, SYN Flood, UDP Flood, and ICMP Flood attacks [10].

Stacheldraht (German term for "barbed wire") is based on the TFN attack. However, unlike TFN, Stacheldraht uses an encrypted TCP connection for communication between the attacker and master control program. Communication between the master control program and attack daemons is conducted using TCP and ICMP, and involves an automatic update technique for the attack daemons. The attack daemons for Stacheldraht implement Smurf, SYN Flood, UDP Flood, and ICMP Flood attacks [10].

Shaft is modeled after Trinoo. Communication between the control master program and attack daemons is achieved using UDP packets. The control master program and the attacker communicate via a simple TCP telnet connection. A distinctive feature of Shaft is the ability to switch control master servers and ports in real time, hence making detection by intrusion detection tools difficult [11].

TFN2K uses TCP, UDP, ICMP, or all three to communicate between the control master program and the attack daemons. Communication between the real attacker and control master is encrypted using a key-based CAST-256 algorithm [1]. In addition, TFN2K conducts covert exercises to hide itself from intrusion detection systems. TFN2K attack daemons implement Smurf, SYN, UDP, and ICMP Flood attacks [2].

2.3 Defenses Against Attacks

Many observers have stated that there are currently no successful defenses against a fully distributed denial of service attack. This may be true. Nevertheless, there are numerous safety measures that a host or network can perform to make the network and neighboring networks more secure. These measures include:

Filtering Routers: Filtering all packets entering and leaving the network protects the network from attacks conducted from neighboring networks, and prevents the network itself from being an unaware attacker [12]. This measure requires installing ingress and egress packet filters on all routers.

Disabling IP Broadcasts: By disabling IP broadcasts, host computers can no longer be used as amplifiers in ICMP Flood and Smurf attacks. However, to defend against this attack, all neighboring networks need to disable IP broadcasts.

Applying Security Patches: To guard against denial of service attacks, host computers must be updated with the latest security patches and techniques. For example, in the case of the SYN Flood attack [8], there are three steps that the host computers can take to guard themselves from attacks: increase the size of the connection queue, decrease the time-out waiting for the three-way handshake, and employ vendor software patches to detect and circumvent the problem.

Disabling Unused Services: If UDP echo or chargen services are not required, disabling them will help to defend against the attack. In general, if network services are unneeded or unused, the services should be disabled to prevent tampering and attacks.

Performing Intrusion Detection: By performing intrusion detection, a host computer and network are guarded against being a source for an attack, as while as being a victim of an attack. Network monitoring is a very good pre-emptive way of guarding against denial of service attacks. By monitoring traffic patterns, a network can determine when it is under attack, and can take the required steps to defend itself. By inspecting host systems, a host can also prevent it from hosting an attack on another network [19].

2.4 Yahoo! Attack

The attacks on the major Web sites began in early February 2000, with the first major attack being on Yahoo! on February 7 [15]. The surprise attack took the Yahoo! site down for more than three hours. It was based on the Smurf attack, and most likely, the Tribe Flood Network technique. At the peak of the attack, Yahoo! was receiving more than one gigabit per second of data requests.

Yahoo! has stated that it was unprepared for this magnitude of an attack, and hence, was not ready to defend itself. Yahoo! receives huge number of visitors every day, and most likely believed that the bandwidth provided to the users by their Internet service providers would defend them against any type of denial of service attacks. However, they did not account for a large distributed denial of service attack from numerous servers and networks across the Internet.

3. Simulation Scenarios

Using the simulation tools, we examine how various queuing algorithms implemented in a network router perform during an attack, and whether legitimate users can obtain desired service. We use the ns–2 network simulator [18] to simulate a distributed denial of service attack on a targeted router.

We simulate a simplified version of the distributed denial of service attack on a single targeted router, shown in Figure 2 [17]. We simulate the attack using UDP packets. Our main goal is to compare several queuing algorithms and to determine whether the queuing methods in the target router could provide a better share of bandwidth to the legitimate users during the attack. We measure the throughput provided to the legitimate users and to the attackers when using the following queuing algorithms: DropTail, Fair Queuing, Stochastic Fair Queuing, Deficit Round Robin, Random Early Detection, and Class Based Queuing (Packet by Packet Round Robin and Weighted Round Robin).

Figure 2: The simulated distributed denial of service attack scenario [17].

We conduct the ns-2 simulations using three network topologies. Each network topology consists of one legitimate user, one target host, and a varied number of attack daemons, as shown in Figure 3.

In our simulation scenarios, we use a single target router with a 1 Mbps bandwidth. All network links have 1 Mbps bandwidth, with a delay of 100 ms. The legitimate user is defined as a UDP agent sending packets of size 500 bytes at a rate of 0.1 Mbps. The attack daemons are defined as UDP agents sending packets of size 500 bytes at rates of 0.3 to 1.0 Mbps. All sources generated constant bit rate (CBR) traffic.

Figure 3: Simulated network topologies A, B, and C (left to right). Target is the right most node in the networks.

In our simulations, we use the following queuing algorithms available in ns-2:

- *DropTail* is a queuing algorithm based on a first-come-first-serve discipline.
- *Fair Queuing* is an algorithm that attempts to allocate bandwidth fairly among all input flows.
- *Stochastic Fair Queuing* involves a hash function used to map flows to a queue.
- *Deficit Round Robin* scheduling is an algorithm that services each flow in a predefined sequence.
- *Random Early Detection (Drop)* tries to anticipate congestion by monitoring the queue [13]. When the specified threshold is reached, it randomly discards or marks the packets.
- *Class Based Queuing (Packet by Packet Round Robin and Weighted Round Robin)* queues packets according to criteria defined by an administrator [14]. It provides differential forwarding service for each class. Packets are divided into a hierarchy of classes defined by input flows. For our simulations, we develop two classes of Class Based Queuing: a class of known flows (legitimate users) and a class of unknown flows. For each class, we allocated a minimum bandwidth allotment.

4. Simulation Results

We now discuss the results obtained from our simulation study. For each queuing algorithm, except DropTail, we simulate topologies A, B, and C.

In the case of DropTail queuing algorithm, we only simulate topology A because only two attackers were required to overload the target router. The legitimate user has 0.1 Mbps, while the two attackers each have allocated 0.6 Mbps bandwidth. Since the target router has a buffer size of 1 Mbps, and the input links have 1.3 Mbps bandwidth, overloading the buffer in the router only requires two attackers and packet loss in the router is to be expected. The legitimate user's bandwidth was reduced to zero once the attack executed by the two attack daemons was fully engaged. As shown in Figure 4, the user's bandwidth (bottom curve) falls to zero at approximately 2.5 sec after the beginning of the attack.

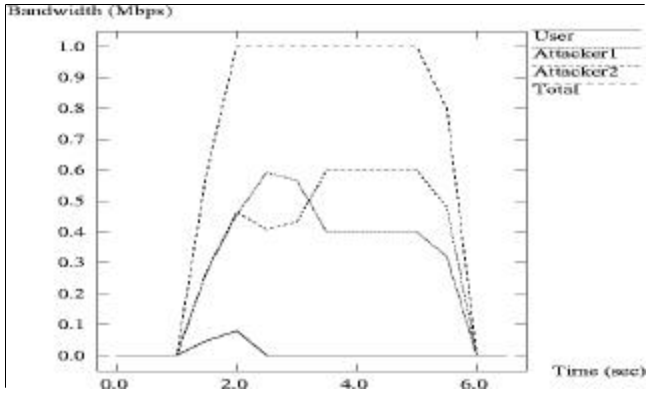

Figure 4: Simulation results using DropTail queuing algorithm and network topology A. Shown are User 1,

Attacker 1, Attacker 2, and the total bandwidth. Once the attack is fully engaged, the legitimate user (bottom curve) is left with little or no bandwidth.

In simulation scenarios using the remaining queuing algorithms, topologies A and B did not provide enough traffic overload to prevent legitimate users from receiving requested bandwidth. However, simulations based on topology C resulted in several target routers being overloaded during the attack by the twelve attack daemons. For the simulation scenario with topology C, we allocated 0.1 Mbps to the legitimate user, 0.5 Mbps bandwidth was given to attackers 1-6, and 1.0 Mbps was given to attackers 7-12. The total input bandwidth of the attackers is 9.0 Mbps.

Two queuing algorithms (Random Early Detection and Class Based Queuing) are successful in providing bandwidth requested by the legitimate user during the simulations using network topology C. Fair Queuing, Stochastic Fair Queuing, and Deficit Round Robin Queuing are algorithms that provided little or no bandwidth to the legitimate user during the attack. Although the user did not receive full throughput with Random Early Detection queuing, he/she continued to receive service through most of the duration of the attack scenario, as shown in Figure 5 (bottom).

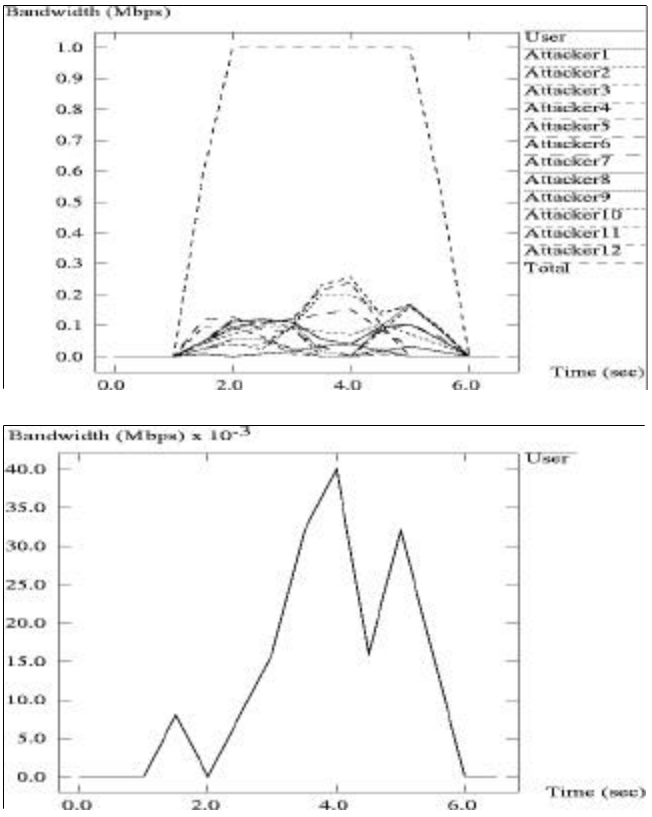

Figure 5: Simulation results using Random Early Detection queuing algorithm and network topology C. Shown are the legitimate user's bandwidth, the twelve attacker's bandwidth, and the total bandwidth for the target router (top), and the legitimate user's bandwidth (bottom).

Finally, two variants of the Class Based Queuing algorithms were successful in providing full bandwidth requested by the legitimate user. In fact, with appropriately defined flow classes (with known and unknown flags), the legitimate user was able to obtain all requested bandwidth, as shown in Figure 6 (bottom).

Unlike other queuing algorithms that we simulated, Class Based Queuing algorithms require extra overhead and effort to be implemented, because input flows to a Class Based Queuing based routers must be a priori categorized and managed. Hence, this queuing discipline may not be a feasible solution for routers with a large number of input links.

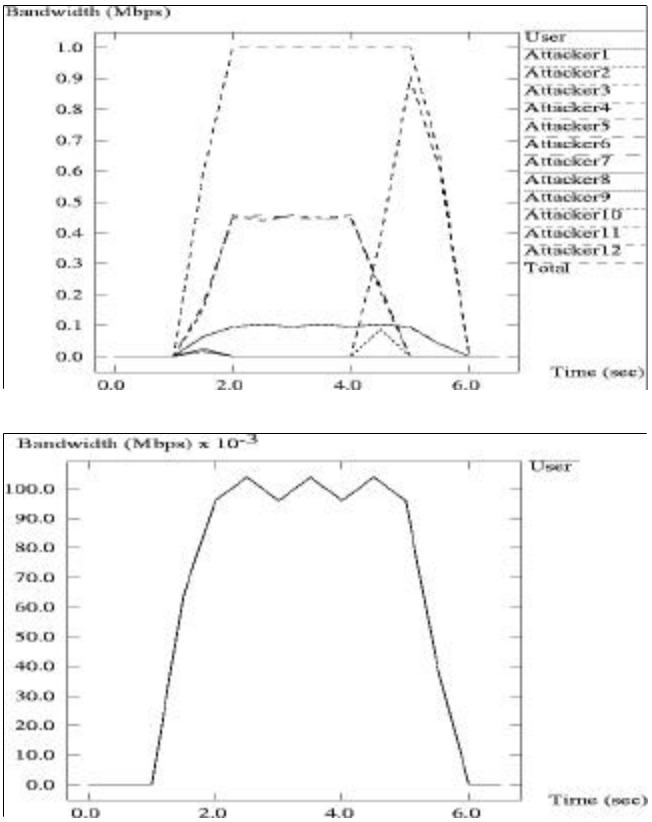

Figure 6: Simulation results using Class Based Queuing (packet by packet round robin) algorithm and topology C. The legitimate user obtained most of the requested 0.1 Mbps bandwidth. Shown are bandwidths for the legitimate user, twelve attack daemons, and the total bandwidth for the target router (top), and the bandwidth for the legitimate user only (bottom).

In summary, our simulation results indicate that Class Based Queuing had the best performance and provided the full bandwidth that the legitimate user requested. The Random Early Detection algorithm was the best among algorithms that required no additional overhead. It was able to provide 40% of the bandwidth requested by the user.

5. Conclusions

In this paper, we discussed distributed denial of service attacks on the Internet. We described how distributed attacks are conducted, we reviewed some well known distributed denial of service techniques, and discuss various defense mechanisms that could be employed by networks and hosts.

We used network simulation tools to examine the performance of various queuing algorithms in alleviating the distributed denial of service attacks and in providing desired service to the users. It was found that the majority of routing algorithms that we considered provided no bandwidth to the legitimate user during the attack. Nevertheless, even under persistent denial of service attacks, Class Based Queuing algorithm could guarantee bandwidth for certain classes of input flows, while Random Early Detection was successful in providing limited bandwidth to legitimate users. Since implementation of a Class Based Queuing algorithm required additional effort, there is a tradeoff between its performance and the implementation overhead. In summary, our simulation results indicated that implementing queuing algorithms in network routers may provide the desired solution in protecting users in cases of distributed denial of service attacks.

Acknowledgements

We thank V. Markovski for his assistance with the ns-2 network simulator. This research was supported by the NSERC Grant 216844-99 and the BC Advanced Systems Institute Fellowship.

References

[1] C. Adams and J. Gilchrist, "RFC 2612: The CAST-256 encryption algorithm," June 1999, http://www.cis.ohio-state.edu/htbin/rfc/rfc2612.html.

[2] J. Barlow and W. Thrower, "TFN2K – an analysis," Feb. 2000, http://packetstorm.securify.com/distributed/TFN2k_Analysis.htm.

[3] S. Bellovin, "Security problems in the TCP/IP protocol suite," *Comput. Commun. Rev.*, vol. 19, no. 2, pp. 32-48, Apr. 1989.

[4] S. Bellovin, "Distributed denial of service attacks," Feb. 2000, http://www.research.att.com/~smb/talks.

[5] S. Bellovin, Ed., "The ICMP traceback message," Network Working Group Internet Draft, Mar. 2000, http://www.research.att.com/~smb/papers/draft-bellovin-itrace-00.txt.

[6] CERT® Coordination Center, Cert Advisories: "CA-2000-01 denial-of-service developments," http://www.cert.org/advisories/CA-2000-01.html; "CA-99-17 denial-of-service tools," http://www.cert.org/advisories/CA-99-17-denial-of-service-tools.html; "CA-98-13-tcp-denial-of-service: vulnerability in certain TCP/IP implementations," http://www.cert.org/advisories/CA-98-13-tcp-denial-of-service.html.

[7] CERT® Coordination Center, "Results of the distributed systems intruder tools workshop," Nov. 1999, http://www.cert.org/reports/dsit_workshop.pdf.

[8] Cisco Systems, Inc., "Defining strategies to protect against TCP SYN denial of service attacks," July 1999, http://www.cisco.com/warp/public/707/4.html.

[9] Daemon9, Infinity, and Route, "IP-spoofing demystified: trust-relationship exploitation," *Phrack Mag.*, June 1996, http://www.fc.net/phrack/files/p48/p48-14.html.

[10] D. Dittrich, "The DoS project's 'Trinoo' distributed denial of service attack tool," Oct. 1999; "The 'Stacheldraht' distributed denial of service attack tool," Dec. 1999; "The 'Tribe Flood Network' distributed denial of service attack tool," Oct. 1999, http://www.washington.edu/People/dad.

[11] D. Dittrich, S. Dietrich, and N. Long, "An analysis of the 'Shaft' distributed denial of device tool," Mar. 2000, http://netsec.gsfc.nasa.gov/~spock/shaft_analysis.txt.

[12] P. Ferguson and D. Senie, "RFC 2267: Network ingress filtering: defeating denial of service attacks which employ IP source address spoofing," Jan. 1998, http://info.internet.isi.edu/in-notes/rfc/files/rfc2267.txt.

[13] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Trans. Networking*, vol. 1 no. 4, pp. 397-413, Aug. 1993.

[14] S. Floyd and V. Jacobson, "Link-sharing and resource management models for packet networks," *IEEE/ACM Trans. Networking*, vol. 3 no. 4, pp. 365-386, Aug. 1995.

[15] A. Harrison, “The denial-of-service aftermath,” Feb. 2000, http://www.cnn.com/2000/TECH/computing/02/14/dos .aftermath.idg/index.html.

[16] C. A. Huegen, “The latest in denial of service attacks: ‘Smurfing’ description and information to minimize effects,” Feb. 2000, http://users.quadrunner.com/chuegen/smurf.cgi.

[17] B. Martin, “Have script, will destroy (lessons in DoS),” Feb. 2000, http://www.attrition.org.

[18] ns-2 network simulator, http://www.isi.edu/nsnam/ns .

[19] T. H. Ptacek and T. N. Newsham, “Insertion, evasion, and denial of service: eluding network intrusion detection,” Secure Networks, Inc., Jan. 1998, http://www.clark.net/~roesch/ idspaper.html.