A^{2}M: Access-Assured Mobile Desktop Computing

Angelos Stavrou[1], Ricardo A. Barrato[2], Angelos D. Keromytis[2], and Jason Nieh[2]

[1] Computer Science Department, George Mason University
[2] Computer Science Department, Columbia University

Abstract. Continued improvements in network bandwidth, cost, and ubiquitous access are enabling service providers to host desktop computing environments to address the complexity, cost, and mobility limitations of today's personal computing infrastructure. However, distributed denial of service attacks can deny use of such services to users. We present A^{2}M, a secure and attack-resilient desktop computing hosting infrastructure. A^{2}M combines a stateless and secure communication protocol, a single-hop Indirection-based network (IBN) and a remote display architecture to provide mobile users with continuous access to their desktop computing sessions. Our architecture protects both the hosting infrastructure and the client's connections against a wide range of service disruption attacks. Unlike any other DoS protection system, A^{2}M takes advantage of its low-latency remote display mechanisms and asymmetric traffic characteristics by using multipath routing to send a small number of replicas of each packet transmitted from client to server. This packet replication through different paths, diversifies the client-server communication, boosting system resiliency and reducing end-to-end latency. Our analysis and experimental results on PlanetLab demonstrate that A^{2}M significantly increases the hosting infrastructure's attack resilience even for wireless scenarios. Using conservative ISP bandwidth data, we show that we can protect against attacks involving thousands $(150,000)$ attackers, while providing good performance for multimedia and web applications and basic GUI interactions even when up to 30% and 50%, respectively, of indirection nodes become unresponsive.

1 Introduction

In today's world of commodity computers and increasing broadband network connectivity, the existing computing infrastructure imposes severe limitations on increasingly mobile users. Such users lack a common computing environment as they move between home, office, and while on the road. Mobile users have been forced to adapt by carrying around bulky laptop computers and other stateful devices with battery draining power needs. This approach is increasingly unsustainable as the management and security costs of owning and maintaining these devices grow, especially for large organizations with many users. Maintenance is particularly difficult with devices that may be roaming anywhere, on any network. Furthermore, these portable devices are inherently physically insecure and it is not uncommon for these machines to be damaged or stolen, resulting in the loss of any important data stored on them. This is a critical problem especially in health care computing, where HIPAA compliance is a requirement in supporting the clinical information access of highly mobile medical professionals. Even when such data can be recovered from backup, the time-consuming process of reconstituting the state of the lost machine on another device results in a huge disruption in critical computing service for the user.

Outsourced IT systems often utilize a thin-client computing model to decouple a user's applications and desktop computing session from any particular end-user device by moving all application logic to hosting providers. Graphical displays are virtualized and served across a network to a client device using a remote-display protocol, with application logic executed on the server. Clients transmit user input to the server, and the server returns screen updates. Examples of popular thin-client platforms include Citrix MetaFrame [8], Microsoft Terminal Services [9], AT&T Virtual Network Computing (VNC) [25]. Because all application processing is done on the server, the client only needs to be able to display and manipulate the user interface, enabling clients to be simple and stateless.

A key issue that must be addressed to ensure that users obtain reliable and secure access to hosted computing services is protection of the server infrastructure and the client's connection against denial of service attacks, particularly of the distributed kind (DDoS). DDoS attacks are an increasing occurrence in today's Internet, aiming to deny use of a service to legitimate users [33]. The same ubiquitous network connectivity that improves access to a service provider for legitimate mobile users, also increases an attacker's ability to launch a DDoS against a service provider, sometimes as part of an extortion scheme [12]. One type of DoS attack that is difficult to identify and isolate involves sending enough attack traffic which will cause the links close to the servers to be congested and eventually drop all useful traffic. The potential of such attacks to disrupt user access to applications and data is an important challenge that must be addressed before ASPs can achieve mass acceptance. Unfortunately, existing DDoS protection mechanisms either require large-scale deployment, or offer unacceptably high latency and latency variance [1, 15], especially when under attack. To be of any practical use, interactive and real-time applications such as GUI operations and multimedia streaming demand a low-latency pipe at all times.

In this context, we introduce *Access-Assured Mobile* (A^{2}M) desktop computing, a hosted computing infrastructure that combines a remote-display architecture with a stateless indirection-based network (IBN) composed of dedicated nodes. A^{2}M provides both protected and efficient access to hosted desktop computing environments, even in the presence of denial of service attacks. Nodes participating in the IBN communicate only to exchange control messages, but not to route the client's data, unlike previous overlay-based approaches [1, 15]. A^{2}M clients exploit the path diversity naturally exhibited by a distributed IBN to "spread" their traffic such that directed attacks do not cause service disruptions. To further alleviate any potential delays introduced by the IBN and reduce the latency in the end-to-end communication, A^{2}M uses a number of other optimizations at the remote display level to minimize the impact these delays may have on the user's experience. A^{2}M combines a simple low-level display protocol and a server-push model to minimize client-server synchronization and network round-trips. Atop this basic model, A^{2}M implements higher-lever mechanisms, such as client-managed cursor display, shortest-job-first display command scheduling, and a non-blocking drawing pipeline, further increasing the overall interactive response of the system. The contributions of our work are:

- We implement and evaluate A^{2}M in the real Internet using PlanetLab. Our experiments show that A^{2}M introduces very little latency in most scenarios.
- We are the first to conduct realistic (non-simulation) experiments to evaluate the resilience of our system against DDoS attacks using wireless nodes, and its performance under attack. Our results validate the design of A^{2}M, showing good performance for multimedia and interactive applications even with 30%–50% of the IBN nodes under attack.

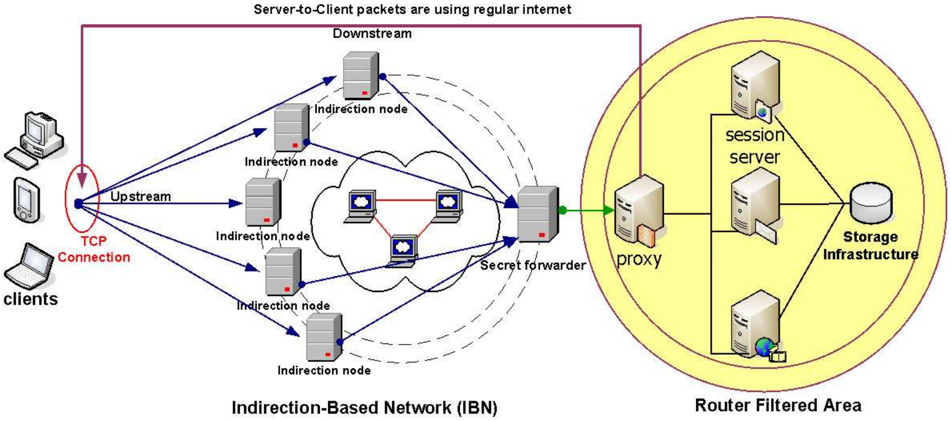

Fig. 1. A^{2}M Architecture. The two directions of the client-server connection take different paths: the client-to-server direction goes over the indirection-based network, while the server-to-client direction goes directly to the client (not through the infrastructure).

2 A^{2}M Architecture

As shown in Figure 1, A^{2}M's architecture is divided in two major components: the hosting infrastructure and the access infrastructure. The hosting infrastructure provides an environment for desktop sessions where a user's session is decoupled from any particular end-user access device, by moving all application state to hosting servers. Applications run within these servers, and their display output is redirected over the network to the access device. Redirection is performed by a per-session virtual display driver that translates from application display-draw commands to A^{2}M's display protocol commands. The protocol commands are then forwarded to the client device for display. A^{2}M extends previous work on desktop hosting infrastructures such as MobiDesk [4] by providing mechanisms that provide continuous access to hosted desktop sessions, even in the presence of distributed denial of service attacks on the hosting servers.

A^{2}M's access infrastructure provides the connection between users on the network and the applications running on the hosting servers. Users make use of a simple client application that merely forwards input events to the applications running on the server, and processes display updates generated in response to these events. This application model results in a highly asymmetric network traffic pattern. On one side, input events (headed uplink, or upstream toward the server) are very small pieces of information that are generated at a relatively slow, human-dependent rate. On the other hand, display updates (headed downlink, or downstream toward the client) are orders of magnitude larger and are generated as fast bursts of activity. For example, during web browsing, a single user input event (a mouse click on a link) results in a full-screen update having to be displayed (the destination web page).

The traffic asymmetry is made more pronounced when we consider the different roles and importance of input events and display updates. In an interactive system user experience is dictated by the response time, which in turn is determined by how quickly input events are processed and display updates are made visible to the user. If response time is too high, the user will become exasperated and frustrated with the system. Since a single input event triggers the generation of display updates, guaranteed delivery of each event becomes crucial for the performance of the system. On the other hand, humans are known to be more tolerant to partial updates than to longer response times, because partial updates provide feedback to their actions. Delivery of updates should then be made such that updates can begin to be displayed as soon as possible, even if the complete update takes longer to appear.

The resource centralization around the hosting infrastructure results in a threat model where denial of service attacks on the system will only affect the uplink direction, *i.e.,* the traffic **to** the hosting servers, by saturating the network links and queuing buffers close to the servers or by directly attacking the hosting infrastructure servers. Therefore, it is crucial for A^{2}M to protect this communications channel from interference, blocking unwanted traffic close to the attacker before it can reach the service providing machines. On the other hand, the downlink direction will for the most part be relatively free of noise, and without any need to be protected. Note that denial of service attacks typically affect the uplink direction, *i.e.,* the traffic **to** the server, by saturating the network links and queuing buffer close to the server. The downlink direction is relatively free of noise. Thus, we are primarily interested in protecting the client-to-server traffic from interference; the opposite direction does need typically any such protection.

Taking advantage of both the traffic asymmetry and the threat model, A^{2}M partitions bi-directional connections between the client and the server into an indirected client-to-server multi-path and a direct server-to-client path. The IBN takes care of routing input events and other client-to-server traffic and protects the hosting infrastructure. Protection is performed by acting as a distributed firewall that conceptually distinguishes between authorized client-generated traffic, and unauthorized and possibly malicious traffic. Traffic permitted to traverse through the IBN is directed to a filtering router close to the hosting servers, whereas all other traffic is dropped or rate-limited providing a distributed "shield" against both network congestion and host directed attacks. In Section 2.2, we provide an estimation of the resistance of the system, using this filtering mechanism, to denial of service attacks, in terms of the average number of machines that must participate in the attack.

The direct server-client path in turn ensures that large and bursty display updates are delivered to the client as quickly as possible, even if parts of them are lost or delayed and need to be retransmitted. A^{2}M's approach represents a sharp departure from traditional interactive client-server architectures, where a vulnerable bi-directional direct connection provides the only means of communication between the client and the server. We should note that A^{2}M does not preclude routing both traffic directions over the IBN, albeit at a possible increase in the end-to-end latency when no replication of packets is present. Since this mode is not necessary for our usage scenario, we do not further consider it in this paper.

2.1 System Operation

To provide seamless and ubiquitous connectivity, A^{2}M encapsulates all functionality within a self-contained client application that manages communication with the indirection infrastructure, forwards user events to hosted applications, and displays application output on the local device. To access a desktop session, users must first obtain access to the IBN, which in turn allows them to authenticate with the hosting infrastructure, and then gain access to their session. Users need to be recognized as legitimate in order for the IBN to distinguish their traffic from other unauthorized, possibly malicious traffic. In contrast to traditional service providing infrastructures such as web-content distributors, A^{2}M requires users to be authenticated and does not allow anonymous users, because only authorized users should be able to connect to the hosting infrastructure. A^{2}M ties the authentication requirements of the IBN and the hosting infrastructure into a single, seamless process.

Client Authentication: Before a client is allowed to send traffic through the IBN, it must obtain a *ticket,* which is then included in all subsequent packets sent to the IBN, until it expires. The ticket is used by the IBN nodes to authenticate the user, validate the routing decisions, and prevent malicious (or subverted) clients from utilizing a disproportionate amount of bandwidth. To obtain a ticket, the client contacts an indirection node at random using a ticket establishment protocol described in detail in previous work [30]. This protocol is fully distributed and resilient to CPU exhaustion attacks. Furthermore, the ticket issuing process is protected against replay and IP spoofing attacks. At the end of the protocol, the client and the IBN have authenticated each other, and the client is in possession of a ticket. The ticket contains a session key K_u, a range of sequence numbers for which it is valid (more on this later), and the IP address of the client, all encrypted under K_M, a secret key negotiated periodically (*e.g.,* every few hours) among all indirection nodes. Note that only the indirection nodes can decrypt the ticket; clients treat the ticket as an opaque value that they must provide to the AAN with each packet they need to forward. A second copy of K_u is independently encrypted under the client's public key. This ticket can only be used by the client to continue the authentication protocol (*i.e.,* prove liveness for both the IBN nodes and the client. Once the full two-party authentication is completed, the last indirection node provides the client with a ticket that is not "restricted," *i.e.,* the corresponding flag inside the ticket is cleared. As we discussed in the previous section, the tickets are periodically refreshed, to avoid situations where a malicious user distributes a valid session key and ticket to a large number of zombies that then simultaneously send attack traffic through the IBN.

The connections to the hosting infrastructure are asymmetric: the client-to-server traffic will travel through the IBN, while the server-to-client traffic will use regular Internet routing. In the case where a session does not already exist, a new session is created and populated, before the client is allowed to connect to it. The authentication and connection setup process is done transparently by the client application, and it does not require special support from the underlying devices. This simplicity allows A^2M users to access their sessions from almost any number of Internet-enabled devices.

Once the connection to the hosting server is established, the client will be recognized as a legitimate user, and user input events will be allowed to traverse the indirection nodes and be routed to the hosted applications. This process continues until the user disconnects from the session, at which point the client's ticket is revoked and the connections are closed. Since a disconnected client is no longer allowed to use the system, previously legitimate devices cannot be reused as attack tools on the infrastructure.

2.2 Assured Access Indirection Network

We have implemented the *Assured Access Network* (AAN), which significantly extends the ideas of SOS [15] and Mayday [1]. Our approach, shown in Figure 1, is to spread the packets from the client across all indirection nodes in a pseudo-random manner. This new communication mechanism protects the client-server connection establishment and provides uninterrupted connectivity to the target server throughout the client's session. The admitted packets are internally forwarded to a secret forwarder (selected at random, and changing over time), which is allowed to forward traffic to the utility server. Only authorized clients are allowed to use the IBN and contact the hosting servers and these clients are provisioned in advance (*e.g.,* at registration time) with the appropriate authentication material, such as an RSA public/private key pair and a public-key certificate [6, 7]. AAN works in conjunction with filtering routers close to the hosting infrastructure, to allow only traffic from the IBN's secret forwarders to reach A^2M's hosting servers. All other traffic is considered unauthorized and possibly malicious, and therefore filtered out.

Contrary to previous overlay architectures, our system achieves this filtering without the use of overlay routing to transfer the client's request to the server. In our system, legitimate packets are reflected to the secret servlet(s) generating a one-hop indirection network. As shown in Figure 1 there is no single path between the client and the server - instead packets are spread from the client to the indirection nodes creating a single-hop multi-path effect. Both the use of the single-hop indirection and the multi-path routing permit our system to scale well in terms of latency, as we shall see in Section 3. For more details on the overlay architecture itself, see [30].

2.3 AAN Encapsulation

When using AAN, every packet sent by a client to an indirection node contains four fields: a client identifier, the ticket, an authenticator, and a monotonically increasing sequence number. Recall that the ticket contains the session key and the maximum sequence number for which the ticket is valid, and is encrypted and authenticated under a secret key known only to the indirection nodes. Note that these indirection 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 IBN with a given session key. The client identifier is a random 32-bit value that is selected by the indirection node that authenticated the client, and is used as an index in the table of last-seen sequence number, maintained by each indirection node for each active client. The authenticator is a fast hash function, such as UMAC [5], computed over the session key and the whole packet (including the ticket, sequence number, and client identifier). Thus, the only amount of state each indirection node needs to maintain per active client are the client's identifier and the last sequence number seen from that particular client. Assuming that both the client identifier and the sequence number are 32-bit values, each indirection node needs to maintain only 64 bits of state for each client; thus, if the system has 1 million active clients, we will only need 8 MB of state — easily manageable even if it is stored in main memory, given current prices of RAM.

2.4 AAN Operation

A client transmitting a packet through the IBN uses the session key and the sequence number as inputs to a pseudo-random function (PRF). The output is treated as an index to a publicly available list of indirection nodes, through which the packet will be routed. The list of available indirection nodes does not need to change frequently, even if nodes become unavailable (*e.g.,* for maintenance purposes), and can be downloaded by clients independently of the protected communication. For the purposes of this paper, we assume that clients trust the IBN's entry points. Discussion and analysis of an environment where access points cannot be safely trusted can be found in [34].

The client encapsulates the original packet (addressed to the final destination) inside a packet for the indirection node, along with the information identified above (client identifier, ticket, sequence number, authenticator). The packet is then forwarded through the IBN to the secret forwarder for that particular destination, and from there to the final destination.

An indirection node that receives such a packet first 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. It then decrypts the ticket, obtaining the session key for that client, with which it verifies the authenticator. The indirection 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 indirection node updates the sequence number table and forwards the packet to the secret forwarder. Packets with lower or equal sequence numbers are considered duplicates (either accidental artifacts of the underlying network, or malicious replays by attackers) and are quietly dropped.

2.5 Attack Resistance

Here we attempt to give a simple analysis on the expected resiliency of our system. Additional work is needed to further refine the model and validate our assumptions. However, this analysis should serve as a good first-order approximation on the effectiveness of the approach.

Since the Internet (and ISPs') 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, according to an informal poll of several providers. 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 IBN). Unfortunately, there do not exist good data on DDoS attack volumes; network telescopes [19] 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 or cable network, each with 256Kbps 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 (10 Gbps) connection 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 are much smaller in practice. Thus, an IBN-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. Additional work is needed to determine the practical limits of the system. In Section 3 we give some experimental results on the resilience of our system against attacks that target the IBN itself.

3 Implementation and Experimental Results

To demonstrate the feasibility of the proposed architecture, we have implemented an A^{2}M prototype which hosts and protects Linux-based desktop sessions. We deployed the indirection nodes of our prototype in 80 PlanetLab nodes, while having the client and server reside in our local network. Our architecture spreads the packets across all indirection nodes. Perhaps the most surprising aspect of our implementation is its size: excluding cryptographic libraries and the JFK protocol, the code implementing the complete functionality of the system consists of 1,600 lines of well commented *C* code. The JFK implementation itself adds another 2,500 lines of code. Although this is a prototype implementation and does not include management code and other facilities that would be required in a production system, we feel that the system is surprisingly lightweight and easy to comprehend.

The implementation consists of the code for the indirection nodes, as well as code running on each client that does the encapsulation and initial routing. A detailed description of MobiDesk may be found in [4]. 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 consists of a linked pairs of virtual network interfaces and character devices that a user-level process can read and write. 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 1a 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 indirection nodes according to the protocol. The code running on indirection nodes receives these UDP packets, authenticates and forwards them to the secret forwarder, 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 indirection nodes, there is also a daemon listening for connection establishment packets from the clients.

In evaluating A^{2}M, we focused on two metrics: the quality of service in terms of latency, as this is perceived by the end user, and the system's resilience when under attack *i.e.,* node failures. PlanetLab provides a realistic network environment for our experiments that stresses the performance of our system because the packets follow different, highly variant paths to reach the protected server. In our experiments, we protected the uplink traffic from the client to the server routing it through the IBN, while the return path followed normal Internet routing (outside the IBN).

Our testbed consisted of a client PC simulating a typical remote-display access device, a server where the benchmark applications executed, and 80 indirection hosts deployed across various PlanetLab locations in the US and Canada. The client computer had a 450Mhz Intel Pentium-II CPU and 128MB RAM running Debian with Linux 2.4.27. Our client PC was chosen to reflect the type of low-power, thin-client devices which we expect to become A^{2}M's access devices. The laptop PC had a 1.5Ghz Intel Pentium M and 1GB RAM running Debian with Linux 2.6.10. The server was an Intel dual-Xeon 2.80GHz with 1GB of RAM running RedHat 9 with Linux 2.4.20.

We measured the performance of A^{2}M in web, video, and basic interactive tasks as representative applications of typical desktop usage. Our web measurements used the Mozilla 1.6 browser to run a benchmark based on the Web Page Load test from the Ziff-Davis i-Bench benchmark suite. The benchmark consists of a sequence of 54 web pages containing a mix of text and graphics. The browser window was set to full-screen resolution for all platforms measured. Video playback performance was measured using Mplayer 1.0pre3 to play a 34.75 second video clip of original size 352*x*240 pixels displayed at full-screen resolution. For our interactive tests we recorded a number of sessions where simple interactive tasks were performed. Recording the sessions allowed us to reliably play back the exact same tasks under different network conditions. The measure of performance for these tests was the latency experienced by a user performing the specific task. The primary measure of web browsing performance was the average page-download latency in response to a mouse-click on a web page link. To minimize any additional overhead from the retrieval of web pages, we used a conservative setup where the web server was directly connected to the hosting server through a LAN connection. The primary measure of video playback performance was video quality [21], which accounts for both playback delays and frame drops that degrade playback quality. For example, 100% video quality means that all video frames were displayed at real-time speed. On the other hand, 50% video quality means either that half the video frames were dropped when displayed at real-time speed or that the clip took twice as long to play even though all of the video frames were displayed.

We first examined the effects that the basic indirection network and various levels of packet replication had on the overall performance of the system. The levels of replication tested were no replication, 50% (meaning one extra copy of each packet with probability 0.5), 100% replication (one extra copy of each packet) and 200% replication (two extra copies of each packet). We also measured the impact of the IBN size by running our experiments on 8 and 80 nodes participating in the IBN. We ran a baseline test where we used a direct LAN connection between the client and the server. Since the indirection nodes were deployed over a wide area with varying network latency, this test provided us with a very conservative measurement of the indirection overhead. In a realistic A^{2}M deployment, the client and server will typically reside at different, topologically distant locations. In that case, it is entirely possible for the indirection path to provide better connectivity characteristics than a direct connection due to the multi-path effect, which allows the packets originating from the client to follow a route with lower latency towards the end server [2,11,14,32]. Although not shown in our results for ease of viewing, we also compared the performance of A^{2}M to that of MobiDesk and found it to be the same on the direct connection case.

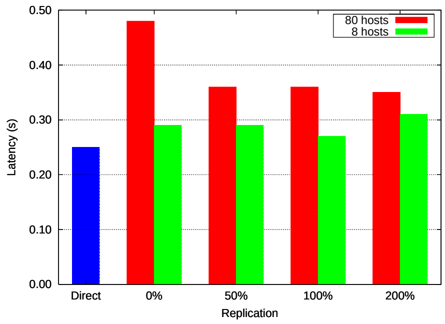

Fig. 2. Web latency *vs.* packet replication. The leftmost bar shows the latency when connected directly to the server using a LAN and no protection.

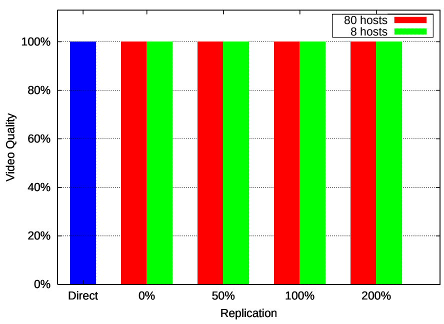

Fig. 3. Video quality *vs.* packet replication — video quality remains 100% under all test scenarios even for a 80-node IBN with no packet replication.

Figure 2 illustrates the end-to-end average web latency results as perceived by the client. We can see that even for the worst-case scenario, an 80-node IBN without packet replication, the overhead from the indirection results in a latency increase of only 2 (*i.e.*, twice the latency of the baseline direct connection). When 50% packet replication is used (*i.e.,* replicating a packet with probability 0.5), the overhead drops significantly to 40% for the 80-node IBN. The drop in the overhead is due to the variant path latency of nodes participating in the IBN. TCP does not behave optimally when packets appear to have high variance when arriving at the end server out of order. Adding packet replication decreases this variance, as the same packet follows more than one paths with different latency and the end server uses the one that arrives first. Boosting the replication beyond 50% follows the law of diminishing returns, as each additional increase in replication gives us less latency improvements. Care must be taken however, as too much packet replication can cause performance degradation, since bandwidth is "wasted" on duplicate packets. This is better exemplified by the results on the 8-node network using 200% replication. The 80-node network does not exhibit the same adverse affect because its average path latency is higher, allowing the secret gateway enough time to process the encapsulated packets received by the IBN.

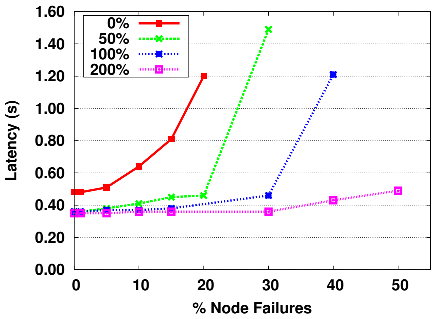

Fig. 4. Web latency under DDoS attack. Latency increases in response to increased nodes failure.

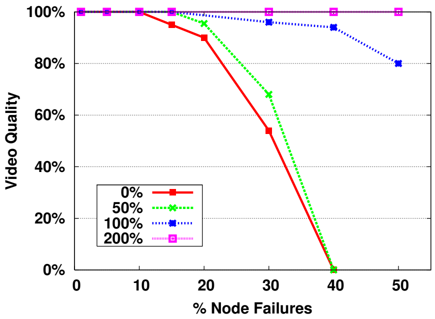

Fig. 5. Video quality under DDoS attack. Video quality drops only after a substantial percentage of nodes become unresponsive.

To measure our system with an application that could generate more upstream traffic and required the system to maintain its quality of service above a threshold for latency, we used video playback. Figure 3 shows the results for video quality as measured at the client side. We can clearly see that A^{2}M performs optimally under all test scenarios, providing the same perfect video quality as the direct LAN connection scenario, even for the worst-case scenario of the 80-node IBN deployed over a WAN with no packet replication.

The behavior of the overall system under attack was measured using a simulated denial of service attack that targeted the IBN itself. Our threat model assumes that the attacker can render a fraction of the nodes participating in the IBN unresponsive, thus inducing packet loss in the TCP connection of a user connected to the hosting server.

All resilience tests were run on the 80-node IBN network. When attacked, a node stops forwarding packets from the client to the end host, acting as a mute node. Since there is no immediate feedback, clients do not know which A^{2}M nodes are operating and which are suppressed by the attacker. Figure 4 illustrates the effects on the average web page latency as we increase the percentage of node failure, and demonstrates both the resilience of A^{2}M and the advantages of packet replication. Without packet replication, latency quickly degrades to twice that of the direct connection when we have 15% of node failures, and reaches three times for 20% node failure. On the other hand, employing packet replication allows A^{2}M to maintain an almost constant latency that is very close to the direct connection, even under 50% A^{2}M node failure, in the case of 200% replication.

Interactive Applications Although video streaming and web browsing are both representative and demanding applications, we felt that we needed to include another set of experiments that require a high level of synchronization between the upstream and the downstream channel. We performed four different tests, each representing typical interactive operations on a desktop environment. The tests were performed by first recording a session of a user performing the appropriate operation, and then playing back the session in a number of different experimental scenarios. Our measure of performance was the user-perceived latency in response to the interactive operations. The four tests performed were: echo, minimize/maximize window, scroll, and move window. The echo test measured the time it takes for a line of text to appear on the screen after the user has pressed and depressed a key. The minimize/maximize window tests measures the time it takes to maximize a window after the user has pressed the maximize button, and then (after the window has been maximized) to minimize it after the user has pressed the minimize button. The scroll test measures the time it takes to scroll down a full-screen web page in response to a single `Page Down` key-press, and then the time it takes to scroll back to the top by leaving the `Arrow Up` key pressed. Finally, the move window test measures the time it takes to move a window across the screen. The window's size is about one fifth of the screen's size, and it is moved by dragging the window while the left-mouse button is pressed. The window operation is opaque, *i.e.,* the contents of the window are continuously redrawn as the user performs the move operation.

The end-to-end latency the end users experience for these operations is shown in Figures 6, 7. These measurements show that without using packet replication, and for attacks up to 20% of the indirection nodes, the client's end-to-end latency increases only by a factor of 2.5 when compared to the direct, non-protected case. On the other hand, if we permit packet replication, we notice an increase in latency only after 50% of the indirection nodes become unresponsive. In some cases, for attack intensities that exceeded 20% of the indirection nodes and without replication the network conditions were too adverse for the test to complete.

4 Related Work

The need to protect against or mitigate the effects of DoS attacks has been recognized by both the commercial and research world, given the ease with which such attacks

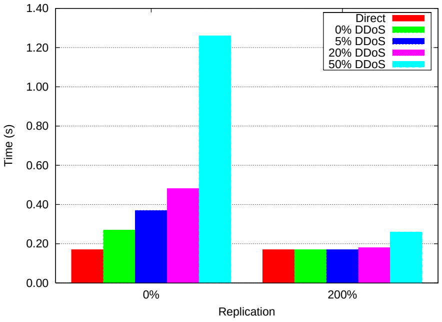

Fig. 6. Interactive performance for the echo test. Even without replication and with attacks affecting up to 20% **of the IBN nodes, the client's end-to-end latency increases only by a factor of** 2.5 **when compared to the direct, non-protected case.**

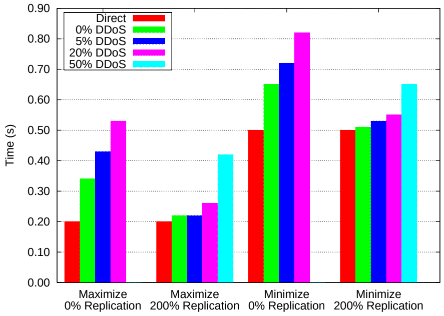

Fig. 7. Interactive performance for minimize/maximize window test. Without replication and for attacks affecting up to 20% **of the IBN nodes, the client's end-to-end latency increases only by a factor of** 2**.**

can be launched and their frequency [19]. A^{2}M provides an attack-resilient utility infrastructure for mobile desktop computing. Due to space limitations, we do not discuss here the extensive work on DDoS prevention or mitigation that requires widespread deployment inside the network or the use of new protocols and end-applications [10, 13, 18, 22–24, 26, 29, 35].

A^{2}M builds on the ideas proposed by the MobiDesk [4] desktop hosting infrastructure, and its remote display architecture, THINC [3]. In contrast to A^{2}M, MobiDesk and THINC do not address the problem of potential attacks on their infrastructure and use a direct connection to communicate between user devices and hosting servers. This makes the system defenseless and vulnerable to simple denial of service attacks that may cause hosted desktop sessions to become unavailable to users. Attacks may either target the hosting infrastructure or the communication channels providing the service, and render the MobiDesk infrastructure useless.

SOS [15] first suggested the concept of using an overlay network to preferentially route traffic, using multi-hop overlay routing, from legitimate users to a secret node that is allowed to reach the protected server. All other traffic is restricted at the Internet Service Provider's Point-of-Presence (POP), which in most cases has enough capacity to handle all attack and legitimate traffic. The same idea is used in MayDay [1]. WebSOS [20] relaxes the requirement for a priori knowledge of the legitimate users, by adding a Graphic Turing Test to the overlay, allowing the system to differentiate between human users and attack zombies. MOVE [31] 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. A system similar to MOVE is described in [16]. There, 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 forwarder) are compromised.

All of these overlay-based systems impose a high latency overhead, making them unfit for time-critical applications. To route the client traffic, these systems create an overlay route whose length increases with the number of overlay nodes (usually with $O(log(n))$, where n is the size of the overlay). Such an increase in the path length leads to higher (and highly varying) end-to-end latency. Moreover, these systems are vulnerable to attacks that target the connection state that is kept by each of their overlay nodes: by attacking a specific node, the attacker forces the users connected to it to detect this attack and re-establish both their connectivity and authentication credentials to another, potentially healthy, overlay node. An attacker can force the users to reset their connections repeatedly, making the system impractical.

Several remote display and thin-client architectures are widely used today, including the X-Window System [27], Citrix MetaFrame (ICA) [8], Microsoft Terminal Services (RDP) [9], VNC [25], and SunRay [28]. However, none of these systems provide resiliency against DoS attacks as A^{2}M does. None of these systems take advantage of the asymmetry in remote-display traffic to improve performance in environments with high variability of network latency. As shown on previous studies [17, 21], all of these systems suffer major performance degradations in high-latency network environments. X, ICA, and RDP use high-level display protocol primitives that can result in worse performance due to the additional synchronization required.

5 Conclusions

We presented A^{2}M, an attack-resilient and latency efficient desktop hosting infrastructure based on a single-hop indirection network. A^{2}M exploits multi-path routing, packet replication, and the high asymmetry inherent to interactive display traffic, to assure access to remote desktop sessions, even in the presence of high-volume DoS attacks. Contrary to the current DoS protection mechanisms, our system guarantess both availability and uninterrupted connectivity to the end server providing a truly secure end-to-end connectivity model. Furthermore, in a departure from traditional client-server systems, A^{2}M provides an asymmetric client-server connection consisting of an indirected client-to-server multi-path, and a direct server-to-client connection. A^{2}M's indirection-based overlay acts both as a first-level distributed firewall and as a redirection mechanism for performance-critical user input-events going from the client device to the hosting servers. In turn, the direct server-to-client connection provides quick delivery of display updates, to provide quick response time and good user experience.

A prototype of A^{2}M was implemented in Linux and we evaluated its performance on PlanetLab. Our experimental results show that, as opposed to existing DDoS protection mechanisms, A^{2}M has minimum latency overhead and can provide good interactive performance for web, video, and general interactive applications. Furthermore, we demonstrate that A^{2}M significantly increases the attack resilience of the hosting infrastructure, being able to provide perfect video playback and low-latency web browsing and GUI interactions even in the presence of large attacks on the infrastructure. A^{2}M maintains 100% video quality in a number of remote video display scenarios, despite the use of overlay routing. Furthermore, end-to-end latency increases by less than 5% even when 40% of nodes have been rendered unusable by an attacker. Given its performance and resilience to DoS attacks, A^{2}M represents a step forward towards realizing the vision of computer utilities that provide ubiquitous, secure, and assured-access desktop computing.

6 Acknowledgements

This work was supported by NSF Grants CNS-07-14277, CNS-04-26623, and Google Inc.

References

1. D. G. Andersen. Mayday: Distributed Filtering for Internet Services. In *Proceedings of the* 4th *USENIX Symposium on Internet Technologies and Systems (USITS)*, March 2003.
2. 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.
3. R. Baratto, L. Kim, and J. Nieh. THINC: A Virtual Display Architecture for Thin-Client Computing. In *Proceedings of the* 20th *ACM Symposium on Operating Systems Principles (SOSP)*, Oct. 2005.
4. R. Baratto, S. Potter, G. Su, and J. Nieh. MobiDesk: Mobile Virtual Desktop Computing. In *Proceedings of the* 10th *Annual ACM International Conference on Mobile Computing and Networking (MobiCom)*, September 2004.
5. 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.
6. M. Blaze, J. Feigenbaum, J. Ioannidis, and A. D. Keromytis. The KeyNote Trust Management System Version 2. RFC 2704, September 1999.
7. CCITT. *X.509: The Directory Authentication Framework*. International Telecommunications Union, Geneva, 1989.
8. Citrix ICA Technology Brief. Technical White Paper, Boca Research, 1999.
9. B. Cumberland, G. Carius, and A. Muir. *Microsoft Windows NT Server 4.0, Terminal Server Edition: Technical Reference*. Microsoft Press, August 1999.
10. D. Dean, M. Franklin, and A. Stubblefield. An Algebraic Approach to IP Traceback. In *Proceedings of the ISOC Symposium on Network and Distributed System Security (SNDSS)*, pages 3–12, February 2001.
11. 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* 6th *Symposium on Operating Systems Design & Implementation (OSDI)*, December 2004.
12. G. Hulme. Extortion online. *Information Week*, September 13, 2004.
13. 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.
14. A. Kaella, J. Pang, and A. Shaikh. A Comparison of Overlay Routing and Multihoming Route Control. In *Proceedings of ACM SIGCOMM*, pages 93–106, August/September 2004.
15. A. D. Keromytis, V. Misra, and D. Rubenstein. SOS: Secure Overlay Services. In *Proceedings of ACM SIGCOMM*, pages 61–72, August 2002.
16. S. M. Khattab, C. Sangpachatanaruk, D. Moss, R. Melhem, and T. Znati. Roaming Honeypots for Mitigating Service-Level Denial-of-Service Attacks. In *Proceedings of the* 24th *International Conference on Distributed Computing Systems (ICDCS)*, pages 238–337, March 2004.

17. A. Lai and J. Nieh. Limits of Wide-Area Thin-Client Computing. In *Proceedings of the ACM International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS)*, pages 228–239, June 2002.
18. J. Li, M. Sung, J. Xu, and L. Li. Large-Scale IP Traceback in High-Speed Internet: Practical Techniques and Theoretical Foundation. In *Proceedings of the IEEE Symposium on Security and Privacy*, May 2004.
19. 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.
20. 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.
21. J. Nieh, S. J. Yang, and N. Novik. Measuring Thin-Client Performance Using Slow-Motion Benchmarking. *ACM Transactions on Computer Systems (TOCS)*, 21(1):87–115, February 2003.
22. C. Papadopoulos, R. Lindell, J. Mehringer, A. Hussain, and R. Govindan. COSSACK: Coordinated Suppression of Simultaneous Attacks. In *Proceedings of DISCEX III*, pages 2–13, April 2003.
23. B. Parno, D. Wendlandt, E. Shi, A. Perrig, B. Maggs, and Y.-C. Hu. Portcullis: protecting connection setup from denial-of-capability attacks. *SIGCOMM Comput. Commun. Rev.*, 37(4):289–300, 2007.
24. P. Reiher, J. Mirkovic, and G. Prier. Attacking DDoS at the source. In *Proceedings of the* 10^{th} *IEEE International Conference on Network Protocols*, November 2002.
25. T. Richardson, Q. Stafford-Fraser, K. R. Wood, and A. Hopper. Virtual Network Computing. *IEEE Internet Computing*, 2(1):33–38, January/February 1998.
26. S. Savage, D. Wetherall, A. Karlin, and T. Anderson. Practical Network Support for IP Traceback. In *Proceedings of ACM SIGCOMM*, pages 295–306, August 2000.
27. R. W. Scheifler and J. Gettys. *X Window System*. Digital Press, 3^{rd} edition, 1992.
28. B. K. Schmidt, M. S. Lam, and J. D. Northcutt. The interactive performance of SLIM: a stateless, thin-client architecture. In 17^{th} *ACM Symposium on Operating Systems Principles (SOSP)*, volume 34, pages 32–47, December 1999.
29. A. Snoeren, C. Partridge, L. Sanchez, C. Jones, F. Tchakountio, S. Kent, and W. Strayer. Hash-Based IP Traceback. In *Proceedings of ACM SIGCOMM*, August 2001.
30. A. Stavrou and A. Keromytis. Countering DoS Attacks With Stateless Multipath Overlays. In *Proceedings of the* 12^{th} *ACM Conference on Computer and Communications Security (CCS)*, pages 249–259, November 2005.
31. 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.
32. A. Su, D. R. Choffnes, A. Kuzmanovic, and F. E. Bustamante. Drafting Behind Akamai (Travelocity-Based Detouring). In *Proceedings of ACM SIGCOMM*, pages 435–446, September 2006.
33. DoS-Resistant Internet Working Group Meetings. `http://www.communicationsresearch.net/dos-resistant`, February 2005.
34. 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.
35. 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.