



### SIMULATION EXPERIMENTATION WITH SECURE OVERLAY SERVICES

Journal:	<i>Summer Computer Simulation Conference</i>
Manuscript ID:	draft
Track:	Military Application & Simulation
Date Submitted by the Author:	n/a
Complete List of Authors:	Fletcher, Harold; Auburn University, Computer Science & Software Engineering Richardson, Kevin; Auburn University, Computer Science & Software Engineering Carlisle, Martin; United States Air Force Academy, Computer Science Hamilton, John; Auburn University, Computer Science & Software Engineering
Keywords:	Bandwidth Denial of Service, Secure Overlay Services, Network Simulation, OPNET

# SIMULATION EXPERIMENTATION WITH SECURE OVERLAY SERVICES

**Harold W. Fletcher**  
fletchw@auburn.edu  
Computer Science &  
Software Engineering  
Auburn University  
Auburn, AL 36849

**Kevin Richardson**  
richakp@auburn.edu  
Computer Science &  
Software Engineering  
Auburn University  
Auburn, AL 36849

**Martin C. Carlisle, Ph.D.**  
Martin.Carlisle@usafa.af.mil  
Department of Computer  
Science  
U.S. Air Force Academy  
Colorado Springs, CO 80840

**J.A. Hamilton, Jr., Ph.D.**  
hamilton@eng.auburn.edu  
Computer Science &  
Software Engineering  
Auburn University  
Auburn, AL 36849

Keywords: Bandwidth Denial of Service, Secure Overlay Services, Network Simulation, OPNET.

## ABSTRACT

The problem of Distributed Denial of Service attacks present a clear threat to the US information infrastructure. As long as that threat exists, malicious users on the Internet have the capability to engage in pinpoint attacks of sites connected to the Internet, rendering them inoperable for minutes or even hours. Much of the work on this subject has focused on reactive approaches to defense, identifying and filtering attack packets, using trace back strategies to locate attackers, etc. In this paper we simulate one of the preventative solutions, Secure Overlay Services. Our simulation shows that SOS provides connectivity throughout even a very powerful and intelligent attack using a widely distributed overlay network.

## 1. INTRODUCTION

The highly publicized DDOS attacks of February 2000 brought the attention of the Internet community to this very real, very palpable threat. Yahoo, CNN.com, Ebay, Buy.com, Etrade and Amazon.com were among those companies whose web services were brought to a near standstill for several hours. According to the Yankee Group, a private communications research company, these attacks cost these and other e-businesses an estimated \$1.2 billion dollars due to capitalization losses, revenue losses and necessary security upgrades [Gonzalez 2000].

The brightest minds in networking research have been working for many years to find a solution to this problem. Dozens of simple stopgaps have been proposed and some implemented. For every feasible quick-fix there are also several solutions that are too big, involving fundamental changes to the way the Internet accomplishes routing and the way TCP handles traffic. In the face of the daunting task of a network overhaul, researchers at Columbia

University proposed a novel way to deal with attacks to maintain service during the throes of a large denial of service attack [Keromytis, Misra, & Rubenstein 2002]. This approach seemed promising for an applied project we are working on with the U.S. Air Force Academy. Though the original authors did rigorous mathematical validation and even created a limited implementation [Keromytis, Misra & Rubenstein 2002], there seems to have been no attempt at simulating, or implementing a full-scale version of an SOS network.

This paper attempts to document and characterize several different solutions and workarounds. In particular we also detail the results of an OPNET simulation we created to test the SOS architecture. We verify the experiments conducted in [Keromytis, Misra & Rubenstein 2004] and show that with careful maintenance of connections in the overlay, even during massive attacks connectivity can be maintained with down periods of 10 seconds or less.

Section 2 of this paper describes different types of DDOS attacks. Section 3 describes several solutions and characterizes them according to the DDoS Taxonomy [Mirkovic & Reiher 2004]. Section 4 details Secure Overlay Services and the underlying Chord Lookup service. Section 5.0 contains a brief description of our model, and Section 6.0 documents the results of our simulations. Finally we discuss future work

## 2. PROBLEM STATEMENT

Early Distributed Denial of Service attacks simply involved large numbers of zombie machines (machines somehow compromised to give a remote user control over them) sending huge amounts of traffic to victim machines. These attacks could take several different forms, from ICMP ECHO flooding (Smurf attack) [CERT 2005] to UDP ECHO flooding (Fraggle attack) down to subtler

This work was partially funded by the Academy Center for Information Security, at the United States Air Force Academy under research contract # F05611-03-D-0003.

attacks like TCP SYN flooding. SYN flooding uses TCP SYN packets with spoofed source addresses to tie up resources on a target machine, by opening many thousands of connections at once to bogus machines, the target will eventually experience resource starvation.

A dangerous trend that was noticed early in this decade was the change to reflected attacks. They differ from a normal flood attack by using a high-bandwidth intermediary or reflector to conduct the actual flood. These attacks use zombies to send connection requests or ECHO packets to some third party (i.e., commercial web server, high speed router) but with source addresses spoofed with the address of the target of the attack. The third party will then become an unwitting participant in the attack by sending acknowledgements or ECHO REPLY packets to a very confused victim machine that never initiated the connection it is now credited with [Garber 2000].

The goal of any solution to this problem, obviously, is to survive the attack while maintaining service to legitimate users. The problem can then be reduced to identifying legitimate traffic and allowing it access to network resources while keeping illegitimate traffic away from the network edges, near the core where higher speed routers can handle excess. Some proposed solutions to this problem are outlined in the following section.

### 3. PROPOSED SOLUTIONS

The authors of [Mirkovic & Reiher 2004] classify solutions to the DDoS problem as either reactive or preventative. Instead of thinking along that division, for the purposes of this paper we prefer to think about solutions being intended either for the source, the destination, or the nebulous middle ground of Internet routing.

#### 3.1 Source-Side Solutions

One source solution that has gained a lot of ground in the Internet community is Network Ingress Filtering [Ferguson & Senie 2000]. RFC-2827 describes a simple filter that could be installed in all edge routers, routers that provide a particular subnet access to the Internet. This filter simply checks all packets arriving from its internal side to make sure that the source address corresponds to the correct address range. This simple solution, if implemented over the entire Internet, can effectively stop IP spoofing—the exception to this being spoofing addresses which occur within the same subnet as the attacking machine.

Of course the obvious drawback to this sort of solution is that for it to work completely it would have to be implemented on the many, many thousands of edge routers that exist on the Internet. This solution crosses lots of operational, geographical and political lines. However, this solution is gaining ground with many leading suppliers of Internet routing equipment. In particular, Cisco has begun implementing this in their edge router models allowing network administrators to more effectively secure their networks and the Internet at large [Cisco Systems 2003].

Another thing to consider on the source side of the problem is individual machine security. The main reason these attacks are so powerful is the massive number of insecure nodes that litter the Internet, providing a zombie army whose might can be wielded at the stroke of a key. There is no solution to poor end-user security short of network-wide mandatory virus scanning, which is totally infeasible at this time.

#### 3.2 Destination-Side Solutions

There are also several solutions proposed to operate on the destination or victim-side of an attack. Most of these involve some sort of authentication protocol to filter out good traffic from bad.

One of them proposes to have a separate IP address to which valid traffic will be routed in the event of an attack [Xu & Lee, 2003]. It works by using an HTTP Redirect on all connected hosts pointing them to another server which is still up and running (and hopefully on a different subnet). Included in this redirect is a short Message Authentication Code (MAC) which allows valid users access to the desired resource. Since most attackers are using spoofed addresses, they will never receive the valid MAC and thus not be able to connect. The obvious problem with this solution is that it depends on valid traffic being able to connect at least once to the server under attack to get a valid MAC. This may be difficult while being bombarded with network traffic. Worse, if an attacker is able to spoof a valid address of a legitimately connected host, this scheme may be circumvented entirely.

Another destination-side method of mitigating the effects of these attacks involves firewall placement and state keeping for all open TCP connections [Chatam 2003]. It basically prescribes putting the firewall protecting the network on the fast side of the bottleneck and registering all valid connections. For example, consider an institution which connects to the Internet via a fractional T1. If the firewall is on the institution's end of the T1 then all traffic must reach it before being

filtered, making it very easy for attackers to clog the line. However if it is on the other side, with access to the traffic at a higher speed, it can make decisions to drop invalid traffic using normal firewall rules.

The purpose of logging TCP connections is, of course, to keep the target machine from being tied up by spoofed SYN packets, or SYN/ACKs for connections that were not requested in the first place. In this way it behaves much like TCP SYN cookies.

Of course the problem then becomes having ISPs host customers' firewalls; many ISPs would be averse to this type of solution because it places too much control of their internal network in the hands of its customers. Though, some middle ground could likely be achieved for this type of situation if a customer had enough clout with the provider.

Finally there is the simple, obvious, and expensive solution of resource multiplication, simply acquiring more bandwidth for key sites in order to allow uninterrupted communication. However, finite resources are exactly that: finite. And one cannot count on the fact that they have Gigabits of bandwidth to protect them from a widely distributed and well executed DDoS attack.

### 3.3 Network Solutions

Several solutions have also been proposed which would require large-scale, network-wide implementation to be effective. One of them proposes that each forwarding router on the network affix a portion of its own IP address to each packet it forwards [Yaar, Perrig & Song 2003]. This creates what amounts to a path signature on each packet, allowing a firewall or other network device to identify it as good or evil. Once one packet has been identified as malicious, all others can be summarily dropped.

One problem with this method is that it includes changing most Internet routers to accommodate it. If only a few implement the change, then path signatures will be incomplete, and some valid traffic would appear to be attack traffic. Another problem is the ever-growing size of packet headers—if at each hop the packet increases by 15 bits as prescribed, packets could get weighty after a few hops.

Another method involves detection of attacks within the network using traffic statistics, the idea being that early detection can lead to early resolution. An entropy value is assigned to a set of consecutive packets painting a picture of normal use. During an attack these

entropy values fall out of the prescribed range and traffic can be slowed down in-network with congestion reduction techniques. This type of solution can be effectively implemented on relatively few Internet routers because each one can provide some relief to victim machines [Feinstein et al. 2003]. Tuning the algorithm can be problematic, however. If the solution generated too many false positives it might cause more problems than it solves.

There are many similar solutions based on some sort of statistical attack detection scheme. Under the taxonomy established in [Mirkovic J. & Reiher, P. 2004] these sorts of solutions would be classified as reactive, in that they detect an attack in progress and then take action to prevent or mitigate the damaging effects of that attack.

### 3.4 Simulated Solution

We chose to take a closer look at Secure Overlay Services, a DoS strategy proposed by researchers at Columbia University [Keromytis, Misra & Rubenstein 2002]. Their solution not only provides protection of resources against flooding attacks, but recoverability in the event that a sophisticated attacker with detailed knowledge of the system and its architecture executes an attack against this system. This survivability is key in situations where just seconds of downtime can be disastrous.

This solution can not be easily pigeon-holed into the categories we outlined above. Certainly it is a network solution, because it involves several nodes of the network acting together to prevent/mitigate an attack, however it does not encompass the entire network. Also it has aspects of a destination-side solution because the participating network elements are working to protect a specific destination. The best way to categorize this solution is to use the system developed in [Mirkovic & Reiher 2004] and simply call it pro-active. This will become clearer in Section 4.

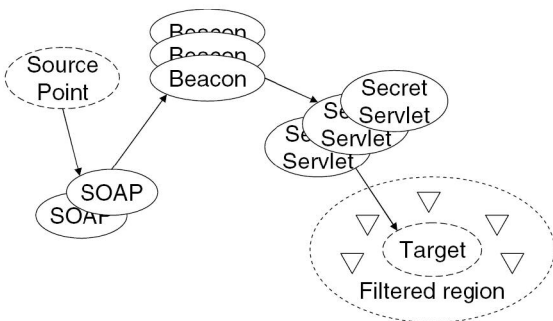
## 4. SECURE OVERLAY SERVICES

Secure Overlay Services were designed with disaster response in mind. In the aftermath of 9/11, the phone network was overwhelmed with family members searching for one another, first-responders communicating, etc. The design of SOS stems from this situation in which communication is essential and timely and when a well-timed DDoS attack could cripple our ability to respond to actions on the ground.

The SOS architecture was best described by the original authors.

The portion of the network immediately surrounding the target (location to be protected) aggressively filters and blocks all incoming packets whose source addresses are not “approved.” The small set of addresses (potentially as small as 1-3 addresses that are “approved” at any particular time is kept secret so that attackers cannot use them to pass through the filters. These addresses are picked from among those within a distributed set of nodes throughout the wide area network, that form a *secure overlay*: any transmissions that wish to traverse the overlay must first be validated at entry points of the overlay [Keromytis, Misra, & Rubenstein 2002].

The overlay itself consists of many nodes that can be widely distributed over a very large area. They communicate with each other using the Chord Peer-to-Peer lookup service which relies on consistent hashing to construct a virtual network on top of the existing network. Figure 1 below shows the communication architecture described by SOS.



**Figure 1 - The SOS Architecture**

When a target wishes to engage the services of the overlay network, it picks one or more Secret Servlets at random to act as proxies between the overlay and itself. Once this relationship has been established, the target employs very aggressive routing in its immediate vicinity, only allowing packets through which have the source address of this Secret Servlet.

By hashing the target’s IP address using a well known hash function, the Secret Servlet gets a Key which it uses to find and engage a Beacon in the Overlay. This same key can be calculated at the endpoints of the Overlay so that traffic entering it destined for the target can be forwarded to the same

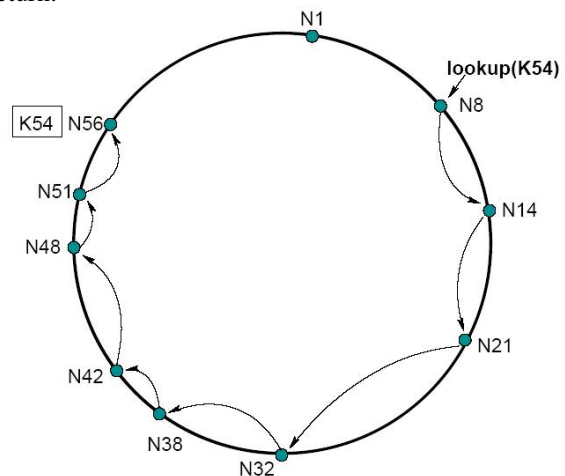
Beacon. A Beacon, having been instructed to send all traffic destined for the target through the Secret Servlet, will then forward traffic there. Once at the Secret Servlet, it need only forward it over the last step to the target. Employing more than one consistent hash function makes possible the use of more than one Beacon and thus further redundancy.

Users access the network via a Secure Overlay Access Point or SOAP. This SOAP provides authentication services before forwarding valid traffic into the overlay. Of course, once inside the overlay, traffic is forwarded as described above.

#### 4.1 The Chord Protocol

Once inside the overlay network, packets are routed using the Chord lookup protocol. Not only does it add an unpredictable element to the routing scheme, but it also maintains, with a high probability, a path length of  $O(\log N)$  nodes, where  $N$  is the number of nodes participating in the overlay [Stoica 2001].

Participating nodes in a Chord ring use a consistent hash function to hash their own IP addresses. The resulting  $M$  bit identifier, or Chord ID, orders the nodes in a ring as pictured in Figure 2. Each node in this ring knows the Chord ID and the address of his successor in the ring. Information is stored in key/value pairs. To store or find a value using a specific key, that key is hashed to produce an  $M$  bit key identifier. The node which succeeds this identifier in the Chord ring stores the associated value. Armed only with the knowledge of the IP address and Chord ID of its successor, any node can query using a key and receive a value in return.



**Figure 2 - A Chord Ring**

If Node 8 (N8) in Figure 2 wishes to retrieve the value associated with Key 54, it simply forwards the

request to its successor who forwards it to its successor, until Node 51 discovers that 54 is between itself and its successor, Node 56. This means that the requested value is stored in node 56.

Lookups can be greatly sped up with the addition of finger tables, which give nodes in the Ring access to nodes as far as half way around the ring. Figure 3 shows the table for Node 8 from the above example. Each node's finger table contains M entries corresponding to nodes succeeding  $k + 2^{i-1}$ , where  $k$  is the Chord ID of the node and  $i$  is the index into the table. When a lookup occurs, the node will find the closest preceding node in its finger table and forward the request to that node, cutting the trip effectively in half. These tables are what allow the Ring to provide  $O(\log N)$  lookup time.

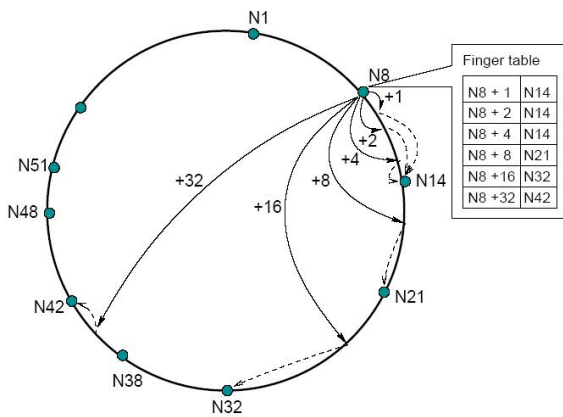


Figure 3 - Chord Ring with a Finger Table

In the event of a node failure, all keys that mapped to the failed node transfer to its successor. It is this feature of Chord that allows for built-in recoverability. This recoverability is based on maintenance. Each node must at all times be aware of its immediate successor for Chord to work, if that successor fails, the successor of that successor becomes the node's link to the rest of the network. To maintain that connectivity, a successor list of length R is built and maintained by each node. Also, a node's finger table must be maintained to ensure proper routing, in order to do this each node must regularly request the successor of  $k + 2^{i-1}$  and insert the value into its finger table. Finally each node must know the address and ID of its predecessor.

To put Chord in terms of SOS, the IP address of the target site is the key. That key is hashed to produce a key identifier in Chord. The result of a lookup on that key would be a Beacon. So when packets arrive in the

network at a SOAP, the SOAP hashes the IP address of the target and then forwards the packet on using the Chord protocol until it reaches a Beacon. The Beacon has stored the address of a Secret Servlet and forwards the packet there, etc.

## 5. EXPERIMENTAL DESIGN

We constructed a network simulation of an SOS network using the OPNET Modeler. In the following subsections we briefly describe the test network, the conditions on the network, and the situations we chose to simulate.

It is important to note that our simulated network concentrates exclusively on the performance of the SOS protocol, not the performance of any underlying protocols like TCP for end-to-end and IP for routing. The traffic we generate incorporates a simple stop and wait protocol to facilitate an easy measurement of time associated with a loss of communication. Also for the purposes of simulation, we represent an attack as a node failing. A node is said to be "under attack" when it stops making its own requests, responding to others' requests and forwarding data. When a node is no longer "under attack," it simply re-initializes and joins the overlay network. We believe that though these compromises represent a loss of fidelity, they do not impact the general findings we set out to gather. Section 8 describes our future work, which includes improvements to our models.

### 5.1 Simulation Topology and Functionality

Our simulated network consists of 24 Chord nodes, one target site and one user node. The target site enlists the aid of one Secret Servlet, and the Secret Servlet in turn registers itself with one Beacon. The user node connects to the overlay via a SOAP, which is also a member of the overlay, and directs requests to the target.

In order to maintain connectivity, each node must occasionally run the stabilization functions described in [Stoica, I., et al., 2001]: stabilize(), fix\_fingers() and check\_predecessor(). Because neither the Chord paper nor the SOS paper stipulates how often these maintenance functions should run, we arbitrarily chose a time frame of once every 10 seconds for each operation. This allows the Chord ring to fix itself upon failure in a maximum time of  $10 + \epsilon$  sec. where  $\epsilon$  is a timeout value for a stabilize() request. Because of the very small size of the network, the high speed of the links and the proximity of the nodes,  $\epsilon$  was set at 2 seconds.

To maintain the Overlay network, some additional maintenance is required. The site must ensure that its Secret Servlets are up and running, and the Secret Servlets must make sure that the Beacons are functioning. Also, if a Secret Servlet goes down, a Beacon must recognize this and stop sending traffic into the void. All of these features were implemented as a series of short messages and acknowledgements between nodes in our simulated network. These keep-alive protocols are described in [Keromytis, Misra & Rubenstein 2004].

## 5.2 Experiments

We performed a series of experiments on the network to determine its resilience to attacks of different sizes and sophistication. First we merely look at the case when an attacker knows which nodes are participating in the overlay, but does not know their individual nodes. Thus the best an attacker could do is attack nodes at random.

First, we assume that the attack will change volume and targets over time and that the attacker has the ability to fail a node with probability  $p$ . Each failed node will wait a random amount of time before coming back up and rejoining the overlay. Another attack possible with this knowledge is a full blown attack against a certain set of nodes  $n$ . Since these nodes are under constant attack, they can never rejoin the overlay, and SOS must heal around them.

Finally, we assume that attackers are sophisticated enough to know the most protected secret of the overlay: the identity of the secret servlet. With this knowledge and sufficient resources all traffic from the overlay to the protected site can be blocked by bringing the servlet down. We also assume that if the attacker could find this information once, she could find it again, so we simulate the failure of the secret servlet, followed by discovery of the new servlet followed by failure of the new servlet, etc.

## 6. RESULTS

The following graphs show the results of our simulation results. The only statistic we measure is the round trip time of each packet sent. If our network is functioning properly and not under attack, this value hovers around 0.07. However, when a packet times out, we set the value to 0.55 in order to make periods of failure easily noticeable in the graph. These first four tables depict an attacker randomly overloading nodes with probability  $p = \{0.25, 0.5, 0.75, 1.0\}$ .

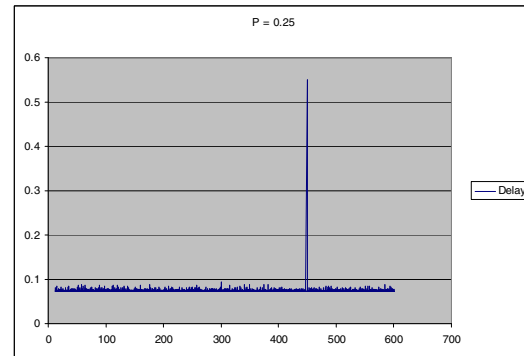


Figure 4 - Round trip time  $p=0.25$

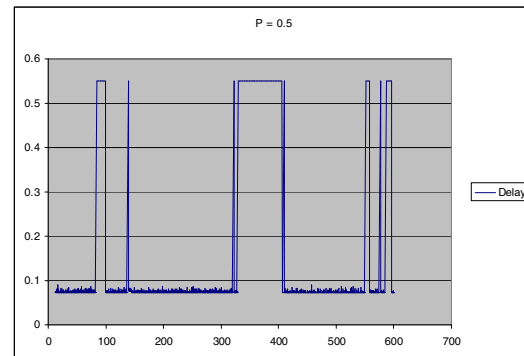


Figure 5 - RTT  $p = 0.5$

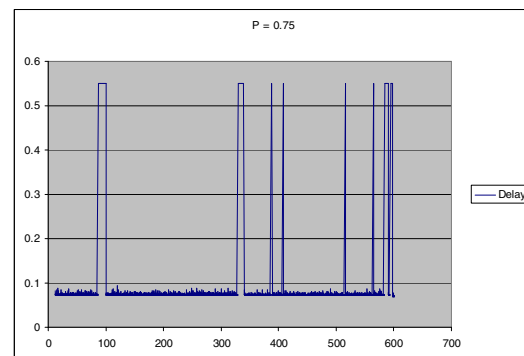


Figure 6 - RTT  $p=0.75$

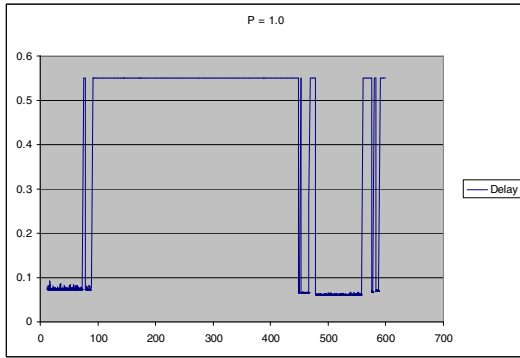


Figure 7 - RTT p=1

The next table shows the average recovery times for a full blown attack against  $n = \{3, 6, 9, 12, 15\}$  nodes in the network. Each attack was simulated 10 times to come up with the average recovery time listed in the table. For instances in which the network never recovered, a time of 400 seconds was recorded.

Table 1 - Recovery time for simultaneous failure of n nodes

n	Avg. Recovery Time
3 (12%)	4.226 sec.
6 (24%)	9.681 sec.
9 (36%)	12.681sec.
12 (48%)	56.09 sec.*
15 (60%)	145.03 sec.**

Finally, Figure 8 depicts the round trip times for traffic during the more sophisticated attack directed only at the secret servlets. Downtime for the most part is limited to less than two seconds, until the special case when the servlet was also the beacon (third peak in Figure 8). In that case it took more than 12 seconds to fully recover; however, this truly is a rare case.

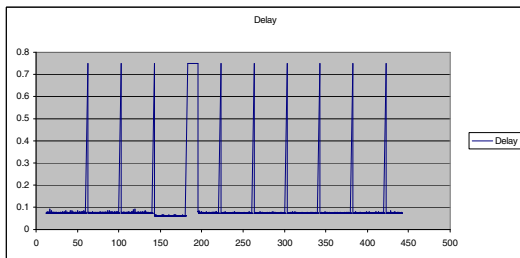


Figure 8 - RTT for 10 consecutive servlet attacks

\* Failed completely 10% of the time.

\*\* Failed completely 40% of the time.

## 7. CONCLUSIONS

With the attack tools that exist today, attackers need only query a network to find willing participants to join their zombie army and launch an attack with the push of a button. This means that defensive strategies must be proactive, that networks must endeavor to protect themselves. As a protection scheme for distributed denial of service attacks, SOS is resilient to catastrophic failures of a certain degree of severity. This property stems from the strength and flexibility of the underlying Chord lookup protocol which can correct itself in a relatively short amount of time in the event of node failure. To provide true protection, the network must be large, and large values must be chosen for R (successor list length) to insure against a total breakdown. Our data show preliminarily that that if implemented correctly, with a large enough network utilizing high-speed links SOS can provide adequate survivability in the face of a bandwidth DDoS attack.

## 8. FUTURE WORK

Our simulation merely brushes the surface of our performance analysis of SOS. We are now beginning to implement a TCP/IP stack and IPSec for more realistic traffic patterns in our simulation to provide higher fidelity and more relevant data. Following that, we plan to construct a full-scale implementation with all of the features described in both the Chord and SOS papers and field it over an actual WAN, or use a network emulator like dummy net [Rizzo 2005] to produce a similar network with a manageable number of machines.

Such an implementation could provide valuable data about the reasonable size of an SOS overlay, and about the amount of bandwidth an adversary would need to wield in order to effectively disable it.

## REFERENCES

CERT CC. "Smurf Attack." Online. Available [www.cert.org/advisories/CA-1998-01.html](http://www.cert.org/advisories/CA-1998-01.html). 2 Feb. 2005.

Chatam, J.W. *Using Strategic Firewall Placement to Mitigate the Effects of Distributed Denial of Service Attacks*, Masters Thesis. Auburn University, 2003.

Cisco Systems. *Transit Access Control Lists: Filtering at Your Edge*. 2003. Online. Available

- <http://www.cisco.com/warp/public/707/tacl.pdf>. 26 September 2004.
- Feinstein, L., Schnackenberg, D., Balupari, R., Kindred, D. DDoS tolerant networks. DARPA Information Survivability Conference and Exposition, 2003. Proceedings, Volume: 2, 22-24 April 2003 Pages: 73 - 75 vol.2.
- Ferguson, P. & Senie, D. Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing. Network Information Center, Request for Comments 2827, May 2000. RFC-2827. Online. Network Information Center. Available <http://rfc.net/rfc2827.html>. 26 September 2004.
- Garber, L. Denial-of-service attacks rip the internet. Computer, Vol.33, Iss.4, Apr 2000. Pages: 12-17.
- Gonzalez, Guadalupe. "Statement for the Record of Guadalupe Gonzalez, Special Agent in Charge, Phoenix Field Division, FBI on Cybercrime" before the Special Field Hearing, Senate Committee on Judiciary, Subcommittee on Technology, Terrorism, and Government Information, Washington, DC, April 21, 2000.
- Keromytis, A.D.; Misra, V.; Rubenstein, D. "SOS: Secure overlay services." Proceedings of ACM SIGCOMM, Pittsburg, PA, August 2002.
- Keromytis, A.D.; Misra, V.; Rubenstein, D. "SOS: an architecture for mitigating DDoS attacks." Selected Areas in Communications, IEEE Journal on, Vol.22, Iss.1, Jan. 2004. Pages: 176- 188.
- Mirkovic, J.; Reiher, P. "A taxonomy of DDoS attack and DDoS defense mechanisms." ACM SIGCOMM Computer Communication Review. Volume 34, Issue 2, April 2004. Pages: 39 – 53.
- Rizzo, Luigi. IP\_DUMMYNET. Online. Available [http://info.iet.unipi.it/~luigi/ip\\_dummy.net/](http://info.iet.unipi.it/~luigi/ip_dummy.net/). 2 February, 2005.
- Stoica, I., Morris M., Liben-Nowell D., Karger D., Kaashoek M., Dabek F., Balakrishnan H. "Chord: A scalable peer-to-peer lookup service for internet applications." Proceedings of ACM SIGCOMM, San Diego, CA, August 2001.
- Xu, J. & Lee, W. Sustaining availability of Web services under distributed denial of service attacks. Computers, IEEE Transactions on, Vol.52, Iss. 2, Feb. 2003. Pages: 195- 208
- Xuan, D.; Sriram Chellappan; Wang, X.; Wang, S. "Analyzing the secure overlay services architecture under intelligent DDoS attacks." Distributed Computing Systems, 2004. Proceedings. 24th International Conference on, Vol., Iss., 24-26, March 2004. Pages: 408- 417
- Yaar, A.; Perrig, A.; Song, D. Pi: a path identification mechanism to defend against DDoS attacks. Security and Privacy, 2003. Proceedings. 2003 Symposium on, Vol., Iss., 11-14 May 2003. Pages: 93- 107.