Countering DoS Attacks With Stateless Multipath Overlays*

Angelos Stavrou
Department of Computer Science
Columbia University
angel@cs.columbia.edu

Angelos D. Keromytis
Department of Computer Science
Columbia University
angelos@cs.columbia.edu

ABSTRACT

Indirection-based overlay networks (IONs) are a promising approach for countering distributed denial of service (DDoS) attacks. Such mechanisms are based on the assumption that attackers will attack a fixed and bounded set of overlay nodes causing service disruption to a small fraction of the users. In addition, attackers cannot eavesdrop on links inside the network or otherwise gain information that can help them focus their attacks on overlay nodes that are critical for specific communication flows. We develop an analytical model and a new class of attacks that considers both simple and advanced adversaries. We show that the impact of these simple attacks on IONs can severely disrupt communications.

We propose a *stateless spread-spectrum paradigm* to create per-packet path diversity between each pair of end-nodes using a modified ION access protocol. Our system protects end-to-end communications from DoS attacks without sacrificing strong client authentication or allowing an attacker with partial connectivity information to repeatedly disrupt communications. Through analysis, we show that an Akamai-sized overlay can withstand attacks involving over 1.3M "zombie" hosts while providing uninterrupted end-to-end connectivity. By using packet replication, the system can resist attacks that render up to 40% of the nodes inoperable. Surprisingly, our experiments on PlanetLab demonstrate that in many cases end-to-end latency *decreases* when packet replication is used, with a worst-case increase by a factor of 2.5. Similarly, our system imposes less than 15% performance degradation in the end-to-end throughput, even when subjected to a large DDoS attack.

Categories and Subject Descriptors

C.2.0 [**Security and Protection**]: Denial of Service; C.2.1 [**Network Topology**]: Overlay Networks

General Terms

Security, Reliability.

Keywords

Spread-spectrum communications, key agreement.

*This work was supported by the National Science Foundation under grant ITR CNS-0426623.

CCS'05, November 7–11, 2005, Alexandria, Virginia, USA.

1. INTRODUCTION

Solving the network denial of service (DoS) problem is extremely hard, given the fundamentally open nature of the Internet and the apparent reluctance of router vendors and network operators to deploy and operate new, potentially complex mechanisms. Overlay-based approaches such as SOS [13] and MayDay [1] offer an attractive alternative, as they do not require changes to protocols and routers, and need only minimal collaboration from Internet Service Providers (ISPs). Such systems use an Internet-wide network of nodes that act as first-level firewalls, discriminating between legitimate traffic and potentially malicious traffic, based on some form of user or end-host authentication. Their distributed nature requires an extremely well provisioned adversary to suppress their functionality, since attack traffic must be split among all the nodes to disrupt protected communications.

Indirection-based overlay network (ION) approaches depend on the inability of an adversary to discover connectivity information for a given client and the infrastructure (*e.g.,* which overlay node a client is using to route traffic). This makes them susceptible to a variety of easy-to-launch attacks that are not considered in the standard threat model of such systems. For example, adversaries may possess real-time knowledge of the specific overlay node(s) a client is routing traffic through, or may be attacking nodes using a time-based scheme that will try to maximize the impact of the attack on clients' connectivity. Such attacks can be network-oriented (*e.g.,* TCP SYN attacks) or application-related "sweeping" attacks or "targeted" attacks.

In targeted attacks, an attacker that has knowledge of the client's communication parameters can "follow" the client connections and bring down the nodes that he tries to connect to. As soon as the client realizes (typically after some timeout period) that the overlay node is unresponsive and switches to a new node, the attacker also switches the attack to this new node. Thus, an attacker that can bring down a single node can create a targeted DoS for specific clients. Similar attacks, exploiting information that must only be available to trusted components of the system but which an attacker can feasibly gain access to, are possible against almost all proposed anti-DDoS mechanisms [24, 4, 11].

In sweeping attacks, the attacker uses its power (which is insufficient to bring down the whole ION) to attack a small percentage of the overlay nodes at a time. This type of attack targets the application-level state maintained by the overlay node responsible for a client. Destroying this state forces the client to re-establish both network and application-level connectivity, degrading the clients' connection and leading to DoS for time-critical or latency-dependent applications. Thus, although IONs can counter blind DoS attacks, they remain vulnerable to a range of simple but debilitating attacks.

1.1 Our Approach

We believe that these *inherent* limitations of first generation overlay-based traffic redirection mechanisms can be addressed by adopting a spread-spectrum like communication paradigm[1]. In a "spread-spectrum" approach, the client spreads its packets randomly across all access points, preventing an attack from "following". The path diversity naturally exhibited by a distributed overlay network serves as the "spectrum" over which communications are "spread." In our system, a token issued by the overlay network to the client is used to verify the authenticity of each packet communicated by the client. The use of a token (akin to a Kerberos ticket) alleviates the necessity to maintain application or network-level state at any of the overlay nodes (unlike previous IONs), at the expense of bandwidth (since the ticket must be included in every packet routed through the ION). In return, our system is impervious to the attacks that use this state dependence to attack the overlay.

The main challenges we must address relate to the scheme's efficiency (in terms of performance and latency of the end-to-end path), resiliency to attacks, amount of state that needs to be maintained by each overlay node (necessary to prevent packet replay or forging attacks), and the elimination of communication pinch points on which attackers can focus their attention.

We argue that such a system is feasible, and describe our specific approach and its implementation in Sections 2 and 3, respectively. For an attacker to successfully attack our system, he will have to subvert or suppress 40% or more of the overlay nodes before the system becomes unusable for all users. Thus, our system has an operational threshold in the order of 40% of the nodes being subverted. Before this 40% threshold is reached, the users will not notice a significant impact to their connectivity. As a comparison, in the original SOS architecture, the user had to find an access point that was not under attack, which becomes increasingly difficult as we increase the portion of nodes under attack. We quantify the increase in the system's resistance to attacks using a simple analytical model, and provide experimental validation by deploying a prototype over PlanetLab, a wide-area overlay network testbed. PlanetLab nodes are distributed across the Internet, serving as an ideal platform for experimentation.

Our analysis shows that an Akamai-sized ION ($\sim$ 2500 nodes) can withstand attacks that bring down up to 40% of the overlay. This corresponds to attacks that involve several million "zombie" (attacking) hosts, which is an order of magnitude larger than the biggest zombie network seen to date. One expects that using an ION will impose a performance penalty. In our case, end-to-end latency increases by a factor of 2 in the worst case, but by using packet replication we maintain latency at the same level as the direct connection case. These results confirm the findings from other research on multipath routing [10, 3, 2]. Furthermore, end-to-end throughput is not significantly degraded, with an overhead of less than 15% relative to the direct-connection case.

1.2 Contributions

The contributions of our work are:

- We introduce a realistic threat model against IONs, in which opponents can use their limited attack capabilities against a time-changing set of overlay nodes. In addition, we consider more sophisticated attackers with access to information that can be used for targeted and/or adaptive attacks against the protection mechanisms themselves.
- Second, we present an architecture for an overlay-based anti-DDoS mechanism that is resistant to DDoS attacks in the new threat model, by using stateless tokens and traffic spreading.
- Third, we provide a first attempt at an analytical model for quantifying security in overlay-based DDoS protection mechanisms.
- Fourth, we demonstrate the feasibility of our approach in a realistic set of experiments over the Internet using PlanetLab.
- Finally, we show that the overhead of our overlay-based mechanism on end-to-end latency is close to zero in several usage scenarios, including real-time traffic, which is acceptable even for time-critical applications.

[1]Note that although we use the term "spread-spectrum" to describe our approach, our work is *not* geared towards wireless networks, nor does it touch on physical-layer issues.

Paper Organization Section 2 describes our system architecture using a spread-spectrum-like paradigm. The system design and implementation are thoroughly explained in Section 3. Section 4 gives our evaluation of the system, in terms of the improvement in resistance to attacks. Section 5 experimentally evaluates the performance and attack resilience characteristics of our approach. The paper ends with a discussion of related work and conclusions.

2. SYSTEM ARCHITECTURE

We begin by giving an overview of how indirection-based mechanisms operate and describing the security issues present in the current generation. We then describe our approach, which uses stateless multipath overlay routing to send each packet through a randomly selected overlay node. The main components of our design are (a) a stateless protocol for authenticating users to the infrastructure such that they are not vulnerable to "step zero" attacks (DoS attacks that prevent them from contacting the overlay), and (b) an efficient per-packet authentication scheme that allows the system to scale to millions of users.

2.1 Overlay Protection Mechanisms & Attack Model

The goal in combating DoS attacks is to distinguish between authorized and unauthorized traffic; the former is allowed to reach the destination, while the latter is dropped or is rate-limited. Thus, at a very basic level, we need the functionality of a firewall "deep" enough in the network that the access link to the target does not become congested. This imaginary firewall performs access control by using protocols such as IPsec or TLS. Traffic is then routed to a secret location, which may be the server itself or a node that is allowed to contact the server (called "secret servlet" in SOS [13]), with all other traffic being filtered. The reason for having a small number of secret servlets is to minimize the number of filtering rules, as they can affect router performance. The secret servlet may vary over time, and may differ for each protected site.

Most such systems concern themselves with *naive attackers*, *i.e.,* those without internal knowledge of the system (other than the list of participating nodes). We assume that such an attacker can mount a DoS attack against a small set of nodes in the overlay for short periods of time, which will force clients using those overlay nodes to reset their connections to new nodes. This attacker blindly "sweeps" all the nodes participating in overlay network focusing the attack from one set of overlay nodes to another, selecting nodes not previously attacked. Presently, a number of DoS attacks can be used as "sweeping" attacks: TCP SYN, ICMP flooding and TCP congestion attacks are among them. If the overhead of detecting the failure and switching to a new access point is high, compared to how long an attack must be sustained to force the connection reset, an attacker can cause significant disruption in the communications. Performance can be seriously degraded, and long-lived connections (such as a teleconference or a large file transfer) can be repeatedly disrupted, rendering them ineffective as communication carriers. This attack is similar to a radio jammer that is randomly broadcasting noise in various channels, forcing communicating parties to continuously reset their network parameters.

Although less efficient against a single user compared to a targeted attack, this attack can be more effective, degrading the connection characteristics or preventing connectivity on most of the clients connected to the overlay. The success of the attack depends on factors such as the attack intensity[2], and the time required to detect the connection failure and then find a new overlay node that is healthy and re-establish both network connection and client authentication credentials (usually on the application level). Moreover, the client's authentication can be complex, *e.g.,* using X.509 certificates for authentication or Graphic Turing Tests [21] to allow anonymous human users. Most such authentication mechanisms require time and user interaction, which make these sweeping attacks a serious problem for real-world deployed overlays.

A more sophisticated attacker, explicitly not considered in other proposed IONs, may know which overlay node a client is using. An attacker can get this information by eavesdropping on an appropriate edge-network link: the client's wireless communications to his access point or the link to his ISP. Such an attacker can "follow" the client and direct DoS traffic against the overlay nodes that he tries to communicate with. The client, detecting a failure in communications, will select another node to access the overlay, which will become the attacker's new target. Using the radio communications analogy, this is akin to an adversary that is eavesdropping on wireless communications, jamming frequencies where traffic is detected; after a short period of time, the adversary searches for new frequencies the attacked parties may have switched to. [1] identifies possible ways an adversary can gain such information; other possibilities include snooping on the local network link, *e.g.,* in a wide-area wireless network such as the upcoming WiMAX, or in some enterprise-wide 802.11 (WiFi) environments.

This threat model is considerably stronger than the typical scenarios anti-DDoS mechanism designers have considered in the past. We can address all of the above attacks by employing a nearly-stateless spread-spectrum communication paradigm in conjunction with an overlay network.

2.2 Traffic Spreading

The first problem we address is how to protect the communications of a client of the overlay from attackers that either have partial knowledge of the communication parameters (*i.e.,* can determine which overlay nodes a client is communicating with), or are blindly attacking overlay nodes using "sweeping" attacks, thus forcing clients to keep re-establishing connections to new overlay nodes. For simplicity, we temporarily assume that the reverse channel (from the overlay to the client) is protected by the overlay in the same manner communications to the server are protected, or is otherwise safe from interference.

Our approach, shown in Figure 1, is straightforward: spread the packets from the client across all overlay nodes in a pseudo-random manner storing no network or application level state in the overlay nodes. An attacker will not know which nodes to direct an attack to; randomly attacking a subset of them will only cause a fraction of the client's traffic to be dropped. By using forward error correction (FEC) or simply duplicating packets (*i.e.,* sending the same packet through two or more different access points simultaneously), we can guarantee packet delivery with high probability, if we place an upper bound on the number of nodes an attacker can simultaneously attack. We quantify this increase in attack resistance in Section 4. In designing our system, we must address several issues:

[2] In this context, attack intensity is the percentage of overlay nodes that can be brought down simultaneously by the attacker.

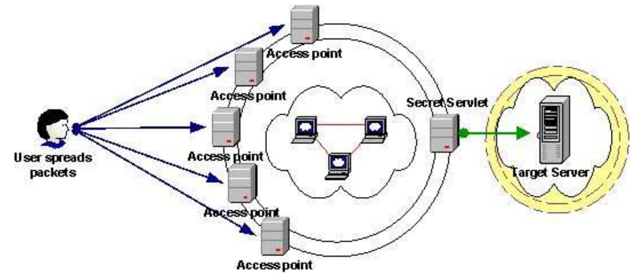

Figure 1: Spreading traffic across multiple overlay access points. Attacks that render a number of overlay nodes ineffective do not impact end-to-end communications.

- First, it should not be possible for an attacker to impersonate a legitimate user and conduct a DoS attack through the overlay. This means that each packet from the user to the overlay must be properly authenticated.
- The second issue we must address is the state that each overlay node must maintain per client: since all overlay nodes can potentially receive traffic from all users, the memory requirements can quickly become prohibitive. Furthermore, a client's end-to-end connection must not depend on the network availability of a small set of overlay nodes. Keeping state that is essential for a client's network or application level connectivity makes the system vulnerable to sweeping or targeted attacks.
- Third, even legitimate clients should not be allowed to "pump" unlimited amounts of data through the overlay; verifying this is complicated due to the packet-spreading approach.
- Finally, the selection of the overlay node to forward a packet through should be as random as possible from the point of view of an external observer (*i.e.,* an attacker), yet verifiable by individual nodes, to avoid flooding attacks by compromised clients.

In the remainder of this section we describe two protocols: one used to establish a restricted ticket and secret session-key between a client and the overlay, and a second protocol used as a stateless communication protocol that allows overlay nodes to verify the validity of received packets without requiring maintenance of large amounts of state.

2.3 Key and Ticket Establishment Protocol

To achieve a stateless communication with the overlay network, a client has to acquire a ticket, which is then included in all subsequent packets sent through the overlay. As we will see in detail in the next section, the ticket is used by the overlay nodes to authenticate the user, validate the routing decisions, and prevent malicious (or subverted) nodes from utilizing a disproportionate amount of bandwidth. Thus, node authentication and ticket acquisition/maintenance is a key component of our approach. Although any authentication protocol could be used, most such protocols require at least two round-trips between the two parties (as well as

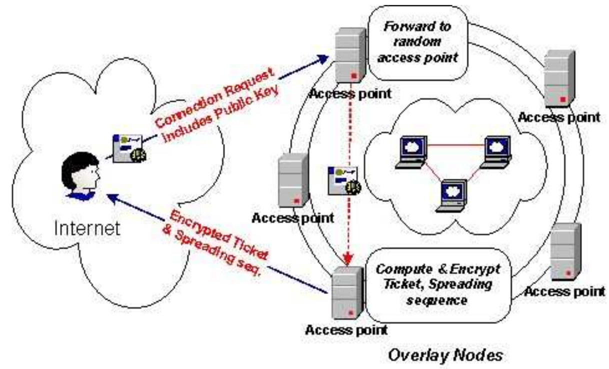

Figure 2: Redirection-based authentication and key establishment. An attacker observing the interactions of a user and the overlay cannot determine which overlay node(s) to target.

considerable computation). However, an attacker that is observing communications between the client and the overlay can direct a congestion-based DoS attack[3] against any overlay node that is contacted by the client for authentication purposes. Since the client does not yet have a spreading sequence, it seems at first impossible to protect the key establishment phase.

Our proposed approach is to randomly redirect the authentication request, as shown in Figure 2. Briefly, the client selects an overlay node at random and sends a packet containing its public key certificate and a request to initiate authentication. The receiving node immediately forwards the request to another overlay node at random; thus, an attacker (who cannot react fast enough to prevent a packet from being forwarded on) does not have a target.

The second overlay node selects a random session key K_u and creates a ticket for that client. The ticket contains K_u, a range of packet sequence numbers for which K_u and the ticket are valid, a randomly selected identifier for the client, the current time-stamp, and flags indicating that this is a "restricted" ticket (more on this later), all encrypted and authenticated under K_M, a secret key negotiated periodically (*e.g.,* every few hours) among all overlay nodes (see Figure 3). The last part of the ticket is a UMAC [6] signature of the encrypted ticket using K_M and a 64-bit nonce, which consists of the first 64 bits of the encrypted ticket. Note that only overlay nodes can validate and decrypt the ticket. The client's certificate is validated, and a second copy of K_u is independently encrypted under the client's public key. Both operations are relatively lightweight (compared to operations involving RSA private keys); as was shown in [14], a node can perform a few thousand public-key operations (*i.e.,* signature verifications or public-key encryptions) per second. The ticket and the encrypted session key are then sent to the client. An extra, optional message can be sent from the overlay to the client with the list of overlay nodes' IP addresses. This one-round-trip protocol is stateless (for the overlay) and computationally fast, resisting both memory and CPU exhaustion attacks on the overlay nodes.

To make it even more difficult for the attacker to mount a CPU exhaustion or IP spoofing attack, we can add one more round-trip on the key establishment protocol, forcing the client to send a UMAC-signed certificate before generating the ticket (which requires validation of the client certificate). Figure 4 displays both the one round-trip and the two round-trip key establishment protocol in detail. In the two-round-trip protocol, the client sends his certificate to overlay node A. A redirects the request to B, a randomly selected overlay node. B treats the certificate as a random number, which he UMAC-signs with the shared key K_M. The client's IP address and the system's timestamp are the nonce used in the UMAC operation. B sends the UMAC signature and the nonce to the client. To prove liveness, the client contacts another randomly selected overlay node, C, sending its certificate, the UMAC signature and the nonce. C validates the authenticity of the UMAC and redirects the request to D, another randomly selected overlay node. Finally, D generates a ticket for the client, encrypting it with the client's public key (retrieved from the certificate). In the two-round protocol, only the last step is computationally expensive (compared to simple UMAC verification). Thus, the two-round-trip protocol, guarantees client liveness. For the one-round-trip protocol we only use the first and the last communication *i.e.,* from A to D as shown in Figure 4. Finally, if there is a version mismatch between the list of overlay nodes' IP addresses stored locally in the client (communicated by the client in the first message) and the one stored in the overlay network, a random overlay node, E, is chosen by D to send the list differences to the client.

[3] Computational DoS attacks can be partially mitigated using proof-of-work techniques [12, 7].

2.4 Client Authentication

The ticket obtained from the previous protocol can only be used by the client to continue the authentication protocol (*i.e.,* prove liveness for both the overlay and the client. Once two-party authentication is completed, the last overlay node provides the client with a ticket that is not "restricted," *i.e.,* the corresponding flag inside the ticket is cleared. The tickets are periodically refreshed, to avoid situations where a malicious user distributes the session key and ticket to a large number of zombies that try to access the overlay.

This authentication step can be followed by a secondary authentication phase that uses a Graphic Turing Test (GTT) [21] to discern the presence of a human at the client node (versus a remotely controlled DDoS zombie). This step can prevent legitimate nodes that have been subverted by an attacker from being used as entry points to the overlay, but can only be used for those applications that have a GUI — such as a web browser. We can implement the secondary GTT-based authentication by issuing a second restricted ticket after the completion of the two-phase authentication step (from above), which only allows client nodes to contact the GTT server. This server is implemented locally by each overlay node, as was shown in [18]. Once the GTT step is successfully performed, the GTT server issues an unrestricted ticket to the client node. The GTT authentication can be performed periodically (to confirm the continued presence of a human). Naturally, this step is not applicable for applications where there is no human being directly controlling the client, or where displaying a graphic is infeasible or impractical (or for vision-impaired persons).

2.5 Client-Overlay Communication Protocol

Once the client has received a session key and an unrestricted ticket, he may start sending packets to the remote destination through the overlay. Each packet sent by a client to an overlay node contains three overlay-related fields: the ticket, an authenticator, and a monotonically increasing sequence number, as shown in Figure 3. The ticket contains the session key and a sequence range for which the ticket is valid, as we discussed previously, and is encrypted and authenticated under a secret key K_M, known to all overlay nodes. Note that these overlay nodes are *not* user machines, but are hosts dedicated to offering a DoS protection service.

The sequence number is a 32-bit value that is incremented by the client for each packet transmitted through the overlay with a given session key. The client identifier is a random 32-bit value that is se-

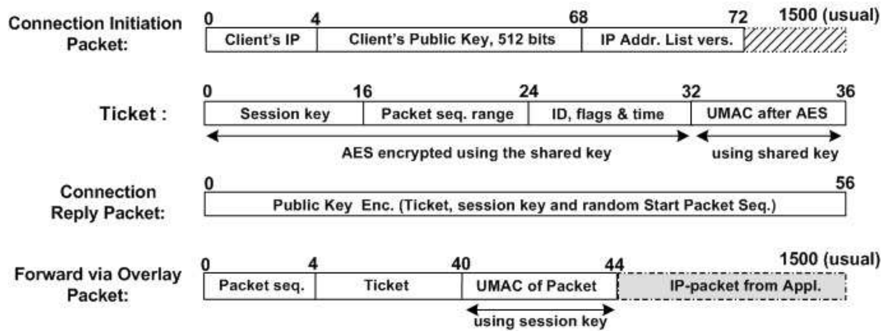

Figure 3: **The layout of the various packets and the ticket used to establish a communication and transmit packets between the client and overlay nodes. All numbers are in bytes, unless otherwise indicated.**

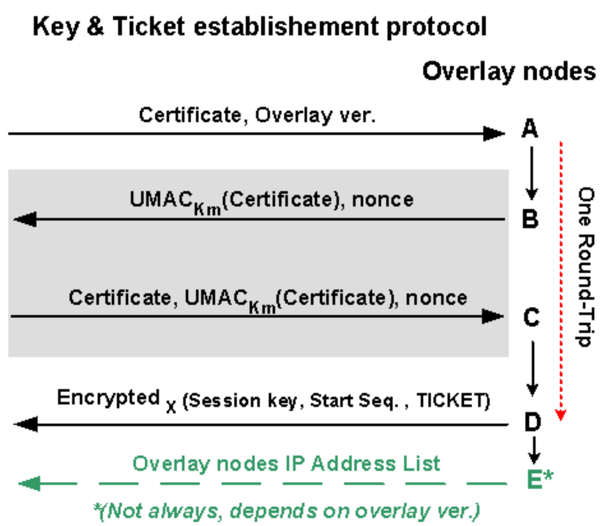

Figure 4: **Key & Ticket Establishment protocol: The client sends node A his certificate. A immediately redirects the request to B, in the two-round-trip protocol, or to D for the one-round-trip protocol. The four-message protocol is more resilient against computational attacks since it ensures the client's liveness before generating an encrypted version of the ticket. A 5^{th} message is transmitted when the client's version for the list overlay nodes is old.**

lected by the overlay node that authenticated the client, and is used as an index in the table of last-seen sequence numbers per client, maintained by each overlay node. The authenticator is a message authentication code (MAC) using a fast transform such as UMAC and the session key K_u. The UMAC is computed over the whole packet, which includes the ticket and the sequence number of the packet. For the UMAC nonce we use the sequence number concatenated with the client's IP address. Thus, the ticket is bound to a specific IP address and cannot be distributed to other clients. The only state each overlay node needs to maintain per client consists of the client identifier and the last sequence number seen by that particular client. This state is not network or application related and is used solely to prevent "replay" attacks. Assuming that both the client identifier and the sequence number are 32-bit values, each overlay node needs to maintain only 64 bits of state for each client; thus, if the overlay could support 1 million active clients (in terms of network capacity), we will only need 8 MB of state.

A client transmitting a packet through the overlay uses the session key and the sequence number as inputs to a pseudo-random function (PRF). The output is treated as an index to the list of overlay nodes, through which the packet will be routed. The list of available overlay nodes does not need to change frequently, even if nodes become unavailable (*e.g.,* for maintenance purposes). There are various ways a client can obtain the list of overlay nodes. For example, it can be done the first time it connects to the overlay network by requesting it after the key establishment phase, or by downloading it independently of the protected communication. After the first time, the client can maintain the freshness of the list by comparing the version of his list with the one stored in the overlay, downloading only the differences of the two versions.

The client then encapsulates the original packet (addressed to the final destination) inside a packet for the overlay node, along with the information identified above (ticket, sequence number, authenticator). This packet is forwarded through the overlay to the appropriate secret servlet, and from there to the final destination.

Upon reception of a packet, the overlay node checks the validity of the ticket. This is a UMAC validation, a fast operation preventing computational DoS attacks against the overlay nodes. After validating the authenticity of the ticket, the ticket is decrypted and the authenticator is verified. This prevents spoofing attacks from an adversary who obtains a valid ticket and generates packets to all overlay nodes with randomly selected sequence numbers, thus preventing the client with the valid ticket to communicate. Furthermore, to detect any replay attacks, an overlay node that receives such a packet verifies that the sequence number on the packet is larger than the last sequence number seen from that client by using the client identifier to index the internal table. The overlay node also verifies that the sequence number is within the acceptable range of sequence numbers for this ticket. Finally, it uses the key and the sequence number along with the PRF to determine whether the client correctly routed the traffic. If all steps are successful, the overlay node updates the sequence number table and forwards the packet to the secret servlet. Packets with lower or equal sequence numbers are considered duplicates (either accidental reordering or malicious replays by attackers) and are quietly dropped.

To avoid reuse of the same ticket by multiple DDoS zombies, the range of valid sequence numbers for the ticket is kept relatively small (and contained inside the ticket), *e.g.,* 500 packets. Moreover, the ticket is bound to the client's IP, since to authenticate the packet the overlay uses the UMAC including the client's IP address as part of the UMAC nonce. In addition, each packet contains a timestamp with which we can validate the freshness of the ticket. After a configurable period of time (*e.g.,* 1 or 2 hours) the overlay expires the ticket. Overlay nodes that receive valid tickets about to expire simply re-issue a new ticket with the same session key but a new range of valid sequence numbers. This approach, combined with the state kept by each node, makes it prohibitive for attackers to reuse the same ticket from a large number of distinct nodes (each of which is only transmitting to a specific overlay node), since the new valid ticket needs to be continuously propagated to all zombies.

The shared key under which the ticket is encrypted is periodically established among all overlay nodes, using a group key management protocol. The precise properties of this protocol are not relevant to this discussion, and there exist a large number of such protocols in the research literature.

3. IMPLEMENTATION

The implementation consists of the code for the overlay nodes, as well as code running on each client that does the encapsulation and initial routing. On the client, a routing-table entry redirects all IP packets destined for the protected servers to a virtual interface, implemented using the *tun* pseudo-device driver. This device acts as a virtual network interface intercepting messages to and from a real network interface. IP packets sent to the *tun0* network interface can be read by a user process reading the device */dev/tun0*. Similarly, if the process writes a complete IP packet to */dev/tun0* this will appear in the kernel's IP input queue as if it were coming from the network interface *tun0*. Thus, whenever an application on the client tries to access a protected server, all outgoing traffic is intercepted by the virtual interface. A user-level proxy daemon process reading from the corresponding device captures each outgoing IP packet, encapsulates it in a UDP packet along with authentication information, and sends it to one of the overlay nodes according to the protocol. The code running on overlay nodes receives these UDP packets, authenticates and forwards them to the secret servlet, which forwards them to the final destination. There, the packets are decapsulated and delivered to the original intended recipient (*e.g.,* web server). The decapsulation can be done by a separate box or by the end-server itself. In addition to the decapsulation code on the overlay nodes, there is also a daemon listening for connection establishment packets from the clients.

Connection Establishment Phase: When a client attempts to contact the protected server for the first time, it receives a small list of randomly selected overlay nodes' IP addresses via regular DNS name resolution. It selects one of them and transmits a "connection initiation" packet (shown in Figure 3) to authenticate itself, acquire a ticket and a session key, and to update its list of overlay nodes. Thus, for the very first IP packet that the proxy daemon on the client's host receives for a previously unknown server, it constructs a connection initiation which it sends to a randomly selected overlay node. The connection request is a UDP packet to a well-known port. It contains the version number of the list of overlay nodes' IP addresses stored locally, if any, along with its public key P_{cl}, as shown in Figure 3. When an overlay node receives such a request, it forwards it to another node at random.

This second overlay node generates a 288-bit (36 bytes) ticket. The first 224 bits of the ticket consist of a 128-bit session key K_u, a 64-bit packet sequence range for which the ticket is valid with the starting sequence randomly selected, and a 64-bit field with the Client ID, time-stamp and flags that is used also to avoid public-key dictionary attacks; this part is AES-encrypted using a Master Key K_M shared among overlay nodes. A 32-bit UMAC authenticator is appended, computed over all fields in the ticket using again the master key K_M. The ticket, the starting packet sequence and the session key K_u (encrypted under the client's public key) are sent to the client. Another optional packet containing differences of the current list of the IP addresses of all overlay nodes is also sent to the client, depending on the version indicated in the connection initiation packet.

Packet Transmission Phase: After receiving a session key and ticket, the client constructs a "forward request" UDP packet containing the packet sequence number, the ticket, and the original IP, as shown in Figure 3. It then determines which overlay node to send the packet to by using the session key, the packet sequence (start sequence plus one for the first packet) and the publicly available sorted list of IP addresses of the overlay nodes. Assuming that the number of overlay nodes is n, the client computes the index in the sorted list of IPs as:

$$index = UMAC(K_u \oplus \text{sequence number}) mod(n)$$

The receiving overlay node validates the ticket using the ticket UMAC. Then the ticket is decrypted using K_m and the packet authenticity is verified. The sequence number is compared against the one stored in the overlay node for this client identifier, if there is one (otherwise, this is assumed to be a packet from a new client). If the sequence number on the packet is bigger, the overlay node stores the new sequence number and checks if the ticket is expired (*i.e.,* packet sequence > max packet sequence), after decrypting the ticket. Then, computing the index as above, it checks whether the packet was correctly routed to this node. If any of the checks fails, the packet is dropped. Otherwise, the packet is routed to the secret servlet, and from there to the actual server.

Ticket Renewal Phase: During the packet transmission phase, overlay nodes may receive requests using valid tickets that are about to expire. In that case, the overlay node issues a new ticket with the same session key but larger max sequence number, and sends the client a connection-request reply packet containing the new ticket.

4. QUANTIFYING ATTACK RESISTANCE

We now evaluate the security of our scheme using a simple analytical model, which we apply to first-generation IONs that are vulnerable to targeted or sweeping attacks. We then quantify the attack resistance generally offered by IONs using a simple model of an ISP and typical POP speeds. In the next section we will characterize the impact of our system on latency and throughput in a series of experiments over the Internet using PlanetLab.

4.1 Impact of Sweeping Attacks

First-generation IONs were geared towards service connection availability. No provision is made for attacks that cause the user to reset his connection, either because the overlay node is unresponsive or because the connection quality is low. After resetting the connection, the user has to re-establish connectivity and re-authenticate himself, making the system unrealistic for real-time applications. Moreover, frequently forcing the user to re-authenticate through a challenge-response or a CAPTCHA will render the system unusable for any type of application.

We assume that an attacker can mount a DoS attack against a small set of nodes in the overlay for short periods of time, which (in first-generation IONs) will force clients using those overlay nodes to reset their connections to new nodes. This attacker blindly sweeps all the nodes participating in the overlay network, focusing his attack from one set of overlay nodes to another, keeping the sets disjoint. Not all of these attacks can be easily detected by the current infrastructure: an attacker can mount a low-rate TCP attack [16] reducing the effective bandwidth of the victim to zero. Thus, a sweeping attacker can cause significant disruption in the end-to-end communication.

To analyze a sweeping attack and quantify its impact to the clients' connection characteristics in first-generation IONs, we create a simple static model. We assume that the attacker can bring down p_d

percentage of the overlay nodes simultaneously. For an attack to be successful on these nodes, it needs t_a time of sustained attack. This is the time required to either drop or severely rate-limit the connections of all the clients connected to nodes under attack. Let t_u be the average time a client is connected to the system. Also, let t_d be the time that is necessary for the client to detect the attack and connect to another overlay node. Moreover, we assume that the overlay repairs the nodes under attack immediately after the attack focus has shifted to another set of nodes (zero reboot or repair time) so the time to repair $t_r = 0$. Finally, we assume that clients are connected uniformly across all overlay nodes in a first-generation ION, *i.e.,* if there are N clients and O overlay nodes, each has $\frac{N}{O}$ clients. The percentage of clients that will have their connection reset by a sweeping attack at least once during the time that they use the system is $P_1(t_u, t_a, p_d) = \dfrac{t_u}{t_a} \cdot p_d$ assuming $t_{det} = 0$.

The above formula is very intuitive: from the attacker's perspective, there are $\frac{1}{p_d}$ disjoint sets of nodes in the overlay network. To attack all of them the attacker needs $\frac{t_a}{p_d}$ time. Assuming a system where we have no joins, an attacker will affect the connectivity of $\frac{t_u}{t_a} \cdot p_d$ clients. Note that some of the clients may never experience the attack because they might have finished their connection by the time the attack reaches them. For this simple model, we have assumed that there is no detection time: the user selects another overlay node to connect to as soon as the attack starts to affect him. Even with this very conservative model ($t_d = 0$, $t_r = 0$, no client arrivals while the system is under attack) we can see that the attack can be significant, depending on the usage time, the size of the attack compared to the size of the overlay and the time required for an attack to be successful. For example, assuming that we have clients with average usage time of an hour, an adversary that can attack 2.5% of the overlay nodes and shifts the attack every 5 minutes will affect 30% of the clients. We can also compute the percentage of nodes that will have to reset their connections more than once. The percentage of nodes that will have to reset their connections at least $k > 1$ times during the attack is:

$$P_k = \sum_{i=1}^{\lfloor \frac{t_u}{t_a} \rfloor} P_{(k-1)}([t_u - i \cdot t_a],\ t_a,\ p_d) \cdot p_d$$

In general, for $t_d < t_a$, we have:

$$P_k = \sum_{i=1}^{\lfloor \frac{t_u}{t_a} \rfloor} P_{(k-1)}([t_u + t_d - i \cdot t_a],\ t_a,\ p_d) \cdot p_d$$

whereas if $t_d \geq t_a$, we have: $P_k = (\dfrac{t_u}{t_a} \cdot p_d)^k$

The probability that a client will be affected does not change, since the attack will continue to another set of overlay nodes and thus when the client tries to reconnect he will have the same probability of being affected, assuming he wants to keep using the system paying a penalty of t_d for each reset. Appendix A provides a rigorous computation of all the previous probability formulas.

Our spread-traffic system is invulnerable to these attacks, since there is no single node that maintains all client-specific state for a given client. Attacking a small percentage of overlay nodes will cause a corresponding packet loss in the end-to-end communication. If the attacked nodes are a small percentage of overlay nodes (corresponding to low packet loss), the end-to-end transport protocol (*e.g.,* TCP) should be able to recover. In Section 5 we show that, with a modest amount of packet replication and striping at the client, we can handle even massive DoS attacks against the overlay.

4.2 General ION Attack Resistance

It is worth estimating the attack volume that any ION system can withstand. Since ISP backbones are well provisioned, the limiting factors are going to be the links close to the target of the attack. The aggregate bandwidth for most major ISP POPs is on the order of 10 to 20 Gbps[4]. If the aggregate bandwidth of the attack plus the legitimate traffic is less than or equal to the POP capacity, legitimate traffic will not be affected, and the POP routers can drop the attack traffic (by virtue of dropping any traffic that did not arrive through the overlay). Unfortunately, there do not exist good data on DDoS attack intensities; network telescopes [17] tend to underestimate their volume, since they only detect response packets to spoofed attack packets. However, we can attempt a simple back of the envelope calculation of the effective attack bandwidth available to an attacker that controls X hosts that are (on average) connected to an aDSL network, each with 256 Kbps uplink capacity. Assuming an effective yield (after packet drops, self-interference, and lower capacity than the nominal link speed) of 50%, the attacker controls $128 \times X$ Kbps of attack traffic. If the POP has an OC-192 link (10 Gbps) to the rest of the ISP, an attacker needs $78,000$ hosts to saturate the POP's links. If the POP has a capacity of 20 Gbps, the attacker needs $156,000$ hosts. Although we have seen attack clouds of that magnitude (or larger), the ones used in actual attacks seem to be much smaller in practice. Thus, an overlay-protected system should be able to withstand the majority of DDoS attacks. If attacks of that magnitude are a concern, we can expand the scope of the filtering region to neighboring POPs of the same ISP (and their routers); this would increase the link capacity of the filtered region significantly, since each of the neighboring POPs see only a fraction of the attack traffic. Our discussion is not meant as a proof of security against DDoS attacks, but as an exploration of the limits of such mechanisms. It is important to note that these findings agree with other similar studies [22].

These numbers give us a baseline from which to determine how much more resistant our spread-spectrum system is compared to a basic indirection approach. Assume an attacker can create an effective attack bandwidth of K Mbps, and that each overlay node can be disabled through an attack sustaining D Mbps; thus, an attacker can simultaneously disable $\frac{K}{D}$ out of the N overlay nodes. When an attacker can observe a client's actions (*i.e.,* which overlay nodes a client routes traffic through), the effectiveness of the attack (defined as the probability of disrupting communications) is 1, as long as $K > D$. What is a likely value for D? The Click software router with commodity hardware [15] claims a switching capacity of 435,000 64-byte packets, or 222 Mbps. Taking a more conservative value of 50 Mbps, an attacker can saturate an overlay node by using $1,740$ hosts. Furthermore, an attacker controlling $100,000$ nodes (not enough to directly attack the target) can render approximately 60 geographically dispersed, well connected overlay nodes inaccessible at a time. Assuming an overlay network of a size comparable to Akamai's (approximately $2,500$ nodes), the attacker can render 2.5% of the overlay unusable.

To guarantee packet delivery at a given probability P_s in the presence of such attacks, we need to select the number of packet replicas R such that $P_s = 1 - (\frac{K}{D \times N})^R$ or $P_s = 1 - f^R$, where f is the percentage of the attacked nodes. If we assume that users initiate TCP connections with the protected server, then P_s should be no less than 90%, otherwise the connections stall [19]. From the formula for P_s and using the fact that $P_s = 0.9$ for TCP, we can compute the required bandwidth given the size of the network, or the fraction of nodes that need to be successfully attacked to disrupt the user's TCP session. For example, if we send each packet twice, *i.e.,* have a packet replication $R = 2$, the attacker has to bring down 32% of the nodes participating in the overlay network. For an over-

[4]For example, see `http://global.mci.com/about/network/interactive`

lay network of $2,500$ nodes, an attacker needs to gain access and coordinate a network of $1,375,000$ zombies. In addition, if we increase the packet replication value to $R = 3$, the percentage of nodes that need to get compromised jumps to 46% — almost half of the nodes in the overlay network.

To avoid imposing extra traffic on the network by replicating each packet, we can instead select the packets that we replicate at random with a probability P_r. Now P_s becomes $P_s = 1 - f(1 - P_r(1 - f))$ since the probability that a packet will fail the first time transmitted is f and the failure probability for the possibly replicated packet is $(1 - P_r(1 - f))$. Again, using $P_s = 0.9$ for TCP, we see that if we replicate 50% of the transmitted packets, the fraction of the nodes that need to get compromised is 17%, which is significant for medium to large overlay networks. Another approach to replication is to use forward error correction codes such as Erasure Codes, which we intend to examine in future work.

We experimentally verified the validity of this analysis with the prototype on the PlanetLab network, as we discuss next.

5. PERFORMANCE EVALUATION

Just as important as security is the impact of our system on regular communications, whether under attack conditions or otherwise; a prohibitively expensive mechanism (in terms of increased end-to-end latency or decreased throughput) is obviously not an attractive solution. In our experiments, we measured the communication overhead of our system in terms of end-to-end throughput and latency. To provide a realistic network environment, we deployed and used our prototype with 76 PlanetLab nodes.

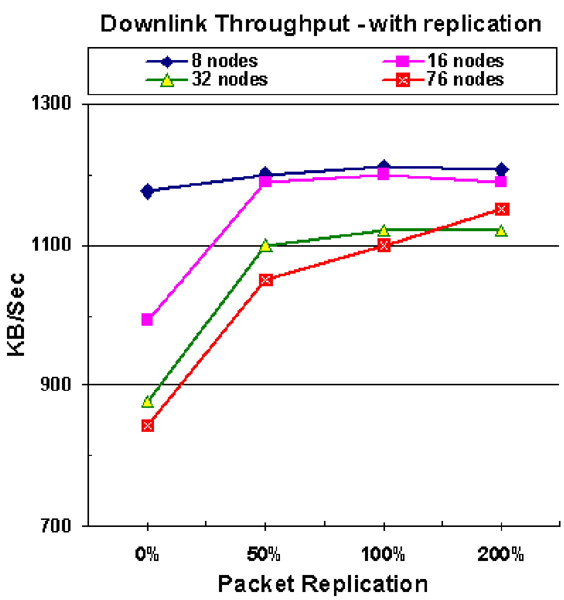

Figure 5: Throughput results in KB/sec. When we increase the replication, the results become closer to what we have observed for the direct connection (1250 KB/sec).

For our evaluation, we used a testbed consisting of Planetlab machines located at various sites in the continental US. Those machines were running UML Linux on commodity *x86* hardware and were connected using Abilene's Internet-2 network. Using these fairly distributed machines, we constructed our overlay network of access points by running a small forwarding daemon on each of the participating machines. In addition, we used two more machines, acting as client and server respectively. In our experiments, we measured link characteristics such as end-to-end latency and throughput when we interposed the overlay network of access points between the client and the target server. To measure throughput, we used a target server that was located at Columbia. For our latency measurements, we used `www.cnn.com` as the target. In both cases, the goal of the client was to establish a communication with the target server. To do so, the client used UDP encapsulation on the TCP packets generated by an SCP session and then spread the UDP packets to the nodes participating on the overlay network, as we described in Section 2.2. Those packets were in turn forwarded to a pre-specified overlay node (the secret servlet). This node decapsulated and forwarded the TCP frames to the target server. Since our throughput connection measurements involve a client and a server that were co-located, we effectively measured the worst possible scenario (since our otherwise local traffic had to take a tour of the Internet). A non-co-located server would result in a higher latency and lower throughput for a direct client-server connection, leading to comparatively better results when we use the overlay. Surprisingly, in some cases we can achieve better latency using the overlay rather than connecting directly to the server.

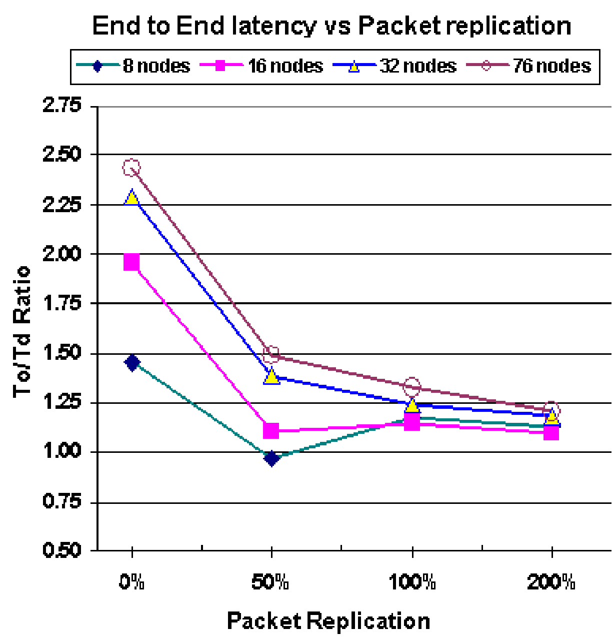

Figure 6: End-to-end average latency results for the index page and a collection of pages for www.cnn.com. The different points denote the change in the end-to-end latency through the overlay (T_o) when compared to the direct connection (T_d). Different lines represent different sized overlays. Increasing the replication factor, and for larger networks, we get lower average latency results because of the multipath effect on the transmitted packets.

Figure 5 shows that the impact on the downlink is only 33% in the worst case scenario, and it is easily amended by adding packet replication in the uplink direction. Again, we notice that the replication factor can cause a drop in the throughput for values $> 100\%$ in small overlay networks. Looking at the end-to-end average latency results in Figure 6, we notice that as we increase the replication factor, and for larger networks, we get better average latency results. In the worst-case scenario, we get a 2.5 increase in latency, which drops to 1.5 with 50% packet replication (*i.e.,* probability of replicating a packet of 50%).

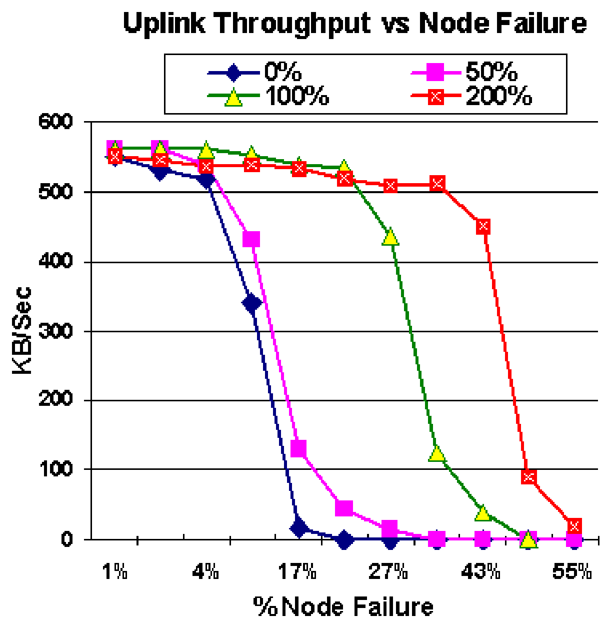

Figure 7: Throughput results in KB/sec when we utilize the uplink of our client under attack. The attack happens on a random fraction of the overlay nodes. Packet replication helps us achieve higher network resilience, something that we expected from our analytical results.

To measure the effectiveness of our system in the presence of attacks, we performed an attack by bringing down overlay nodes at random. In our experiment, the client kept spreading data across all overlay nodes, since he was unaware which of the overlay nodes were being attacked. We then varied the portion of the overlay nodes we attacked and we measured the throughput of the resulting link. Figure 7 shows the decrease in the uplink throughput of the system when under attack. The attack happens on a random fraction of the overlay nodes. When we do not use any replication and depend on TCP to "recover" the lost packets, the connection performs relatively well when the losses are up to 9%-10% of the total packets transmitted. Notice that as we increase the packet replication factor, we achieve higher network resilience, something that we also expected from our analysis. Corresponding results for latency are given in Figure 8.

Finally, we measured the number of tickets a single overlay node can generate. The ticket can be broken into four parts: the session key generation, the AES encryption of the ticket, the computation of the UMAC tag and the encryption of the packet using the client's public key. It appears that even for small size public keys (*e.g.,* 256 bits) the public key encryption takes up 95% of the ticket generation time. The number of tickets that can be produced by an overlay node decreases as we increase the size of the client's public key, as shown in Figure 9. Using a 3GHz Intel Pentium 4 machine, we were able to generate approximately 11,862 tickets/sec. In an ION with 128 nodes, the ticket-generation subsystem could sustain 1.5 million new users per second, assuming a random distribution of users across ION nodes.

6. RELATED WORK

As a result of its increased popularity and usefulness, the Internet contains both interesting targets and enough malicious and ignorant users that DoS attacks are simply not going to disappear on their own; indeed, although the press has stopped reporting such incidents, recent studies have shown a surprisingly high number of DoS attacks occurring constantly throughout the Internet [17, 5].

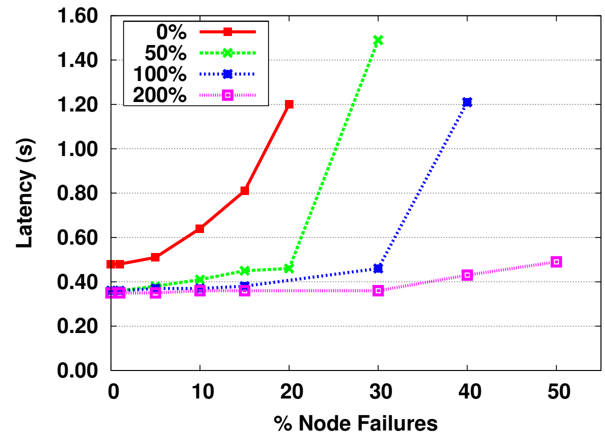

Figure 8: Impact of attacks against the overlay network on end-to-end latency. Different curves represent varying levels of packet replication. With 200% packet replication, latency increases by less than 25% when up to 50% of nodes are rendered unusable by an attacker.

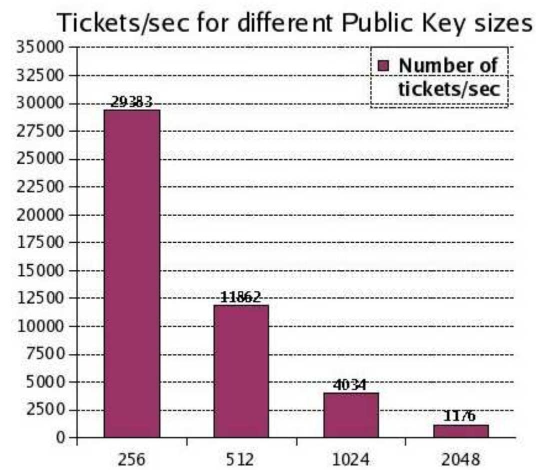

Figure 9: Tickets/sec produced from a single overlay node as we vary the size of the client's public key. The machine used was a 3GHz Intel Pentium 4 with 1GB of RAM.

SOS [13] first suggested the concept of using an overlay network to preferentially route traffic from legitimate users to a secret node (that can change over time), which is allowed to reach the protected server. All other traffic is restricted at the ISP's POP, which in most cases has enough capacity to handle all attack and legitimate traffic (the bottleneck is typically in the protected server's access link). Since the routers perform white-list filtering, the overhead of the system is negligible. In the original SOS approach, admission to the overlay was done based on public-key (or, more generally, cryptographic) authentication, requiring prior knowledge of the set of legitimate users. WebSOS [18] relaxes this restriction by adding a Graphic Turing Test to the overlay, allowing the system to differentiate between human users and attack zombies. MOVE [20] eliminates the dependency on network filtering at the ISP POP routers by keeping the current location of the server secret and using process migration to move away from targeted locations. Mayday

[1] explores separately the two main facets of the SOS architecture, filtering and overlay routing, with several alternative mechanisms considered. It is observed that in some cases, the various security properties offered by SOS can still be maintained using mechanisms that are simpler and more predictable. However, some second-order properties, such as the ability to rapidly reconfigure the architecture in anticipation of or in reaction to a breach of the filtering identity (*e.g.,* identifying the secret servlet) are compromised. In most other respects, the two approaches are very similar. An analysis of some security/performance design tradeoffs in IONs appears in [23]. Wang *et al.* [22] used an online network simulator to investigate the resistance of proxy networks (such as SOS) against simple DoS attacks. They conclude that the resistance of a proxy network to flooding attacks increases linearly with its size. However, they assume that users can instantaneously detect attacked ION nodes and switch to new ones with zero overhead, an assumption that did not hold for any ION architecture prior to ours.

[24] is the first system to create stateless flow filtering by having each router add "capabilities" to packets that traverse them; the receiver of these packets is then responsible for sending these capabilities to its peers, which will allow them to send traffic at higher rates (privileged traffic). Unprivileged traffic is limited to a fraction of the available bandwidth; thus, although a DoS attack can prevent new connections from being established (by overloading the control channel used to communicate these capabilities), existing connections will be unharmed. Estrin *et al.* first proposed a capability-like mechanism for network packets in [8].

Gligor [9] proposed the use of a server that can produce tickets at line speeds. Clients must obtain a ticket from this server before they are allowed to access a protected service. The approach is primarily geared towards application-level DoS protection. Anderson *et. al* [4] subsequently proposed a similar system for use at the network layer of an Internet-like architecture designed with a clean slate, assuming a distributed token server architecture and rate-limiting/filtering traffic on routers based on these tokens.

7. CONCLUSIONS

We examined the vulnerability of indirection-based overlay networks (IONs), as used for DDoS protection, to more sophisticated attackers than have been considered to date by proposed systems such as SOS, I3, MayDay and Tor. Our scope is both the simple types of flooding attacks, as well as more sophisticated attackers that can eavesdrop the victim's communication link and focus their attack on the specific hosts the victim attempts to connect to. Even with limited resources, a sophisticated attacker can disrupt all the victim's attempts to communicate with other nodes. We presented an analytical model that quantifies the impact of such attacks on the throughput of end-to-end communications, and quantified the resilience of ION DDoS defenses to simple congestion-based DDoS attacks. To our knowledge, this is the first non-trivial attack model for DoS attacks in the literature.

We proposed the use of a spread-spectrum-like paradigm to create per-packet path diversity. Using the same analytical models, we quantified the resistance resistance of our system to DDoS attacks and we showed that a reasonably sized overlay network can resist attacks much larger than we have seen to date. Our performance measurements using an experimental prototype on PlanetLab show that, *despite the interjection of an overlay mechanism between communicating peers,* there is very little to no increase in end-to-end latency when our system uses packet replication, and that throughput drops by less than 15% in all cases. Finally, we show that we can withstand attacks that involve millions of attackers, causing up to 40% of overlay nodes to become unreachable.

Our approach offers an attractive solution against congestion-based denial of service attacks in most environments, as it does not require modifications to clients, servers, protocols, or routers both in terms of hardware and in terms of existing software. Our plans for future work include a better characterization of the tradeoffs that we have explored so far, by introducing a coding scheme for the data transmission that will adapt to the network characteristics of each path used. Furthermore, we are looking into mechanisms to protect our system against attackers that can take over overlay nodes, subverting part of the infrastructure. Finally, we are interested in deployment and use of such a protection system on a larger scale than our experimental testbed to acquire operational experience in a real environment.

8. REFERENCES

[1] D. G. Andersen. Mayday: Distributed Filtering for Internet Services. In *Proceedings of the* 4^{th} *USENIX Symposium on Internet Technologies and Systems (USITS)*, March 2003.

[2] D. G. Andersen, H. Balakrishnan, M. F. Kaashoek, and R. N. Rao. Improving Web Availability for Clients with MONET. In *Proceedings of the* 2^{nd} *Symposium on Networked Systems Design and Implementation (NSDI)*, May 2005.

[3] D. G. Andersen, A. C. Snoeren, and H. Balakrishnan. Best-Path vs. Multi-Path Overlay Routing. In *Proceedings of the Internet Measurement Conferencee*, October 2003.

[4] T. Anderson, T. Roscoe, and D. Wetherall. Preventing Internet Denial-of-Service with Capabilities. In *Proceedings of the* 2^{nd} *Workshop on Hot Topics in Networks (HotNets-II)*, November 2003.

[5] M. Bailey, E. Cooke, F. Jahanian, J. Nazario, and D. Watson. The Internet Motion Sensor: A Distributed Blackhole Monitoring System. In *Proceedings of the ISOC Symposium on Network and Distributed Systems Security (SNDSS)*, pages 167–179, February 2005.

[6] J. Black, S. Halevi, H. Krawczyk, T. Krovetz, and P. Rogaway. UMAC: Fast and Secure Message Authentication. *Lecture Notes in Computer Science*, 1666:216–233, 1999.

[7] T. Diament, H. K. Lee, A. D. Keromytis, and M. Yung. The Dual Receiver Cryptogram and Its Applications. In *Proceedings of the* 11^{th} *ACM Conference on Computer and Communications Security (CCS)*, October 2004.

[8] D. Estrin, J. Mogul, and G. Tsudik. VISA Protocols for Controlling Inter-Organizational Datagram Flow. *IEEE Journal on Selected Areas in Communications*, May 1989.

[9] V. D. Gligor. Guaranteeing Access in Spite of Distributed Service-Flooding Attacks. In *Proceedings of the Security Protocols Workshop*, April 2003.

[10] K. P. Gummadi, H. V. Madhyastha, S. D. Gribble, H. M. Levy, and D. Wetherall. Improving the Reliability of Internet Paths with One-hop Source Routing. In *Proceedings of the* 6^{th} *Symposium on Operating Systems Design & Implementation*, December 2004.

[11] J. Ioannidis and S. M. Bellovin. Implementing Pushback: Router-Based Defense Against DDoS Attacks. In *Proceedings of the ISOC Symposium on Network and Distributed System Security (SNDSS)*, February 2002.

[12] M. Jakobsson and A. Juels. Proofs of Work and Bread Pudding Protocols. In *Proceedings of the IFIP TC6 & TC11 Joint Conference on Communications and Multimedia Security*, September 1999.

[13] A. D. Keromytis, V. Misra, and D. Rubenstein. SOS: Secure

Overlay Services. In *Proceedings of ACM SIGCOMM*, pages 61–72, August 2002.

[14] A. D. Keromytis, J. L. Wright, and T. de Raadt. The Design of the OpenBSD Cryptographic Framework. In *Proceedings of the USENIX Annual Technical Conference*, pages 181–196, June 2003.

[15] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek. The Click modular router. *ACM Transactions on Computer Systems (ToCS)*, 18(3):263–297, August 2000.

[16] A. Kuzmanovic and E. W. Knightly. Low-Rate TCP-Targeted Denial of Service Attacks. In *Proceedings of ACM SIGCOMM*, pages 75–86, August 2003.

[17] D. Moore, G. Voelker, and S. Savage. Inferring Internet Denial-of-Service Activity. In *Proceedings of the* 10^{th} *USENIX Security Symposium*, pages 9–22, August 2001.

[18] W. G. Morein, A. Stavrou, D. L. Cook, A. D. Keromytis, V. Misra, and D. Rubenstein. Using Graphic Turing Tests to Counter Automated DDoS Attacks Against Web Servers. In *Proceedings of the* 10^{th} *ACM International Conference on Computer and Communications Security (CCS)*, pages 8–19, October 2003.

[19] E. M. Nahum, M.-C. Rosu, S. Seshan, and J. Almeida. The effects of wide-area conditions on WWW server performance. In *Proceedings of the ACM SIGMETRICS*, pages 257–267, June 2001.

[20] A. Stavrou, A. D. Keromytis, J. Nieh, V. Misra, and D. Rubenstein. MOVE: An End-to-End Solution To Network Denial of Service. In *Proceedings of the ISOC Symposium on Network and Distributed System Security (SNDSS)*, pages 81–96, February 2005.

[21] L. von Ahn, M. Blum, N. J. Hopper, and J. Langford. CAPTCHA: Using Hard AI Problems For Security. In *Proceedings of EUROCRYPT*, May 2003.

[22] J. Wang, X. Liu, and A. A. Chien. Empirical Study of Tolerating Denial-of-Service Attacks with a Proxy Network. In *Proceedings of the* 14^{th} *USENIX Security Symposium*, pages 51–64, August 2005.

[23] D. Xuan, S. Chellappan, and X. Wang. Analyzing the Secure Overlay Services Architecture under Intelligent DDoS Attacks. In *Proceedings of the* 24^{th} *International Conference on Distributed Computing Systems (ICDCS)*, pages 408–417, March 2004.

[24] A. Yaar, A. Perrig, and D. Song. An Endhost Capability Mechanism to Mitigate DDoS Flooding Attacks. In *Proceedings of the IEEE Symposium on Security and Privacy*, May 2004.

APPENDIX

A. ANALYSIS OF SWEEPING ATTACKS

PROPOSITION 1. *The percentage of users that will have to reset their connections at least* $k > 1$ *times during the attack is:*

$$P_k(t_u,\ t_a,\ p_d) = \sum_{i=1}^{\lfloor \frac{t_u}{t_a} \rfloor} P_{(k-1)}(t_u - i \cdot t_a,\ t_a,\ p_d) \cdot p_d \qquad (1)$$

with t_u*: avg user time,* t_a*: attack time,* p_d*: % of nodes attacked simultaneously. We assume immediate attack detection (*$t_d = 0$*).*

PROOF. The percentage of users that will be affected by the attack at least once is:

$$P_1(t_u,\ t_a,\ p_d) = \frac{t_u}{t_a} \cdot p_d \qquad (2)$$

Notice that the above probability can go above 100% if $t_u >> t_a$, meaning that the attack will certainly affect the clients possibly more than once. When $P_1 > 100\%$ we say that $P_1 = 100\%$, *i.e.,* $P_1 = min(100, \frac{t_u}{t_a} \cdot p_d)$. We will prove (1) using induction.

Base case for $k = 2$, in that case (1) becomes:

$$P_2(t_u,\ t_a,\ p_d) = \sum_{i=1}^{\lfloor \frac{t_u}{t_a} \rfloor} P_1(t_u - i \cdot t_a, t_a, p_d) \cdot p_d \qquad (3)$$

In our model, the attacker can only attack $\frac{t_u}{t_a} \cdot p_d$ sets of nodes. We say that a client suffers an attack when the set of overlay nodes that he is connected to is attacked. The probability for a client to be at the first node is p_d. After realizing an attack is underway, in t_d time, the client will select a new overlay node. The probability that this new overlay node is part of the attack window, and thus the client will suffer another attack, is $P_1(t_u - t_a, t_a, p_d)$ since the attacker will have to spent t_a time attacking the first set of nodes.

Thus, the probability to be attacked at least twice when the client happens to be in the first set of attacked nodes is $P_1(t_u - t_a, t_a, p_d) \cdot p_d$. For a client connected to the second set of nodes the probability to be attacked twice is $P_1(p_d, t_u - 2 \cdot t_a, t_a) \cdot p_d$ since the attacker will have to spent $2 \cdot t_a$ time attacking the first and the second node before reaching any other node. Another way of saying the same thing is that the user will have $t_u - 2 \cdot t_a$ time left in the system reducing the probability of being attacked. A client that is connected to a node in the i^{th} set has a probability $P_1(t_u - i \cdot t_a, t_a, p_d)$ to be re-attacked. A client has probability p_d to be connected to a set and by summing up the fraction of clients connected to i^{th} set for which $t_u - i \cdot t_a > 0$, we get (3).

We assume that the formula holds for $k = j$ and we will prove that it holds for $k = j+1$. P_k is the probability that a client will be re-attacked at least k times. If the client is on the first set attacked, the probability of being attacked $j + 1$ times is the probability of initially being at the first set, which is p_d, multiplied by the probability that he will select overlay nodes which can be re-attacked j times in the $t_u - t_a$ remaining time. The probability of both being in the first attacked set and being re-attacked j more times is: $P^1_{j+1} = P_j(t_u - t_a, t_a, p_d) \cdot p_d$. For a node that connects initially to the i^{th} set we get that the probability of being attacked $j + 1$ times is $P^i_{j+1} = P_j(t_u - i \cdot t_a, t_a, p_d) \cdot p_d$. If we sum all the sets i for which $t_u - i \cdot t_a > 0$, we get (1). □

PROPOSITION 2. *In the general case. where* $t_d \geq 0$*, the percentage of nodes that will have to reset their connections at least* $k > 1$ *times during the attack is:*

a) if $t_d \geq t_a$ *we have that:* $P_k(t_u,\ t_a,\ p_d) = (\frac{t_u}{t_a} \cdot p_d)^k$

b) if $t_d < t_a$ *we have:*

$$P_k(t_u,\ t_a,\ p_d) = \sum_{i=1}^{\lfloor \frac{t_u}{t_a} \rfloor} P_{(k-1)}([t_u + t_d - i \cdot t_a],\ p_d,\ t_a) \cdot p_d \quad (4)$$

PROOF. To compute the probability when $t_d > 0$, we assume that the user is not going to be discouraged by the attack and will want to use the system for t_u time.

We derive (2) using the fact that since $t_d \geq t_a$, the client will have the same probability to select a set of overlay nodes that will be attacked as it had at the beginning of the attack: $\frac{t_u}{t_a} \cdot p_d$. The percentage of the users that will be attacked k times is $(\frac{t_u}{t_a} \cdot p_d)^k$

Equation (4) follows from proposition 1 if we change the usage time of a user from t_u to $t_u + t_d$, *i.e.,* the user will have to pay a penalty of t_d each time he is attacked, increasing his total time usage time by the same amount. □