Shubac: A Searchable P2P Network Utilizing Dynamic Paths for Client/Server Anonymity

Aharon Brodie, Cheng-Zhong Xu
Dept. of Electrical and Computer Engineering
Wayne State University, Detroit, MI 48202
asb@brodienet.com, czxu@wayne.edu

Weisong Shi
Department of Computer Science
Wayne State University, MI 48202
weisong@wayne.edu

Abstract

A general approach to achieve anonymity on P2P networks is to construct an indirect path between client and server for each data transfer. The indirection, together with randomness in the selection of intermediate nodes, provides a guarantee of anonymity to some extent. It, however, comes at the cost of a large communication overhead. In this paper, we present Shubac, a searchable, anonymous peer to peer (P2P) overlay network. It implements a flexible dynamic path approach that shrinks paths in size to reduce overhead and delays and meanwhile reconfigures paths dynamically throughout a communication to maintain a high level of privacy. This dynamic path approach enables Shubac to make a good tradeoff between anonymity and efficiency.

1 Introduction

Privacy on the Internet has been an important issue for many years. Censorship and copyright lawsuits are some of the reasons privacy has become a mainstream concern. Napster [10], being at the forefront of P2P technology, utilized a centralized network. All clients connected to a main server providing a listing of the files they shared. This made it simple to conduct a search for a file, querying the server and receiving a list of nodes sharing that file. This, however, was also the reason the Napster network was so easily shut down. By eliminating the main server, the network became headless. Nodes could not find or connect to others, nor could they search for files. This prompted next generation P2P programs such as Gnutella [9] to develop decentralized P2P networks. Nodes broadcast hello requests on the Internet until others are found, and communication such as searches is propagated from one neighboring node to another. This type of implementation does not contain central servers and cannot be easily shut down. Therefore, most P2P networks were later developed with this approach providing some variations for optimization. Since these networks cannot be shut down, companies wishing to restrict material from appearing on these networks have resorted to lawsuits. The approach is to intimidate users from sharing material that companies feel should be censored or falls under copyright infringement. This, among other reasons, is the reason anonymous P2P networks have become more popular.

Anonymous P2P networks allow users to share and retrieve files without revealing their identity. Their implementations vary from partial to full anonymity. Some forms of partial anonymity include client-side anonymity, where the protection is built towards nodes downloading files. Server-side anonymity, where nodes sharing files do so without having their identity revealed. And publishing anonymity, in which nodes anonymously upload files to the network which acts as a unified file storage [15]. A P2P network providing full anonymity hides the identity of nodes downloading files as well as of those sharing them.

Most implementations of anonymity on P2P networks rely on indirect communication. By transferring messages through intermediary nodes, networks hide the identity of one side of the communication from the other. Until recently, node resources were relatively low compared to the resources required by a fully anonymous network. Computers did not have the resources to maintain a large amount of concurrent connections. This is one of the reasons many anonymous P2P programs were developed with partial anonymity in order to maintain efficiency. Recently, node resources increased significantly, in terms of computing power, memory space, and network capacities. The average user does not fully utilize these resources allowing more intensive anonymity networks to be implemented. We introduce Shubac, a searchable P2P network offering anonymity to both clients and servers. Our implementation attempts to provide the best possible anonymity to all nodes on the network while using as little resources as possible.

Indirect communication requires intermediary nodes in

1-4244-0054-6/06/$20.00 ©2006 IEEE

an indirect path to remain active throughout the connection; if a node leaves the path becomes disconnected and the transfer fails. This requires nodes to dedicate large amounts of network resources for each transfer, which degrades network performance. At its core, Shubac was designed after the indirect communication approach. However, it has been modified and enhanced to be stable and more efficient. Shubac uses a small amount of resources from each node in order to maintain the higher amount of overall network resources needed. Shubac shares design attributes similar to other unstructured P2P programs. Nodes can enter and leave the network freely; there is no implementation of a coordinated log-on or log-off process against a server.

Freenet [4] provides anonymity, but does not support the function of searching. File searching on Shubac is based on a random walk protocol. Searches on Shubac play an important role, constructing a path to be used for a subsequent file transfer. One of the main differences between Shubac and other P2P networks is the file transfer protocol. Most P2P programs employ a direct file transfer between the client and the server, which although efficient, is lack of anonymity. There are several anonymous implementations, such as Mantis [1], Mutis [13], and Hordes [12], which use intermediary nodes for communication and as a result become resource intensive and unstable. Shubac improves upon this approach, using fewer resources and adding stability. Traditionally, each node on a path dedicates resources when forwarding communication. It is, however, unnecessary to use the resources of all the nodes in a path. We observe that P2P file transfers can often take from minutes to hours to complete, especially for large files. In light of this, we propose a *dynamic path* approach that reconfigures a path dynamically throughout a communication. That is, during the time paths dynamically change in size, shrinking to improve transfer speeds, and expanding to maintain a high level of anonymity. Additionally, we present further improvements to the algorithm as well as additional methods that enhance efficiency, stability and privacy.

The rest of the paper is organized as follows. Section 2 gives an overview of anonymous P2P systems. Section 3 presents a high level overview of the Shubac architecture. Section 4 gives the details of the file transfer protocol. Its efficiency is discussed in Section 5. Section 6 discusses a number of security issues related to the approach. Section 7 concludes this paper with remarks on future work.

2 Related Work

There has been significant work in P2P networks towards protecting user privacy. We present a brief overview of related work in this field including some of their strengths and weaknesses. Many implementations have been introduced to provide a transparent overlay network over which

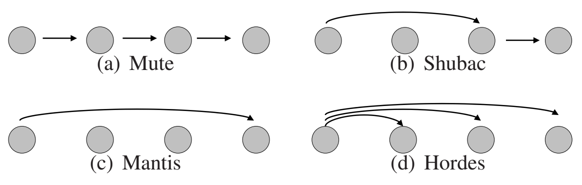

Figure 1: File transfer protocols in different anonymous P2P networks: Mute (a), Shubac (b), Mantis (c), and Hordes (d).

standard network communications can be anonymized. Although these vary, most construct a tunnel through other nodes on the network for the communication to travel. The final node in this tunnel contacts the server, which resides outside the overlay network.

Several implementations exist, among them are Tor [7], Tarzan [8] and Crowds [14]. They do not offer P2P features such as file sharing or searching, but mainly a means for existing communications to be transparently conducted anonymously. Anonymizing communications via tunneling is the most common implementation. This is also true for anonymous P2P networks with file sharing capabilities.

Freenet [4] is a P2P program providing anonymity. It implements the tunneling approach, but does not provide standard searching. Instead of searching for files to download on Freenet, users view lists of available files on Freesites, index Web sites hosted inside the Freenet network. Files are listed along with their virtually unique hashes. Users search the network using these hashes instead of by file names. There are also recent studies on mutual anonymity protocols based on trusted index servers [16]. Shubac's goal is to implement an anonymous *searchable* P2P network, which behaves similar to P2P networks like Gnutella. The majority of P2P programs implement searching based on full or partial file names since this is the most convenient approach to the user. Mute [13] is a file sharing searchable P2P network providing anonymity to clients and servers. Anonymity is accomplished by tunneling client and server communication through other nodes on the network. Searching for files on the network is accomplished by passing the search from one neighboring node to another. The path created by the search is later used to tunnel further communication between the client and server, this communication mainly being the file transfer. Figure 1(a) shows the tunneled transfer, in comparison with the protocol of Shubac in Figure 1(b). The bulk of communication on P2P networks is file transfers, and one of the problems of this approach is the large transfer delay and network overhead incurred by it.

Hordes [12] and Mantis [1] are examples of P2P programs that attempt to overcome the limitations of the tunneling approach. Hordes provides client anonymity by instructing the server to send data in a multicast message, as shown in Figure 1(d). A file is sent to multiple nodes, one of which is the true recipient. This effectively creates a direct communication from the server to the client without explicitly revealing the client's identity, providing an anonymous and more stable solution. A few problems with this approach are evident, large amounts of bandwidth are required by multicasting data to numerous nodes. Client privacy degrades, unlike the tunneling approach where a client is almost definitely unknown, here the list of suspected clients is narrowed to those receiving the multicast. And finally, servers transmit data directly to other nodes eliminating their privacy. Mantis, one of the more recent P2P programs released, has specifically targeted the negative impacts tunneling has on the network. Searches are constructed in much the same way as Mute and Shubac, and are the foundation for communication tunnels. These tunnels, however, are used to communicate control data and not file transfers. The file transfers take place via spoofed source UDP streams from servers to the clients, as shown in Figure 1(c). This alleviates other nodes on the network from spending resources on tunneling, and at the same time maintains server anonymity despite the direct communication. The main drawback of this approach is the lack of client anonymity. Shubac's implementation maintains both client and server anonymity via indirect communication. Communication paths are flexible, dynamically changing in size. Shrinking them reduces overhead and the amount of nodes involved, while expanding them maintains anonymous communication between clients and servers.

3 Shubac Architecture

Our goal is to design an anonymous P2P system similar in functionality to traditional P2P programs. That is, the network should allow for searching of files; the identity of nodes on the network remains hidden; implementation of the anonymity feature should be transparent to end users; the efficiency of the network should be maximized by reducing the overhead incurred by transfers.

Shubac comprises three key components: *tunneling*, *searching*, and *routing*. Shubac uses indirect communication by creating paths composed of arbitrary nodes. Communication travels this path, creating a tunnel between client and server. In Shubac, searches propagate the network as in traditional P2P programs. Shubac implements a search protocol that is effective but not resource intensive. Nodes on Shubac store path information in routing tables. These are used for forwarding communication and to correlate between searches and their corresponding paths.

3.1 Tunneling

Nodes on Shubac are near sighted, able to see and communicate with adjacent nodes. Distant clients and servers communicate indirectly, propagating messages from one neighbor to the other until the destination is reached. A node's knowledge of the network is limited to its neighbors' identities. It lacks information about a path it participates in such as client and server identities or path length. This is achieved by constructing initial messages, and forwarded messages similarly. Ensuring nodes cannot differentiate between neighbors forwarding messages or generating them.

3.1.1 Path Stability

The indirect communication approach protects node privacy but is not efficient on a P2P network. The main issue is path instability, nodes exit the network breaking paths they were connected to. A node leaving a path causes its neighbors, who do not know each each other's identity, unable to forward communication. Churn [3], the rate at which nodes exit and leave a P2P network, has been shown to be high and causes paths to break at a high rate. Another issue is the desired length of a path. Long paths are composed of a large number of nodes, and are initially advantageous as will be discussed later. However, they are more susceptible to churn, can affect transfer speeds, and introduce excessive overhead to the network. Since the bulk of communication on P2P networks is file transfers, this can present significant delays.

Shubac addresses these issues by introducing a *dynamic path* method, which shrinks paths length as communication progresses. An initial path consists of many nodes and progressively shrinks to a length more efficient for file transfers. Paths shrink as nodes exit by connecting the neighbors positioned before and after them to each other. Additionally, this allows nodes to self regulate the amount of communication and overhead to partake in.

3.1.2 Suspicion

Tunneling communication hides the identity of clients and servers but does not eliminate suspicion. Nodes remaining on a path for the duration of a file transfer can be suspected by their neighbors to be the client or server. Shubac addresses this issue by maintaining a dynamic path that continuously grows and shrinks. Each node is able to leave a path or insert another node into it. Nodes are only able to insert their neighbors since they do not know the identity of non-neighboring nodes. Inserted nodes may be placed before, after, or instead of the inserting node's position. Each node makes an independent decision whether to exit a path or insert a node into it. This decision is based on a random factor as well as the node's resources. The neighbor is asked to join is chosen randomly as well. These abilities reduce the time a node remains connected to the same neighbor. A node cannot differentiate whether its neighbor has exited a path or inserted a node since both actions cause the node to communicate with a new one. Nodes cannot successfully suspect neighbors that do not appear to remain in the path for the duration of the communication.

3.2 Searching

A node searching for files on Shubac sends a search to its neighbors where it continues to propagate from one neighbor to the next. Nodes receiving a search process it, store its routing information, and forward it onward to their neighbors. Nodes compare a search's file pattern to files they share, returning a list of matching files to the searching node. Search replies are sent back to the neighbor the search was received from which in turn sends the reply to the node it received the search from. The reply is sent back on the search's path until the searching node is reached. After receiving replies, a searching node sends a request for a matching file. The request is forwarded on the path until it reaches the node sharing the requested file. The file is then transferred on the path back to the searching node.

Each search sent on Shubac builds a path as it propagates the network. Maintaining routing information about multiple paths, and forwarding their communication, presents a burden on nodes and the network. A search protocol should reach as many nodes as possible to increase the chance of finding a file, while generating as little traffic as possible to maintain an efficient and scalable network. A flood-based approach, as used in Gnutella, would ensure the search reaches every node but would overwhelm Shubac's network with the amount of paths created. We therefore implemented a multiple biased random walk protocol. Searches have a high potential of returning results while generating relatively low network traffic.

Using the random walk approach nodes send and forward searches to one random neighbor. Randomly selecting neighbors improves the chances of subsequent searches to reach new nodes. This approach, however, can create path loops, a situation where nodes receive a search more than once. Path loops cause nodes to send duplicate search results and assume multiple positions in a path. Introducing a biased approach enforces nodes to reject searches they have already received based on their unique IDs. Nodes receiving a rejection will attempt to forward a search to a different neighbor. Although this approach consumes a small amount of network resources, it reaches a small number of nodes since it travels a single path.

For a more effective search, each node on Shubac sends out multiple searches simultaneously. Each is sent through a different neighbor, reaching a larger number of nodes. This has been shown to provide adequate results, and generate little traffic compared to flooding, and remain scalable [3]. Shubac enhances this approach allowing nodes to download from multiple sources simultaneously. This decreases transfer time and distributes the transfer load to multiple nodes. To prevent searches from endlessly traveling the network, a time to live (TTL) value expires a search after it passes through a set amount of nodes. Initially reaching a large number of nodes yields higher search results. However, during file transfers, long paths are more likely to break and decrease transfer speeds. A hard coded TTL cannot be implemented since a client's location will be known and its identity can be discovered by backtracking the path. Therefore, for every search created, a semi-random TTL is calculated in the range of a hard coded value. This provides an adequate path length to be used without the client's identity being compromised.

3.3 Routing

Each node participating in a path forwards communication from one neighbor to another, and maintains this information in its routing table. Every entry in a routing table consists of three fields:

- `SearchID`, which is a unique identifier created for a search and used to identify all communication on that searchs path.
- `SourceID`, which is the identity, the IP address and port, of the neighbor the search was received from. This node is effectively the previous hop in the path.
- `DestinationID`, which is the identity of the neighbor the search was forwarded to. This node becomes the next hop in the path.

Routing tables are modified throughout a communication in various ways. A node receiving a search creates an entry for it in the routing table and deletes it when the path is invalid or no longer used. Changes to a path during a transfer, such as nodes exiting or inserting others, cause nodes to modify the paths entry by updating the identity of the previous hop or the next one.

Routing tables are crucial for indirect communication to function, as each node on a path is required to forward communication in both ways, between the client and the server. If a node on a path inserts an incorrect value in the paths entry, it would not know the identity of its adjacent hop and would not be able to forward path communication. Maintaining local routing tables has many advantages over a global approach:

1. Nodes only know the identity of their adjacent neighbors in a path and do not make this information public.

2. Local routing tables allow for quick responses to changes in the network. Problems with a node are immediately detected by its neighbors allowing searches to be redirected and paths to be invalidated quickly.
3. Changes on the network are kept local, not affecting distant nodes.
4. Overhead required for network changes is kept to a minimum, providing a dynamic and scalable network.

Communication on Shubac is done via *virtual cut through* (VCT). With this implementation nodes forward packets as soon as they are received. This results in a near constant flow of file segments from server to client. VCT is significantly faster than store-and-forward (SF), in which messages, such as file transfers, are queued until they are fully received before being forwarded onward in the path. Transfers using VCT assume data flows quickly through a node. Path changes can cause delays until routing tables are updated. To minimize transfer delays, while nodes change the path, such as inserting a new node, they continue forwarding messages to their existing neighbor. Messages will be redirected to the new node only after it has updated its routing table and is ready to transfer. Since delays are inevitable and newly inserted nodes can unexpectedly create bottlenecks due to low network resources, nodes on Shubac can send congestion control messages. These messages are sent back against the direction of the transfer until they reach the end of the path. They contain requests to slow down the transfer rate in order to prevent network buffers from overflowing, waste node resources, and possibly discard packets requiring them to be resent. Congestion messages may also be sent to inform nodes that transfer rates can be increased after a bottleneck has passed.

4 Dynamic Path Reconfiguration

Shubac organizes its communication messages in a format of four fields:

- `SrcNode` field contains the identity of the sender, and the `DestNode` field contains the identity of the recipient.
- `SearchID` field contains a virtually unique identifier for communication related to each search.
- `Command` field indicates the purpose of the communication. The possible messages a node may receive include `searches`, `search replies`, `search rejection`, `file requests`, `file transfers`, `path updates`, and `congestion control`.
- `FileData` field contains data relevant for the communication. This includes search patterns for searches, lists of matching files for search replies, filenames or their contents for file requests and subsequent transfers, and path update information.

Paths on Shubac are dynamic, throughout a communication nodes continuously enter or leave them. A node's decision to leave a path is based on a random decision and its resource usage. If a node participates in a large number of paths, forwarding communication for each of them can use its available bandwidth. In such a case, a node will randomly select a path from its routing table to exit. Each node running on Shubac begins a countdown timer once it starts participating in paths. The countdown is set by a random function returning a value between one and two minutes. Once the timer finishes its countdown the node executes a random function to determine whether to exit a path or not. If it decides to do so, a random path is chosen from the routing table. The node proceeds to inform its two adjacent hops in the path that future communication should be forwarded directly between them and not through it.

A node's decision to insert another node into the path is based on a random function as well. This function is executed after a countdown timer, which is also initialized when a node begins participating in paths. The value a timer begins counting from insertions relies on a random function and varies between three and four minutes. Insertions occur less often than removals since we wish the path to shrink over time. When performing an insertion, a node randomly selects a path it is a part of and a neighbor, then invites the neighbor to join the path. The neighbor accepts the invitation if it has available resources and it is not already participating in that path. This is verified by looking up the invitation's `searchID` in the routing table. If it is not found, the neighbor is not participating in the path and may join it. Further steps include inserting an entry into the invited node's routing table with the appropriate neighbors, and updating the inviting node's routing table along with its neighbor's that they will be forwarding communication to a new node. When the decisions to exit a path and insert a node occur at the same time, a path replacement takes place. A replacement consists of a node leaving a path while inserting a neighbor into its position.

The countdown timers used to launch the decision functions are reinitialized and run continuously until a node is no longer participating in any paths. A node abruptly exiting a path is called a dying node since it does not perform a clean exit, notifying the neighbors in its paths. A dying node's neighbors unsuccessfully forward communication to it or time out while expecting communication from it. When this occurs, its neighbors send a message in the other direction of the path notifying of a broken path. Nodes on the paths receiving this message forward it onward and remove the path's entry from the routing table.

To maintain an updated routing table, each node maintains an expiration timer of five minutes for each path. Every time a communication for that path is received, the timer is reinitialized. If messages are not received for a path within five minutes, the path is considered obsolete and is removed from the node's routing table.

As mentioned, if an inserted node has low network resources, or if an existing node unexpectedly yields a reduced amount of network resources, a bottleneck can occur since messages will arrive at a greater rate than those leaving. In such a scenario, the node constructs a message with the amount of throughput it can process and sends it to the previous hop in the path, the neighbor that is forwarding at a high rate. The message is transferred back along the path from one node to another until the server is reached. The server then lowers its transfer rate to the amount requested in the message. Once the node causing the bottleneck exits the network or achieves a higher throughput, it can send a message back along the path instructing the server to increase the transfer rate.

5 Performance Analysis

Most anonymous P2P networks achieve anonymity by indirect communication. We compare the differences between file transfers on MUTE, Mantis, and Shubac in terms of data transfer throughput. In addition, we compare the protocols in terms of the amount of network resources used by an intermediary node on a path, a server, and the total amount used by a transfer.

5.1 Network Resources

In MUTE, over the course of a transfer, each node on a path receives the complete file and forwards it onward. Its network resources used equal twice the file size since paths in MUTE are static and nodes remain in their paths until transfers complete. Mantis' implementation of direct file transfers results in nearly no committed resources for intermediary nodes since they only transfer control communication. Nodes on Shubac initially perform similarly to those on MUTE. However, on average, the amount of used resources is significantly lower. Downloading from multiple servers concurrently, multiple paths are used, each transferring only a fraction of the file. Furthermore, shrinking paths and additional path dynamics reduce the time a node participates in a path. Since a node does not need to participate in a path for the duration of the transfer, the amount of resources it commits is significantly lower than it would commit on MUTE.

Servers in Mantis and MUTE are expected to forward files in their entirety, the file size being the amount of resources used. Shubac requires a smaller amount by distributing the download, retrieving segments of a file from additional servers. While finding additional servers is not guaranteed, it is common for popular content. The overall amount of resources used a file transfer on MUTE is proportional to the path length and file size. Each node on a path, except for the client and server receive and forward the entire file throughout a transfer. Mantis on the other hand uses significantly less resources since the file is transferred only once between server and client . While on Shubac, the amount of resources is initially high as in MUTE, it progressively shrinks along with the path length.

5.2 Node Throughput

We compared the performance of Shubac to traffic similar to MUTE and Mantis. Mantis is the optimal approach since transfers are done directly. Therefore, we simulated it with direct transfers on Shubac. MUTE performs similarly to paths on Shubac that have just been constructed and therefore have not shrunk. It is the non-optimized approach, and was simulated by performing transfers on static paths on Shubac. We compared these approaches on a Local Area Network where the environment is the same for each, and external factors are not an issue. It should be noted we attempted to test Shubac on PlanetLab but did not notice benefits since we were not able to emulate the amount of nodes as on a real P2P network.

We transferred files of various sizes across the network, directly as in Mantis, on a static path as in MUTE, and on Shubac' dynamic path. Paths used for transfers initially consisted of six nodes but nodes on Shubac left or entered the path throughout the transfer. Figure 2 shows Mantis and MUTE with consistent throughput throughout the transfers. During transfers of smaller files, Shubac's throughput is similar to that of MUTE. However, with transfers of larger files the transfer time increases giving the path more time to shrink. The 10MB file transfer on Shubac was too short to allow for any path dynamics, therefore, the path length was of six nodes throughout the transfer. Two nodes exited during the 100MB transfer resulting in a four node path from approximately the middle of the transfer. The path of the 200MB transfer shrunk by three nodes leaving at almost identical intervals. Finally, while transferring the 500MB file, three nodes entered the path and five nodes left it. As can be seen from the graph, transfers on Shubac progressively increase speed significantly improving upon the static path approach.

5.3 Path Dynamics

Figure 3 represents a file transfer conducted on Shubac, transferring a 200MB file across a six node path. During the transfer three nodes left the path. This is the reason

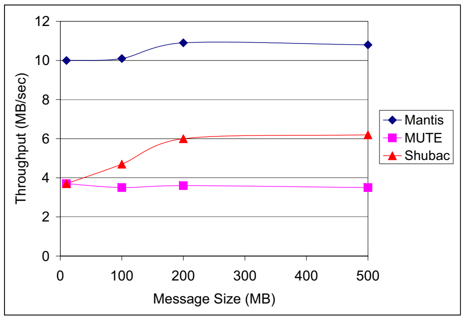

Figure 2: Throughput of data tranfer in different anonymous P2P networks.

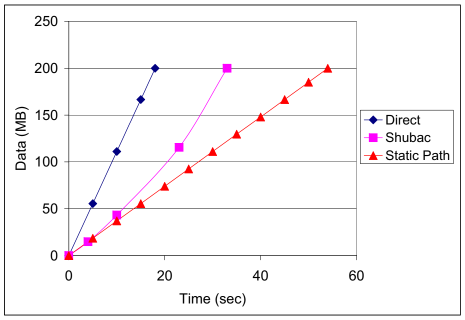

Figure 3: Profile of data transfers in different anonymous P2P networks

for the curve and a decrease in transfer time. This transfer took place on a local network and lasted approximately thirty three seconds. Four seconds after the transfer began the first node exited the path, a second node left after ten seconds, and the third left after twenty three seconds. The improvement is noticeable compared to the transfer on a static six node path that took almost twice as long.

6 Security Considerations

Attacks on P2P networks range from revealing node identities to manipulating and disrupting communications. The majority of P2P networks lack authentication and verification of participating nodes. Communication is done with a certain level of trust, as nodes are unable to validate data they receive from others. Therefore, most of the current attacks cannot be prevented, but their effects can be minimized.

6.1 Identity Exposure

Timing attacks. A common attack at discovering a client or server of a communication is a timing attack. A malicious node participating in a tunnel measures the time it takes a server or client to respond to a message. An adjacent neighbor can be exposed to be a client or server if responses are immediate. To counter this attack nodes on Shubac insert random delays before sending messages, creating timing inconsistencies. A similar attack exists for tunneled communication where nodes can suspect neighbors of being servers or clients if they remain on a path for the duration of the communication. Shubac reduces suspicion by introducing dynamic paths in which nodes can exit or insert new nodes into their paths. A node leaving a path gives both of its neighbors their IP addresses, effectively making them adjacent neighbors in the path. A node inserting another into a path performs a similar action by notifying its previous neighbor of the new node's IP address. In both scenarios, a neighboring node begins communicating with a new node, not knowing the status of its previous neighbor.

Network analysis. Timing attacks [11] provide suspicions, but cannot certainly identify a client or a server. Analyzing network communications via router and server logs allows backtracking [2] a communication to its source, reconstructing the path. This is a resource intensive scenario that would require access to network logs of Internet Service Providers (ISP) around the world due to the global usage of P2P networks. While difficult to accomplish, this would provide concrete evidence to a node's actions on the network. Shubac attempts to hide communication by allowing nodes to randomly send meaningless messages to one another. Nodes sending numerous messages to their neighbors create multiple dummy paths ending at random nodes. A possible way to overcome this is to record all packets on the network to differentiate between real and dummy messages. However, unless this occurs on a local network, this is an unrealistic scenario. Paths on Shubac are dynamic, created based on a random walk protocol. An eavesdropping node would need access to the Internet backbone of every ISP the path crosses. Furthermore, under normal circumstances, ISPs do not record full packet payloads do to the volume of traffic on the Internet. Therefore, recording communication and backtracking it would need to be done in real time, as the path remains active.

6.2 Active attacks

A common attack aimed at disrupting communication is the Denial-of-Service (DoS) [5] attack, overwhelming a victim with messages. Nodes on Shubac throttle the amount of messages they accept from neighbors, preventing a flood on the overlay network. This prevents more sophisticated attacks such as overwhelming a node with false identities to include as neighbors, and prevents bottlenecks during file transfers.

Attacks such as Man in the Middle (MitM) and Sybil [6] are between the most sophisticated attacks present, allowing nodes to control a communication. In the MitM attack, a malicious node participating in a path can monitor and modify messages passing between the server and client. Since communication is encrypted the attacker would need to be a part of the path since it was formed in order to intercept the encryption keys passed between the client and server. After this, the attacker is able to decrypt messages and modify them, before re-encrypting them and forwarding them. This allows an attacker to inject malicious data into the file transfer such as virii. Since this attack is virtually invisible to the client and server, it is nearly impossible to detect. Clients on Shubac are able to detect this attack when downloading files from multiple nodes. The integrity of data downloaded from a server is verified using an MD5 checksum with another server, discarding all invalid data.

Sybil, a similar attack, involves a node assuming the identity of other nodes. Common attacks include masquerading as a server or multiple nodes in a path. Since the identity of clients and servers is hidden, the extent of these attacks includes disrupting communications and modifying the data transferred. Both scenarios are not critical to the victim since malicious data can be detected when using a distributed download, and disrupted communications only cause a new search and transfer to take place.

7 Concluding Remarks

Anonymity preserving P2P programs have either been inefficient or have relaxed anonymity in order to maintain efficient file transfers. Shubac is a searchable P2P network that provides anonymity to both clients and servers. It introduces a means for efficient and stable file transfers without the need to reveal the identity of neither clients nor servers. Anonymity is achieved by transferring files through indirect paths on the network. Efficiency is achieved by path shrinkage, by allowing nodes to exit paths. This increases throughput and reduces the resources used by forwarding nodes. Maintaining efficiency throughout a transfer is done with path dynamics. Nodes are able to insert others into the path or replace their position with a new node. This provides a constantly evolving path, in which it is difficult to differentiate between a forwarding node, a server, or a client. Further improvements were introduced to improve transfer rates and stability, such as distributed downloads.

Acknowledgement

This research was supported in part by U.S. NSF grant ACI-0203592 and NASA grant 03-OBPR-01-0049.

References

[1] S. Bono, C. Soghoian, F. Monrose, Mantis: A Lightweight, Server-anonymity preserving, searchable P2P network. Technical Report TR-2004-01-B, Information Security Institute, Johns Hopkins University, June 2004.

[2] H. Burch and W. Cheswick, Tracing anonymous packets to their approximate source, *Proc. of USENIX Large Installation Systems Administration Conf. (LISA)'00*, 2000, pages 319–327.

[3] Y. Chawathe, S. Ratnasamy, L. Breslau, S. Shenker, Making Gnutella-like P2P systems scalable. *Proc. of ACM SIGCOMM*, 2003.

[4] I. Clarke, O. Sandberg, B. Wiley, T. W. Hong, Freenet: a distributed anonymous information storage and retrieval system, *Proc. of Designing Privacy Enhancing Technologies*, July 2000, pages 46-66.

[5] N. Daswani, H. Garcia-Molina, Queryflood DoS attacks in gnutella, *Proc. of ACM Conf. on Computer and Communications Security*, 2002.

[6] J. Douceur, The Sybil attack, *Proc. of the IPTPS02*, March 2002.

[7] R. Dingledine, N. Mathewson, P. Syverson, Tor: The second-generation onion router, *Proc. of Usenix Security Symposium*, 2004

[8] M. Freedman, R. Morris, Tarzan: A peer-to-peer anonymizing network, *Proc. of the 9th ACM Conf. on Computer and Communications Security*, November 2002.

[9] Gnutella homepage, `www.gnutella.com`.

[10] K. T. Greenfeld, Meet the Napster, *TIME Magazine*, October 2000.

[11] B.N. Levine, M. Reiter, C. Wang, M. Wright, Timing attacks in low-latency mix systems, *Proc. of 2004 Financial Cryptography*, February 2004.

[12] B. Levine, C. Shields, Hordes: A multicast-based protocol for anonymity. *Journal of Computer Security*, Vol. 10(3), 2002, pages 213-240.

[13] Mute, http://mute-net.sourceforge.net.

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

[15] A. Rowstron and P. Druschel. Storage management and caching in PAST, a large scale, persistent peer-topeer storage utility, *Proceedings of ACM SOSP'01*, 2001, pages 188-201.

[16] L. Xiao, Z. Xu, and X. Zhang. Mutual anonymity protocols for hybrid peer-to-peer systems, *Proc. of Int. Conf. on Distributed Computing Systems*, 2003.