Mantis: A Lightweight, Server-Anonymity Preserving, Searchable P2P Network

Stephen C. Bono, Christopher A. Soghoian, Fabian Monrose
Information Security Institute
The Johns Hopkins University
Baltimore, Maryland, USA

{sbono|csoghoian|fabian}@jhu.edu

Technical Report TR-2004-01-B-ISI-JHU

June 17, 2004

Abstract

We introduce Mantis, a searchable, peer-to-peer (P2P) network of anonymous nodes aimed at protecting the privacy of individuals acting as servers in the network. In order to minimize the traffic relayed by peers, servers transfer data directly to clients via a separate, source-spoofed UDP stream. This is extremely important as users of a P2P system are content to give up bandwidth while downloading or uploading, but are unwilling to donate the majority of their bandwidth in order to relay traffic for other peers. By relaxing the requirement of full client anonymity, Mantis enables efficient data transfers from anonymous servers while limiting the bandwidth costs incurred by other peers participating in the network.

1 Introduction

The need for anonymous communication on the Internet has motivated a number of anonymous networking techniques. Since the early 80s, applications have been designed to protect Internet users from censorship, privacy violations, and any number of lawsuits.

For the most part, the overall goal of these systems is to ensure privacy for a sender and recipient, such that neither party can be identified as an endpoint of the communication stream, and that the two parties cannot be linked. Moreover, the goal of most anonymizing systems has been to protect the privacy interests of only the client in a client-server relationship, where only the client knows the identity of the server with which to communicate, but not vice versa. This is representative of privacy with respect to performing web transactions [14], sending electronic mail [13, 5], as well as publishing documents [4].

However, as Internet speeds increase, individual users desire to and are more capable of performing the actions of a server themselves. Individuals often act as servers when participating in file sharing networks or hosting personal web pages. It is increasingly in their interest to remain anonymous when providing such services.

Strong privacy requirements aimed at protecting the client have challenged the ability of anonymizing networks to meet the performance requirements of their users. Many solutions are high-latency, as communications between client and server are typically weighted very heavily in the server-to-client direction, requiring server responses to be forwarded through a number of anonymizing hops.

To address these issues, we introduce Mantis, a searchable, peer-to-peer (P2P) network of anonymous nodes aimed at protecting the privacy of in-

dividuals acting as servers in the network. Our system takes advantage of the asymmetric nature of client-server communication, where servers send far more data than the clients they serve, and alleviates the overhead inherent in other anonymous networks where participants must forward entire transactions for other peers.

Mantis's ability to provide anonymity is modeled after Crowds [14], a system for anonymizing web transactions. Two nodes, a client and server, establish a channel through a crowd of other users which they use to search, respond to service requests, and later coordinate a full transfer session. Anonymity is preserved by delegating the responsibility of sending a message from one peer to another, making the true initiator of a message extremely difficult to determine.

Unlike Crowds, however, our network is arranged in a tree-like structure, similar to Gnutella [21], providing users with the ability to search the P2P network. Each participant connects to multiple peers and forwards search requests received from one peer to all others. This allows each search to reach a vast number of participants acting as servers as they are propagated throughout the network in a ripple-like manner. Search replies are returned to the initiator along the reverse path traversed by the original search request, and a tunnel between the client and server is established through which future transactions between the two nodes may be performed.

In order to minimize the traffic relayed through the P2P network, servers transfer data directly to clients via a separate, source-spoofed UDP stream, leaving only control data to be tunneled through the crowd. This reduces the amount of bandwidth exhausted by peers for relaying messages as well as increases overall speed and throughput significantly more than other anonymizing networks.

Finally, our system does not restrict the entry or exit of nodes in the network. Nodes along forwarding paths are assumed to be volatile and may exit at any time. We show that our protocol is reliable even as tunnels extending throughout the network are broken.

The remainder of this paper is organized as follows: Section 2 discusses related work. Section 3 discusses the goals and terminology. Section 4 provides a detailed explanation of the methodology and implementation of Mantis. Section 5 delivers performance results comparing Mantis to systems with similar goals. Section 6 is a detailed security analysis of our scheme. Section 7 discusses directions for future work. Section 8 concludes our paper.

2 Related Work

Crowds, Mixes [3] and Onion Routing [9] each present different methods for providing either sender anonymity, receiver anonymity, sender-receiver unlinkability, or a combination of the three, when performing transactions over the Internet. In each of these architectures, the sender (or client) is required to know beforehand the identity of the receiver (the server). All these methods strive first and foremost to protect the identity of the client. Our scheme differs in both of these regards. In Mantis, clients lack knowledge of the server's identity and whereabouts prior to searching, and protecting the anonymity of the server is paramount.

Crowds originally introduced a scheme to enable a large group, or "crowd", of Internet users to anonymize their actions on the web by repeatedly delegating the responsibility of making web transactions to other crowd members. One crowd member performing an action on behalf of another cannot know with certainty whom they are assisting. Each crowd member that forwards messages in this manner, including the true initiator, can plausibly deny being the message source, as they too may have been given the request to fulfill for another crowd member.

Crowds alone is incapable of meeting our design goals, as it primarily protects the client, does not present an environment where a client may seek out an anonymous server, and has poor throughput as peers must forward all data both ways between client and server. Additionally, Crowds restricts the creation of static paths to periodic intervals and is faced with global path reconstruction should a participant abruptly disconnect. Mantis is robust in the face of disconnect-

ing peers, does not require prior knowledge of a server's identity, and utilizes the exponential number of paths formed by creating a tree structure when seeking out anonymous servers.

The first protocol that we are aware of to relax the need for a bi-directional path through an anonymizing network is Hordes [12]. Hordes provides client anonymity by including the client as a recipient of a multicast transmission from the server to a subset of users. The intended recipient accepts the transmission while peers not interested in receiving the multicast ignore it. By multicasting the majority of data from server to client, nodes along the forwarding path are relieved of using large amounts of their upstream bandwidth for assisting in message transmission. Though Hordes provides an improvement over Crowds, peers in this scheme are not entirely free of bandwidth donation while participating, since receiving these multicast transmissions still wastes valuable downstream bandwidth. Mantis, on the other hand, uses a direct UDP connection for server-to-client communication and affects no peer other than the client and server themselves. Only a small amount of control data must be sent along the forwarding path in the direction of client-to-server to maintain reliability of the communication. A comparison of Hordes and Crowds demonstrates how improvements are made to link utilization and network latency by eliminating the need for a return path through the Crowd. Mantis takes advantage of the same property and greatly reduces the utilized bandwidth along the forwarding path, even more so than Hordes.

A subsequent protocol, AFPS [16], similar in design to Hordes, uses multicast to achieve anonymity and introduces the , as far as we know notion of responder anonymity. The idea of hidden servers has appeared in onion routing as well with the advent of the second-generation onion router, Tor [6]. However, these schemes do not meet our design objectives. In addition to posting search terms, each system requires all servers to advertise information needed in order for clients to locate them, which includes a multicast address in the case of AFPS, and a set of onion routers when using Tor. AFPS further restricts clients and servers by only allowing periodic join times for entering the network. In contrast, Mantis allows servers to handle all search queries directly, avoiding the need to register search terms and allowing instantaneous admission to the network.

As multicast is not a widely supported service, the practical uses of Hordes and AFPS are limited in scope, and their application to P2P networks run by individuals (including home users) is minimal. Mantis addresses these limitations by utilizing only universally accepted protocols. Mantis also achieves improved efficiency over Hordes, AFPS and Tor by not subjecting relay nodes to unnecessary , we haven't talked about traffic that peers in a multicast group or onion route inevitably encounter.

Two systems aimed at producing a searchable anonymous P2P network are UDPP2P [24] and MUTE [22]. UDPP2P uses a highly inefficient process of broadcasting source-spoofed UDP search messages and control data to all other participating clients. MUTE requires all data being transferred to be forwarded hop-by-hop through the P2P network. Both of these systems are susceptible to disconnections disrupting file transfers, bottlenecking at the hop with the least available bandwidth, ineffective searching mechanisms, security flaws, and suffer from depth-of-search performance issues which we discuss in Section 6.

3 Goals and Terminology

The charts in Figure 1 provide an illustration of the chief differences between Crowds and Mantis. Crowds is structured so that a static path may be formed, through which web transactions are delegated between peers, providing a client's anonymity. Neither the server nor any peer is capable of determining the true initiator of a service request. Mantis adapts the same concepts in providing anonymity, where no peer can know for certain the identity of a client or server that has initiated a service request or reply.

The tree structure is formed by broadcasting search requests throughout the network. Nodes cache recently seen messages and drop duplicates in order to prevent routing loops from forming.

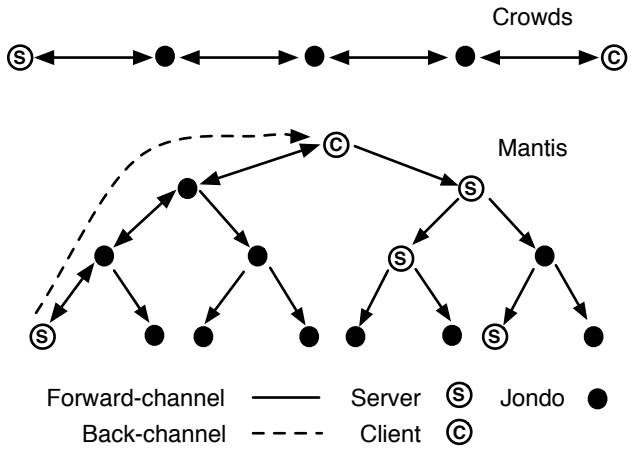

Figure 1: Crowd Paths vs. Mantis Trees

As a request propagates throughout the tree, it reaches more and more nodes potentially acting as anonymous servers. If the search criteria matches, a server will respond along the reverse path traversed by the original service request.

In this paper, we use the following terminology (for the sake of consistency, some of the terminology has been borrowed from section 4 of Crowds [14]): A *jondo* is any node in the P2P network and is capable of acting as a client, server, or message relayer on behalf of other jondos. The directory server that allows jondos to find other peers is called a *blender*. The hop-by-hop path on which jondos forward messages from client-to-server is called a *back-channel*, and a pair of communicating jondos over a back-channel do so in a *session*.

Mantis has been designed to meet the following goals:

- *Full server anonymity.* Jondos acting as servers in the P2P network have plausible deniability with respect to the services they provide (i.e. what files are served by a participant in a file-sharing network). Additionally, any client requesting a service does not first need to know the identity of the server.

- *Partial client anonymity.* Jondos searching for a service have plausible deniability for the requests that they initiate until the transfer begins, at which time a client's identity is revealed to the server.

- *Searchable network.* Clients may search for services provided by an unknown server. The server is not required to register or post search parameters to a forum prior to being located. Servers are capable of receiving search queries themselves and reviewing them individually.

- *Minimal back-channel overhead.* The amount of work that must be performed by each jondo acting as a relay in the back-channel is reduced to only handling control data. This allows for more anonymous connections to be maintained and for a greater allowable search depth.

- *Ease of entry/exit from the network.* Mantis allows for the easy addition of peers to the network without requiring path reconstructions or periodic entry times. As the lifetimes of jondos in the Mantis network are assumed to be short, Mantis is capable of handling the reconstruction of broken back-channels upon the untimely removal of jondos along the path.

The Blender As in Crowds, the blender acts as a directory server for jondos in the Mantis P2P network. It allows jondos to register so they can locate and be located by other peers in the network. When a jondo initializes, it can register with the blender. This registration is optional, as the P2P network is functional without an active blender, but it is necessary for the purpose of locating peers. Registration requires a jondo to reveal its IP address and a listening port for new connections.

We note that it is not necessary for the blender to be a specific entity; it can be any kind of forum allowing jondos to register and others to retrieve this information. However, revealing additional information to the blender is discouraged. We discuss in section 6 different attacks a blender can perform after acquiring such information.

JONDOS While bootstrapping, a jondo typically queries a blender for a list of connections. After identifying another jondo willing to participate, the two nodes perform an unauthenticated key exchange to agree upon a shared encryption key. All further communication between these two nodes is encrypted using this key. The jondo repeats this procedure until it has obtained a user defined minimum number of connections.

Once a jondo has acquired a sufficient number of connections, it may actively participate in the P2P network as a client or server, and is required to passively participate by forwarding messages on behalf of its neighbors.

4 Communication

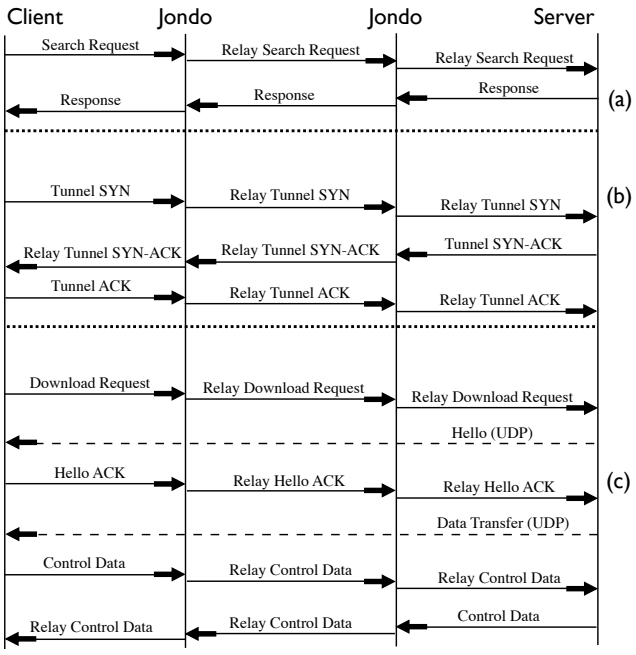

Figure 2: The Mantis Protocol: (a) Searching, (b) Establishing a Tunnel, (c) Data Transfer

Messages between jondos in the Mantis protocol are formatted as follows:

{SRC_ID, DST_ID, TYPE, DATA}

where SRC_ID and DST_ID are random source and destination identifiers used for individual sessions, TYPE indicates the message's purpose, and DATA contains any information incidental to the message. The DST_ID field is used to route messages through the back-channel to their intended recipient. DST_ID values of 0 indicate broadcast messages such as searches and are relayed to all neighbors.

Anonymous communication in the Mantis network is made possible by masking the true source and destination addresses of jondos by using random identifiers. A jondo receiving a message with a previously unseen source identifier associates the jondo that relayed this message as the next hop along the return path. A message arriving with a known identifier is relayed to the known next-hop for the message's DST_ID value.

4.1 Searching

Each jondo is connected to a number of other jondos, who are in turn connected to additional jondos. A broadcasted service request is sent from the originating jondo to all neighboring jondos. Each neighbor subsequently re-broadcasts the service request to all of its connected neighbors, and so on. Search requests spread through the network in a ripple-like form, similar to other power-law networks [11] such as Gnutella. Duplicate broadcast messages are ignored to prevent routing loops, and a tree structure is formed. The tree is rooted at the initiator of the service request, and each node below the root node is a recipient of the request. However, no node is aware its location relative to the true initiator within the tree.

To search, the client first generates a random session identifier that will thereafter be associated as the source address for this transmission. This value is used by the initiator to identify responses and allows jondos to record next-hop information for the return path. The client then creates a message with the chosen SRC_ID, sets the DST_ID value to 0 to indicate broadcast, sets the TYPE field to indicate a search and finally includes any search terms as DATA.

For each neighboring jondo, the client encrypts a copy of the entire message with the shared key between the client and neighbor. The messages are then sent to the respective neighbors and the

client awaits responses.

4.2 Relaying

Clients and servers intend to anonymize their actions with the help of jondos acting as relays between the two endpoints of the communication. Jondos must maintain a table of destination identifiers mapping to next-hop jondos where messages should be forwarded.

When a message is received by a jondo it is first decrypted using the key shared by this jondo and the jondo delivering the message. Depending on the value of the DST_ID field, there are a number of actions a jondo relaying a message may perform. The jondo first checks whether this DST_ID is intended for this node, and if it is, accepts the message and continues. If the message has an unknown DST_ID value, the message is dropped. Otherwise, the neighboring jondo is retrieved from a table and the message is prepared to be relayed, unless the message's DST_ID field indicates the broadcast value 0, in which case the message is prepared for forwarding to all neighbors.

While it is necessary for broadcast messages to reach as many nodes as possible, doing so mindlessly is damaging to the network as a whole if these messages propagate too far. Consequently, relaying jondos are required to perform a weighted coin toss to decide whether to continue the message propagation. We defer discussion of the need for a limited path length for performance to section 4 and the security implications of a probabilistic dropping mechanism to Section 6.

4.3 Responding

A server willing to meet the needs of a client first creates a random session identifier of its own. This is used as the SRC_ID for the server in the reply message to be returned to the client. The DST_ID field is set to the search originator's identifier. The SRC_ID field to the the responder's newly generated identifier. The message TYPE is set to indicate a search response, and any additional information needed such as search results are included in DATA field. Finally, the message is encrypted with the key shared between this node and the next-hop jondo mapped to by the DST_ID, and the message is relayed. Jondos along the path then relay the message back to its original source.

4.4 Tunneling

When a client wishes to communicate directly with a server it must create a tunnel over the back-channel to the server. The two entities, client and server, participate in an unauthenticated key exchange to agree upon a shared secret key. All subsequent messages are encrypted end-to-end using this key, in addition to being encrypted hop-to-hop between relaying jondos. Any further communication through the tunnel is done by creating a message with the appropriate pseudo-source and destination addresses as well as a DATA field encrypted using the shared key between client and server.

Our system favors client-server communication where the server transmits the bulk of the data, allowing the server to communicate with the client directly over UDP. This requires the client to reveal its IP address to the server, which may be done over the encrypted tunnel. The server then communicates directly to the client anonymously over a source-spoofed UDP stream. The back-channel tunnel is only used to communicate control data for the reliability of this stream between the client and server. If need be, the back-channel can be used for any kind of communication between client and server.

4.5 Reconnecting

Mantis is designed to tolerate peers with short life spans. Losing any jondo along the back-channel disrupts the connection, with the result being that the client can no longer communicate to the server. As the number of jondos forming a back-channel increases, the back-channel itself becomes increasingly volatile. The time needed to complete bulk data transfers from server-to-client may outlast the existence of the back-channel. This makes it imperative that a server be capable of reestablishing a back-channel.

This can be accomplished as the server is aware of the client's true identity and Mantis has relaxed restrictions on protecting client anonymity. The server passes a message requesting that a new back-channel be established with the client, including the client's identity and listening port. The back-channel reestablishment message is relayed between jondos a number of times, each time performing a weighted coin toss, until finally one of the relay jondos makes a direct connection with the client. The back-channel is then reestablished and communication can continue.

Revealing the client's identity can be avoided in smaller tightly knit networks. In smaller networks where nearly all nodes can be reached from any given point within the P2P community, only the pseudo-source identifier needs to be broadcast until an alternate path is found.

We note that reconnection is only possible once a full tunnel has been established between client and server and the client has revealed its address to the server. If a relay jondo leaves the network before the tunnel is fully established, then there is no hope for performing this reconnection.

5 Performance Analysis

Our performance criteria for an anonymizing system is based on the amount of work performed by nodes relaying messages on behalf of other nodes. We conjecture that, for the most part, users of a P2P system are willing to give up bandwidth when downloading for themselves or uploading as a server, but are unwilling to donate the majority of their downstream and upstream bandwidth to assist in transfers for other peers. The overhead of constant message relaying makes it increasingly difficult for peers to function as clients or servers themselves.

Our second quantification of performance is the measurement of overall data throughput. In a point-to-point communication, the transmission rate is hindered solely by the server's upload speed or the client's download speed. However, in an anonymizing network requiring relays, the highest achievable data rate is that of the slowest relay. This presents a particularly devastating problem to an anonymizing network that allows the participation of home users. Home broadband users often have asymmetric download and upload speeds, with significantly less upload bandwidth. When relay jondos are required to forward all data, dialup modem users and outbound-bandwidth restricted home users continually bottleneck the system.

5.1 Eliminating Passive Jondo Overhead

The largest drawback to using a system such as Crowds or MUTE is that they perform poorly given each of our performance objectives. Each requires jondos acting as relays to forward every data packet between client and server. The transfer speed of the communication is only as fast as the slowest relay jondo's upload bandwidth, and in an environment where many P2P users are home users with restricted outbound bandwidth, this is unacceptable. Servers and clients experience frustratingly low data rates and relay jondos are aggravated by the saturating effects of assisting others.

Crowds's and MUTE's design goals prohibit the two systems from capitalizing on the properties of a client-server relationship. Typically the server sends far more data to the client than vice versa. This is clearly the case with web servers or file sharing networks where an individual node in the act of sharing is considered a server. Within Crowds and MUTE, these bulk-data transfers must be relayed hop-to-hop through the P2P networks to maintain client anonymity.

By revealing the client's identity to the server, Mantis allows the complete extraction of bulk-data transfers from the back-channel in the direction of server-to-client. The server is able to send a UDP data stream (while spoofing the source address to maintain its own anonymity) directly to the client avoiding all relay jondos. It is only necessary for a minimal amount of control data to be transmitted through the back-channel for reliability purposes. By doing so, relay jondos are almost completely relieved of donating bandwidth to other users and transmission speed is no longer

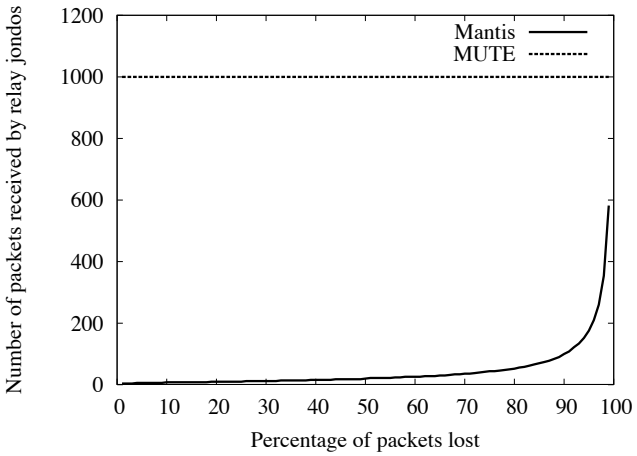

Figure 3: Packets Forwarded by Relay Jondos in Mantis vs. MUTE

dependent upon the slowest jondo.

Figure 3 compares the amount of work a relay jondo performs when transferring all communications through the back-channel (MUTE) vs. transferring only control data through the back-channel (Mantis), as the percentage of lost UDP packets increases. The amount of work is measured as the total number of packets that must be relayed. The method for reliable UDP used for our testing is a simplistic method whereby all packets are numbered and transferred to the client directly. The client then asks the server for packets that were not received. The server transfers each of the remaining packets and again the client asks the server for those packets that were lost. This cycle continues until all packets are received, requiring two control packets per cycle. Our tests were performed in the network simulator Simnet [10], with the client and server transferring 1000 packets through a single relay jondo. The results are averaged over 1000 runs for each loss rate at 1% intervals.

Packet relaying is performed over reliable TCP connections and therefore the number of packets to be relayed is static regardless of packet loss. When sending direct via UDP to the client, only control data must be relayed through the back-channel. As the loss-rate increases, the number of necessary control data packets increases only slightly. The graph in Figure 3 shows that even after an unrealistic 90% of all UDP packets are lost, relay jondos still have far less work when only forwarding control data as opposed to relaying the entire message.

It is noted that as the UDP loss rate increases, the work that must be performed by the server increases accordingly. The server is required to repeatedly retransmit lost datagrams to the point where total transfer time would be faster if using the reliable back-channel for the entire communication. However, as the efficiency of Mantis rests on the amount of work performed by relay jondos, forcing entire communications through the back-channel would violate this performance objective.

Hordes also benefits by removing bulk transfers from the back channel. Client anonymity in Hordes is achieved by adding the client to a multicast recipient group populated with other jondos. This eliminates the need for a two way back-channel just as Mantis does. However, multicast decoy jondos in the network are still affected by slews of incoming traffic that they will ultimately ignore. As the size of multicast groups and the number of server-to-client communications increases, the bandwidth saturating effect on jondos acting as decoys will intensify.

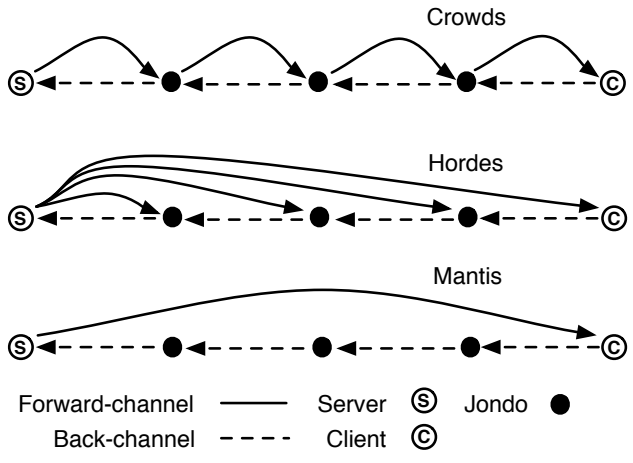

Figure 4: End Host Communication in Crowds, Hordes, and Mantis

A comparison of Hordes to Crowds in [12] ver-

ifies that one can achieve a tremendous performance improvement by obviating the forwarding of most messages, and analysis shows that in doing so that the round-trip latency of messages is cut nearly in half. Still, as the number of server-to-client transfers within the network grows, relay (or in this case, decoy) jondos are forced to donate larger and larger amounts of their incoming bandwidth to the greater good, until they become saturated. Figure 4 conceptualizes the communication between end-hosts in Crowds, Hordes and Mantis, showing how a direct server-to-client communication channel requires the least work to be performed by each relay jondo.

Mantis has an additional advantage over Hordes in that it operates over protocols that are available to all potential participants, whereas multicast is not deployed widely enough to allow many home users to benefit by using Hordes.

5.2 Path Length Restrictions

When relaying a service request, each jondo performs a weighted coin toss before decided whether or not to propagate the message. If the probability of forwarding is large enough and the network size is adequate, service requests reach exponential numbers of potential servers. This has the benefit of insuring volumes of responses, but has the serious drawback of saturating the network with requests, replies and established tunnels. We refer to nodes reached by a propagating service request as being within the *vicinity* of the initiator.

For each hop a service request is expected to propagate, the number of nodes in the initiator's vicinity increases by a factor of the average number of connections a jondo maintains. The number of nodes in the vicinity of a given node can be calculated as k^n, where k is the average number of connections to neighboring jondos and n is the expected path length. The expected path length n is computed as as $n = \sum_{i=1}^{\infty}(i \times p^i) = (1-p)/p^2$, where p is the probability that a service request is dropped.

A jondo must forward requests and may manage tunnels for any node within its vicinity. It should be clear that as the size of a jondo's vicinity increases, the amount of work it must perform in order to assist other nodes increases accordingly. The number of service requests, replies and tunnel establishments relayed by a jondo each increase relative to the expected path length. It is therefore imperative to select a forwarding probability to provide a vicinity size that does not saturate jondos within the network.

6 Security Analysis

In this section we discuss the degrees to which participants in the P2P community can maintain their anonymity as originators or recipients of messages, and potential attacks aimed at revealing a participant's true identity. In doing so, we adapt the metric put forth in Crowds for quantifying degrees of anonymity.

6.1 Degrees of Anonymity

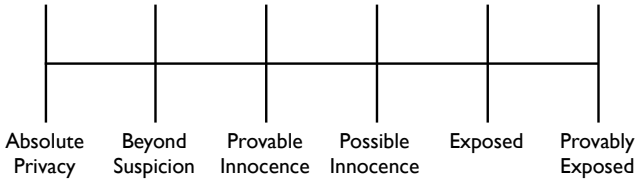

Figure 5: The Crowds Anonymity Metric

The anonymity metric provides an "informal continuum", with levels ranging from *absolute privacy*, in which it is near impossible to determine a participant's identity, to *provably exposed*, where the participant's identity is revealed and their actions cannot be refuted. A jondo is *beyond suspicion* if it is no more likely to be the originator of a message than any other jondo in the network. A jondo is considered to have *probable innocence* if an attacker has reason to suspect the jondo over some other participants as being the originator of a message, but is still most likely not the originator. Finally, a jondo has *possible innocence* when from the attacker's point of view there is a non-trivial probability that the participant is not the message initiator.

6.2 Adversarial Model

In our system, participants must be protected from a range of potential attackers. The adversary may take the form of a malicious blender and or colluding jondos. We describe attacks aimed at building topologies of the P2P network, identifying the client or server as communicating end points, and linking communicating clients and servers. We discuss each aspect of the system that may be malicious, defenses against possible attacks and the degree of anonymity achieved during those attacks.

We make the following assumptions about an adversary that may pose a threat to participants in the Mantis network:

- An adversary may control one or more blender services, but cannot control all blender services.
- An adversary may control one or more jondos, but cannot control all existing jondos in the network.
- An adversary may view all encrypted traffic arriving at or originating from one or more nodes.
- An active adversary located on a back-channel may launch a man-in-the-middle (MITM) attack against the end-to-end encrypted tunnel, revealing all control data and possibly the client's identity.
- An adversary is capable of initiating, forging or replaying any number of broadcast messages with the intent to expose a server based on a statistical analysis of the replies returned.
- Encryption is sufficiently strong such that adversaries are incapable of recovering the original plaintexts for encrypted messages without use of the secret key.

6.2.1 Practical Attacks

We omit a number of attacks that are outside the scope of this paper. For example, denial of service attacks performed by blenders or malicious jondos at the message routing level can be trivially performed. However, this is true of any anonymizing network as well as most file-sharing networks. The same may be said for malicious servers lurking in the P2P network. It is acceptable and required in our system for a client to reveal its identity to a server for proper communication, and therefore discussion of how a malicious server may expose a client is irrelevant. Finally, we do not discuss the threat posed by a global eavesdropper capable of witnessing all actions performed by all participating nodes. This adversary is for the most part theoretical, except in relation to small networks, and impractical with respect to a widely distributed P2P network. We therefore limit discussion to that of a local eavesdropper capable of overseeing only a minimal number of peers.

6.3 Client and Server Anonymity

It is a design goal of our system to partially protect client anonymity. However, as a necessary consequence of receiving a source-spoofed UDP transmission, the client must reveal its identity to the server. As a result, man-in-the-middle (MITM) attacks launched during tunnel establishment enable an attacker to witness all traffic sent between the client and the server along the back-channel, including the client's address. In the event of a MITM attack, the client is *exposed*. The server, however, cannot be identified by eavesdropping on the back-channel alone. The address of the server is never revealed to the client and therefore preserves *absolute privacy* in this regard. Jondos along the back-channel can still attempt to identify the server using more sophisticated attacks.

Mantis provides the same *probable innocence* for servers and clients in the network, as clients receive in Crowds, with respect to guessing the initiator of a message. Without more sophisticated attacks and analysis techniques, the probability of guessing whether or not a jondo sending a message is the true initiator, is the probability that a malicious node appearing on the path is directly following of the originator. Mantis requires the probability of forwarding a service request to be

greater than 1/2 in order to maintain *probable innocence*.

6.4 Topology Building

It is in the interest of an adversary, when attempting to expose clients and servers, to first acquire a topology of the overlay P2P network. Even partial mappings can reveal routes used by two communicating parties. As path lengths are limited, the proximity of jondos can potentially implicate them in a communication. An accurate network topology can identify nodes that couldn't possibly be acting as server or client in a given communication. Obtaining an accurate overlay mapping and demonstrating that a large partition of the network is not involved in the communication decreases the anonymity of the true client and server, as they are now placed in a group of decreased size. Should the topology created be sufficiently extensive, it can be used to lower the degree of anonymity held by a server from *beyond suspicion*, where the server is no more likely than any other node to be a message initiator, to *probable* or *possible innocence* as the subset of possible nodes decreases, and in the most extreme cases may *expose* the server completely.

6.4.1 Topology Mapping By The Blender

The blender is in an ideal position for such a topology building attack. The information gathered by the blender about the number of connections, member list requests or any other data about jondos can potentially be used in mapping the network. It is therefore necessary to keep the blender from obtaining all but the minimal information needed to serve its purpose. It may appear beneficial to allow a blender to keep track of seemingly insignificant information, such as a jondo's connection count, for the purpose of fairness when returning connection lists. However, even given this minimal amount of information, the blender can infer new connections and disconnections based upon the simultaneous updating of connection count values for two individual jondos.

The blender is not only a threat due to the data it receives, but also due to the information it distributes. A malicious blender can selectively return connection lists containing only connection information for colluding peers or known invalid destinations. A trivial example of this attack is for a blender to return a connection list with only a single valid peer, or a valid address accompanied by many invalid connections. The blender can then infer that a connection has been made between the requestor and the single valid host.

There are a number of defenses available to prevent a blender from making accurate inferences and constructing a topology. First, jondos requesting a connection list should request many more connections than they actually need. In addition, they should only connect to a randomly chosen subset of the returned list. As the number of unused results increases, the probability of a blender inferring a jondo's newly formed connections decreases.

Second, list requests can be delegated to other jondos, anonymizing the list request process. This prevents a blender from inferring connections based on knowledge of the jondo that requested the list. By anonymizing the list request process and only attempting connections with a portion returned list, the blender is prevented from inferring either party in the newly formed connections.

Finally, utilizing multiple blenders when locating peers is essential to prevent many kinds of attack. It is obvious that as the number of queried blenders increases, the less information a single blender or group of colluding blenders is capable of gathering. As the number of queried blenders increases, a single blender will only be able to infer an extremely small percentage of the network topology with low accuracy if any at all.

6.4.2 Topology Mapping By Jondos

An additional topology building threat arises from jondos within the P2P network. When messages are broadcast, they are sent to each neighboring jondo, who could potentially infer the overlay topology based upon the order in which broadcast messages arrive. Messages arriving later will most likely arrive from jondos further away from the initiator. Jondos (or groups of colluding jon-

dos) attempting to map the network can connect to a very large number of peers in hope of increasing the probability of receiving duplicate broadcast messages.

This attack is unrealistic as the time needed to collect the necessary data for making the topological inferences can typically outlast the short lifespan of many nodes within the network. Furthermore, connections with nearly all nodes in the network are necessary for accurate inferences. In huge networks this becomes infeasible, as anything less than a direct connection to each node greatly decreases the probability of an accurate inference.

6.5 Colluding Jondos

As in the Sybil attack [7], where a single adversary is capable of impersonating many nodes at once, an adversary in Mantis can physically control many jondos simultaneously. The goal of the attack is to overwhelm a jondo's connection list with connections to colluding jondos. By doing so, the jondo is surrounded and any message originating from or arriving at the target jondo must pass through one of the colluding jondos. The target jondo is therefore *provably exposed*, as messages originating from the target jondo can be seen as not first being delivered by a colluder.

The combination of a malicious blender with colluding jondos can be very powerful. A blender can return to the target jondo only connection data for malicious nodes and notify only malicious jondos when the target is available to receive connections. Once the target jondo's connection list is full, the colluding jondos are as capable of identifying the target jondo as a message originator with probability of at least the ratio of good connections to malicious. The more malicious connections occupied by the target jondo's connection list, the more successful the attack.

To defend against this type of attack, a jondo should accept connections from a variety of blenders, so that when attacked by colluding jondos the ratio of good connections to malicious is as large as possible. By utilizing multiple blenders, and only accepting a portion of remote connections, a jondo can reduce this threat significantly and ultimately obtain a stronger level of anonymity

6.6 Eavesdropping

An eavesdropper is an adversary with the ability to oversee all traffic to or from one or more peers in the P2P community. Realistically, unless all peers exist within the same network and the eavesdropper is sufficiently powerful, the eavesdropper will only be capable of intercepting a small fraction of communications. Even so, merely witnessing a jondo's communications can reveal much information about its actions.

All connected jondos encrypt messages passed between them, so a local eavesdropper cannot determine the contents of a transmission by eavesdropping alone. However, Mantis's use of UDP for bulk data transfers between server and client will identify a given jondo as either the recipient (client) or initiator (server) of a UDP stream. Though the client or server has been *exposed* for operating as such, the contents of the communication remain undeterminable. An eavesdropper can also infer when a jondo is the originator of an encrypted message if the message is sent without first receiving a transmission from a neighboring jondo.

There are a number of methods for injecting irrelevant data into the network aimed at thwarting these kinds of analysis and once again obtaining *probable innocence*. For example, cover traffic can be sent periodically between neighboring jondos, making it unrealistic for an eavesdropper to determine whether messages have been initiated by the jondo or not. With respect to being the initiator or recipient of a UDP stream, Jondos can agree to forward entire bulk data transfers on behalf of another jondo, or alternatively, fake doing so.

It should be noted that as the number of precautionary hops increases, the closer Mantis's performance converges toward known protocols of lesser efficiency such as Crowds or MUTE. The eavesdropper is only capable of exposing a client or server as being such an entity, and is not capable of revealing the contents of data being served. Unless knowledge of a node *being* a client or server is

of importance, independent of content, the eavesdropping adversary does not pose any threat to our network.

6.7 Degradation of Anonymity

A number of results have shown that anonymity in multi-hop relay networks tends to degrade over time if nodes maintain long-term communication sessions. As observed in Crowds and fully quantified as a weakness in [19, 20]. At each path reconstruction, some malicious nodes may occupy positions on the newly-formed path. For each path created, the colluding adversaries identify the node closest to the beginning of the path. After a number of path reformations, the true path initiator will appear more than any other node as the closest to the path origin.

A similar attack may be launched against Mantis networks, aimed at identifying a server responding to a service request. Path reformation in Mantis is much more frequent than Crowds, as nodes are considered to have moderately short-lived connections. Furthermore, any node disconnecting from the network can alter the path by which a service request propagates. Even the fact that paths formed by service requests are non-deterministic adds to the quality of a degradation attack. After repeated searching and reconnecting to new nodes, the node responding to a service request most frequently can be considered either the responding server or the closest node within the tree to the initiator.

This attack is easily thwarted by caching service replies at each node. Once a service request has been made, all future service requests matching the same criteria specified will be responded to immediately by the closest node, not necessarily the actual responder.

In another attack aimed at degrading server anonymity in Mantis, a malicious jondo attempts to determine its distance from a particular node. Service requests are propagated only after a weighted coin toss. In a large network, where a path traversed by a service request is most likely the same each time, a malicious jondo can initiate the search repeatedly and record the number of responses. If the probability of rebroadcasting a service request is p and the malicious node has received k responses out of K requests, the node can solve the equation $k/K = p^{n-1}$ for n, where n is the number of hops away the responder is located. Fortunately, this attack is also prevented by caching search results at each node forwarding a reply. After the initial attack service request, responses received will be from the nearest node only. Furthermore, if the path is non-deterministic, as in a smaller tightly knit network, results will be different for each search regardless of caching.

6.8 Timing Analysis

An attacker may attempt to identify the server as the initiator of a message by analyzing round trip time (RTT) delays between relaying messages and receiving responses. When a request warranting a response is made, it is relayed hop-by-hop and responses are sent back along the reverse path, therefore RTT is larger for more distant servers. Hence, servers are *exposed* by immediately responding to a message. A savvy attacker could determine the number of hops between itself and the server, which is increasingly dangerous if the next-hop is the initiator of a response.

To counter these timing attacks and maintain *probable innocence*, a server must delay responses for some random interval. It should be noted, that the server must continue to use the same delay for all messages sent as more sophisticated attacks can identity the initiator after multiple queries if the delay fluctuates between responses.

6.9 UDP Source Spoofing

A variety of schemes have been proposed for tracing source-spoofed packets back to their source [15, 18, 1, 17]. However, these methods for trace back aimed at identifying a server are beyond the scope of this paper.

It is the prevalence of DoS attacks that pushes Internet Service Providers (ISPs) toward implementing stricter egress [17] filtering rules against home users and portions of their networks. These egress filters could eliminate Mantis's ability

to send source-spoofed UDP streams originating from anonymous servers. However, as most ISPs have not yet implemented widespread, strict egress filtering this does not appear to be a problem.

Less strict filtering rules are still acceptable by Mantis. Servers can determine their level of egress filtering present and spoof within the allowed address space. After creating a tunnel between client and server, the server attempts to spoof each octet of their address. If the connection is not successful, the server can then reveal a portion of its address, the first octet, and attempt the connection again. This process can continue stepping backwards and narrowing the allowed address space, until the level of egress filtering is determined.

It is known that wide scale egress filtering could effectively eliminate most DoS attacks based on source address spoofing, though this is unlikely to occur in the near future.

7 Future Work

Currently, Mantis has only been tested using a simplistic UDP reliability scheme, but many more efficient schemes for providing reliable UDP for bulk data transfers can be applied transparently for communication between clients and servers. Forward Error Correction and the use of Tornado Codes [2] as well as TCP friendly [23] protocols for implementing congestion control over UDP would reduce the overhead experienced from lost UDP packets and increase throughput over the backchannel significantly.

It may be possible to use Mantis as a network for seeking out anonymous web servers resistant to DoS attacks. The server does not need to reply with its true identity and can communicate with the client via anonymous UDP. It may be possible for clients to obtain a membership with an anonymous server and then search the network for some value agreed upon in advance. If the server wishes to revoke a membership, it can simply ignore these requests.

Another future improvement to Mantis could be to study the effects of using multiple backchannels for increasing throughput. It may be possible to utilize a number of these channels and therefore provide for the client's anonymity as well as the server's. If content being served within the network is distributed, it may be possible using hashes of this data, to receive bulk data transfers from more than one server simultaneously.

8 Summary

At the cost of revealing the client's identity, we have shown how a P2P network operating much like today's file-sharing networks can be maintained and provide for anonymous servers. Clients can search for content provided by servers, establish an encrypted communication tunnel with them, and finally perform a large data transfer, all while maintaining the server's anonymity. Bulk communication from server-to-client is sent via a source-spoofed UDP stream relieving relay nodes of needlessly forwarding large amounts of data on behalf of other users.

Acknowledgements

Thanks to Matt Green, Sam Small, Clay Shields, Andrei Serjantov, Annie Chen, James Riordin, Tadek Pietraszek and Malene Wang for their comments and suggestions and very helpful reviews.

References

[1] H. Burch and W. Cheswick. Tracing Anonymous Packets to Their Approximate Source. USENIX Proceedings of 2000.

[2] J. Byers, M. Luby, M: Mitzenmacher and A. Rege. A Digital Fountain Approach to Reliable Distribution of Bulk Data. Proceedings of ACM Sigcomm '98, Vancouver, Canada, September 1998.

[3] D. Chaum. Untraceable electronic mail, return addresses and digital pseudonyms. Communications of the ACM volume 24. February, 1981.

[4] Ian Clarke, Oskar Sandberg, Brandon Wiley, and Theodore W. Hong. Freenet: A Distributed Anonymous Information Storage and Retrieval System. In the Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in

Anonymity and Unobservability, July 2000, pages 46-66.

[5] George Danezis, Roger Dingledine, and Nick Mathewson. Mixminion: Design of a Type III Anonymous Remailer Protocol. In the Proceedings of the 2003 IEEE Symposium on Security and Privacy, May 2003.

[6] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The Second-Generation Onion Router. In the Proceedings of the 13th USENIX Security Symposium, forthcoming.

[7] John R. Douceur. The Sybil Attack. In Proc. of the IPTPS02 Workshop, Cambridge, MA (USA), March 2002.

[8] P. Ferguson and D. Senie. Network Ingress Filtering: Defeating Denial of Service Attacks which Employ IP Source Address Spoofing. RFC 2827.

[9] D. Goldschlag, M. Reed and P. Syverson. Onion Routing. Communications of the ACM volume 42, issue 2. Feburary, 1999.

[10] S. Kamara, D. Davis, R. Caudy, F. Monrose. SIMNET: An Extensible Platform for Simulating Network Attacks and Defenses. Submitted to ACSAC, 2004

[11] Lada A. Adamic, Rajan M. Lukose, Amit R. Puniyani and Bernardo A. Huberman. Search in Power-Law Networks. Phys. Rev. E, 64 46135 (2001).

[12] Brian Neil Levine, Clay Shields: Hordes: a Multicast-Based Protocol for Anonymity. Journal of Computer Security 10(3): 213-240 (2002)

[13] Ulf Mller, Lance Cottrell, Peter Palfrader, and Len Sassaman. Mixmaster Protocol Version 2 Draft, July 2003.

[14] M. Reiter and A. Rubin. Crowds: Anonymity for Web Transactions. ACM Transactions on Information System Security 1. April, 1998.

[15] S. Savage, D. Wetherall, A. Karlin and T. Anderson. Practical Network Support for IP Traceback. SIGCOMM. 2000.

[16] Vincent Scarlata, Brian Neil Levine, and Clay Shields. Responder Anonymity and Anonymous Peer-to-Peer File Sharing. in Proc. IEEE Intl. Conference on Network Protocols (ICNP) 2001. November 2001.

[17] A. Snoeren, C. Partridge, et al. Hash-Based IP Traceback. Proceedings of ACM SIGCOMM 2001.

[18] D. Song and A. Perrig. Advanced and Authenticated Marking Schemes for IP Traceback. INFOCOM 2001.

[19] Matt Wright, Micah Adler, Brian Neil Levine, and Clay Shields. Defending Anonymous Communication Against Passive Logging Attacks. IEEE Symposium on Security and Privacy, Oakland, CA. May 2003.

[20] Matt Wright, Micah Adler, Brian Neil Levine, and Clay Shields. An Analysis of the Degradation of Anonymous Protocols. In Proc. ISOC Network and Distributed System Security Symposium (NDSS 2002), February 2002.

[21] The Gnutella Protocol Specification v0.41 Document Revision 1.2.

[22] The Mute File Sharing System. Web page at `http://mute-net.sourceforge.net/`

[23] The TCP-Friendly Website. Pittsburg Supercomputing Center. Web page at `http://www.psc.edu/networking/tcp_friendly.html`

[24] The UDPP2P Project. Web page at `http://sourceforge.net/projects/udpP2P/`