2011

Evidence Collection for Forensic Investigation in Peer to Peer Systems

Sai Giri Teja Myneedu
Iowa State University

Evidence Collection For Forensic Investigation

In Peer to Peer Systems

by

Sai Giri Teja Myneedu

A thesis submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Major: Information Assurance, Computer Engineering

Program of Study Committee:

Yong Guan, Major Professor

Manimaran Govindarasu

Thomas E. Daniels

Iowa State University

Ames, Iowa

2011

Table of Contents

List Of Tables

List Of Figures

Acknowledgements

I would like to take this opportunity to express my thanks to those who helped me with various aspects of conducting research and the writing of this thesis. First and foremost, Dr. Yong Guan for his guidance in helping me choosing the topic for this work, his patience and support throughout this research and the writing of this thesis. I would also like to thank my committee members: Dr. Manimaran, for his guidance as a mentor and as an advisor for Sankalp and Dr. Tom Daniels for his great teaching style and introducing me to Cyber Defense Competition at Iowa State.

To my family and friends whose support makes me who I am - no words can express my gratitude for you. However I would like to let you know that your support means a lot to me.

Abstract

Peer to Peer(P2P) file sharing networks are amongst the best free sources of information on the internet. Voluntary participation and lack of control makes them a very attractive option to share data anonymously. However a small group of people take advantage of the freedom provided by these networks and share content that is prohibited by law. Apart from copyrighted content, there are cases where people share files related to Child Pornography which is a criminal offense. Law enforcement attempts to track down these offenders by obtaining a court order for search and seizure of computers at a suspect location. These seized computers are forensically examined using storage and memory-forensics tools. However before the search warrant is issued strong evidence must be presented to provide a reason for suspiscion. Deficient investigation in the intial stages might lead to misidentification of the source and steer the investigation in a wrong direction.

Initial evidence collection on peer to peer file sharing networks is a challenge due to the lack of a central point of control and highly dynamic nature of the networks. The goal of this work is to create a working prototype of an initial evidence collection tool for forensics in P2P networks. The prototype is based on the idea that P2P networks could be monitored by introducing modified peer nodes onto the network for a certain time period and recording relevant information about nodes that possess criminally offensive content. Logging information sent by a suspicious node along with timestamps and unique identification information would provide a strong, verifiable initial evidence. This work presents one such working prototype in alignment with the goals stated above.

1 Introduction

This chapter describes the motivation for this work. It also gives a brief introduction to the proposed method and discusses the organization of the rest of this thesis.

1.1 Motivation

Peer-to-Peer (P2P) traffic was accounted for 44% of the entire internet traffic according to a study published by Sandvine in 2008 (11). Even with the advent of high-quality streaming media, P2P traffic has maintained very strong and steady state with about 30% of traffic share in Europe and 20% in North America. The quick emergence and shutdown of 'Napster' which caused a huge uproar in 2001 caused several more advanced P2P file-sharing networks to spawn. Due their decentralization, flexibility, scalability and voluntary participation, peer-to-peer file-sharing networks have been amongst the best free information sources on the Internet. Hence their popularity in unabated even with the emergence of newer technologies.

The number and popularity of peer-to-peer (P2P) file-sharing networks has been increasing since the appearance of the very first P2P file-sharing application (Napster) in 2001. However popularity comes with a price. In case of P2P systems their primary advantage of 'freedom' compared to other more restricted forms of information sharing became a bane in certain cases. Along with exchange of important information, many illegal activities are perpetrated on P2P network. Illegal file sharing was the cause of the shutdown of the first widely popular P2P application. However with the emergence of other more sophisticated P2P networks the illegal activities continued on these other networks. Sharing of illegal material on these networks has been on a constant rise.

Copyright Infringement causes billions of dollars losses every year to media industry. Several articles in the news report notices related to prosecutions under 'Digital Millennium Copyrights Act'. News articles also report monitoring activities from 'Recording Industry Artists

association'(RIAA) and 'Motion Picture Artists Association'(MPAA). These organizations are powerful entities that are fighting seriously against billions of dollars losses incurred every year due to illegal file sharing in P2P networks. RIAA (1), on its website indicates that 'in the decade since peer-to-peer (p2p) file-sharing site Napster emerged in 1999, music sales in the U.S. have dropped 47 percent, from $14.6 billion to $7.7 billion'. It also cites and NPD report that indicates only 37 percent of music acquired by U.S. consumers in 2009 was paid for. While sharing of copyrighted material causes several industries a loss of billions of dollars, these are civil infractions where the loss is only monetary. Unfortunately the abuse of P2P networks also includes an apalling number of criminal activies that cause potential human abuse.

Possession and distribution of child pornography(CP) is one of the pressing criminal activities carried out on P2P network. Government Accountability Agency(GAO) submitted a report on CP in 2003. The report indicated that in one search using 12 keywords known to be associated with CP, GAO identified that about 42 percent results were associated with CP. Mehta et al., (5) monitored a website 'Gnutellameter' which captures data exchanged in Gnutella and provides summaries of key words most commonly entered by users. They suggest that CP features amongst the most commonly searched for files on Gnutella. Grabowski (6) had noted that in February 2003, Palisade Systems collected 22 million requests and queries conducted on Gnutella over a three week period and randomly selected 40000 of these. They found that 42 % of all requests monitored were for adult or CP. The presence of such content, and its very easy access, make the current situation particularly worrying for P2P users, in particular children. Studies by Waters et al. (19),Taylor et al. (20) and Quayle et al. (21) reveal more depressing facts linking CP to pedophilia. The studies include other human psychological factors and observe that easy and/or unwanted access to unwanted content may create the users interest for such content. Hence monitoring of these networks by agencies has already been instituted.

Law Enforcement needs to be extremely cautious in investigating criminal cases related to P2P file sharing. Shoddy investigation often yields insufficient or misleading evidence. Verification of evidence at every level not only builds a strong case but also ensures that innocent people are not falsely implicated due to deficiencies in investigation. Interhack's investiga-

tion (2) represents an example of a case where a teenager was falsely implicated initially, for possession of CP. But an external investigator used user logon and file access timestamps on the operating system to prove that the actual files found on the system were downloaded by malware on system. Timestamps and other such information are useful in investigating these techniques. Tools like P2P Marshall (14) are used by Law Enforcement to forensically examine the confiscated storage devices. However before any computer is is confiscated, it is very important to zero-in on the right IP-address, computer and physical location before a subpoena is filed and a subsequent search warrant is issued based on the initial evidence collected. Hence the initial data and evidence collection leads to a significant impact on the direction in which the case proceeds. Data collection methods and tools that provide investigative agencies with data that identifies the source go a long way in proper handling of any criminal case. This data could be used after confiscation to determine the validity of the case of posession of criminal content.

This work is motivated by the challenges faced in investigation of cases related to illegal distribution of criminal content on the P2P networks. The consequences of availability of such material are very strong and hence this work makes an effort to aid the investigation of criminal content. The importance of conducting an investigation in a complete and technically sound manner so that false positives are eliminated is well-understood. Hence an attempt it made to create a system that overcomes the technical challenges in monitoring P2P.

1.2 Proposed Method

The goal of this work is to demonstrate that a reliable monitoring system could be built for initial data collection on a P2P network. For this purpose different types of P2P networks were considered and their characteristics were analyzed. Based on criteria like topology and search mechanism different types of P2P systems could be classified in a few common categories. Possible data collection methods for each of these categories were examined for the purpose of implementation of a prototype. The data collection method could be extended to other similar P2P networks to provide initial evidence to obtain a search warrant with a stronger confidence level.

Note: This work does not attempt to identify the initial uploader of criminal content or conclusively prove when a file was obtained by a node. It attempts to identify the nodes in possession of a particular illegal file.

1.3 Thesis Organization

Chapter 2 of this thesis provides a detailed introduction of P2P networks and describes the P2P protocols relevant to this work in detail. Chapter 3 presents the legal and technical challenges involved in implementing a solution for initial evidence. Chapter 4 presents a review of work related to monitoring of P2P systems. Chapter 5 describes the design and implementation for a prototype for initial evidence collection in a chosen P2P network. A few experiments were performed to evaluate the different stages of the proposed evidence collection methodology. Observations from the experiments are presented in Chapter 6. Chapter 7 summarizes the work of this thesis and identifies the directions in which this work could be improved in the future.

2 Background

This chapter introduces some concepts and terminology necessary to understand the contents of this thesis. Section 2.1 presents a brief introduction of peer to peer(P2P) networks and characteristics of P2P Networks. Section 2.2 presents a classification of P2P Networks based on the type of network topology, protocol class and hierarchy of nodes. Section 2.3 briefly describes the applications of P2P systems. Section 2.4 introduces the protocols Gnutella and Gnutella2 which are the file sharing protocols for which a forensic investigation monitoring system is proposed in this thesis.

2.1 P2P Systems Overview

P2P systems are distributed systems that formed by a set of interconnected nodes where each node has equal capability. Formation and maintenance of these networks is based on a well defined communication protocols used to by nodes to join the networks and exchange information. P2P systems are overlay networks built at application layer. All the nodes of a P2P network run software that implements the communication protocol. Peers facilitate exchange of information at the application layer which helps in network maintenance and has several different applications.

2.2 Classification of P2P Networks

Factors like topology and communication protocol are inter-related in P2P systems. Based on these two factors the P2P Networks and taking into consideration the primary purpose of information exchange - P2P networks could be classified into the following three categories:

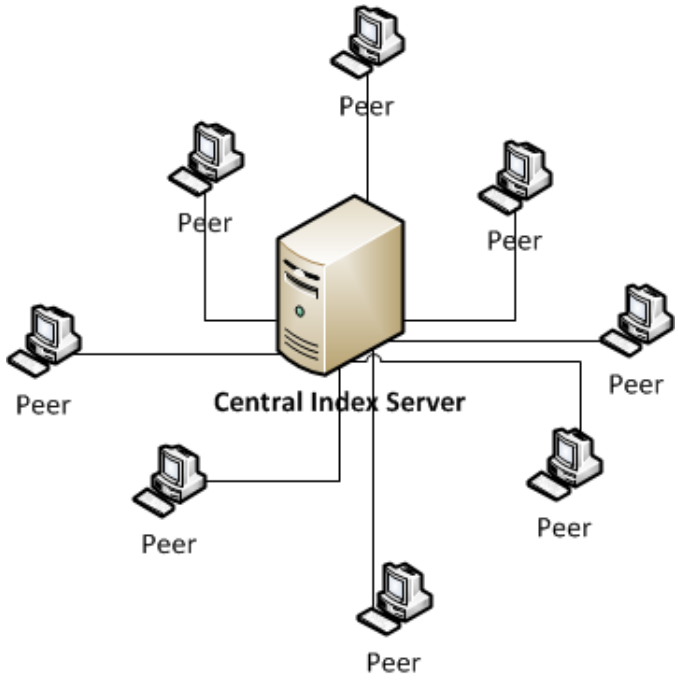
1. 1. Centralized Networks

2. 2. Completely Decentralized Networks

3. 3. Semi Centralized Networks

2.2.1 Centralized Networks

One central index/directory server maintains a list of all active/live nodes on the networks. Nodes that wish to join the network need to send information about themselves to the central index server periodically. Since the nodes are dependent on a central server for communication, these are not pure P2P networks.



P2P Networks.png

Figure 2.1 Centralized P2P Network Topology

Examples: Napster the first popular file-sharing service of late 1990's had a central index server. Napster protocol required a central index server. BitTorrent protocol which is amongst the most popular P2P networks of today is a centralized P2P network.

2.2.1.1 BitTorrent Protocol

File download with BitTorrent protocol is based on the idea that one large file can be downloaded as several smaller chunks that could be finally joined together to form the file. The BitTorrent network contains web-servers that could be searched for .torrent files which are

metadata files. These .torrents files contain information about one particular file that needs to be downloaded, the number of chunks it is divided into and an index server called 'tracker'. Any peer that currently is downloading the file or possesses the file completely periodically sends a message to the tracker server with information about the chunks of the file it already possesses and the chunks it needs to download. Since all the peers that already downloaded or are currently downloading the file communicate this information with the tracker, it has information about how many different peers in the network currently possess on chunk of a file.

So the file search and download process works as follows:

1. 1. User searches for a file-name on a website and downloads the corresponding ¨.torrent¨ file.

2. 2. The BitTorrent client uses the .torrent file to identify the tracker and queries the tracker about the list of peers that possess different pieces of the file.

3. 3. The tracker sends a response with a list of peers containing different pieces of the file.

4. 4. The client establishes a TCP connection with the peers that possess the pieces of the file and downloads the pieces required.

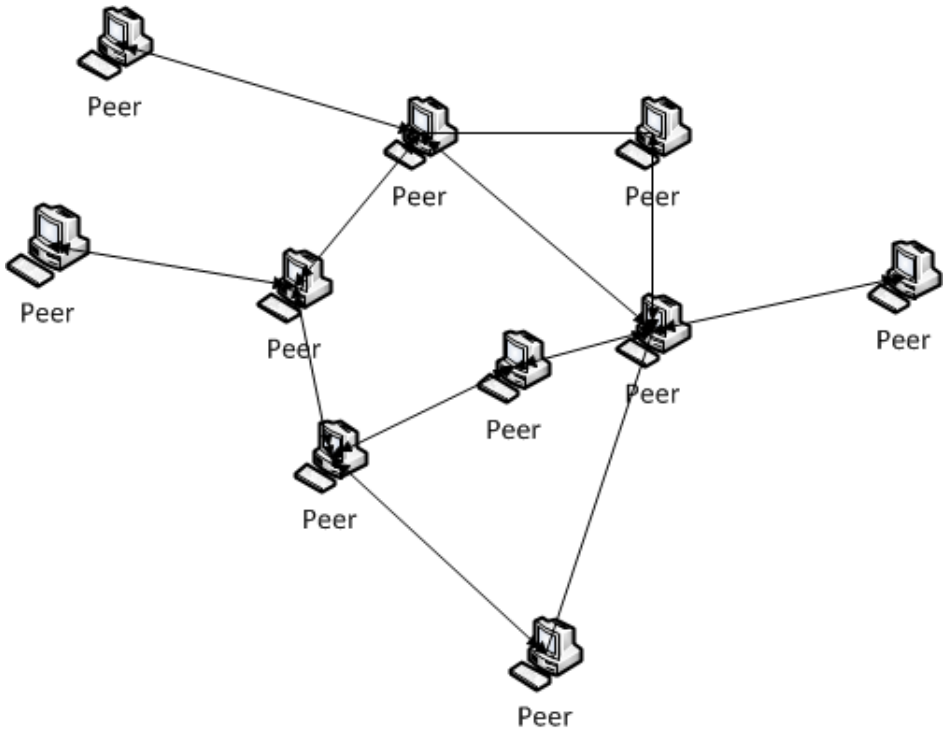
5. 5. During the download process the client continuously communicates with the tracker, updating it about the pieces that it downloaded already and querying about the list of peers that possess the required pieces of the file.

6. 6. Thus the tracker continuously keeps tracks of which node has which chunks of a file. It sends this information to clients that request information about a particular file.

2.2.2 Completely Decentralized Networks

: All the nodes on the network are exactly identical in terms of functionality and information stored. No node contains more information or performs more functions than its peer. Central index servers are absent.

Examples: Traditional Gnutella protocol formed a completely decentralized P2P network where all the nodes were



- mod.png

Figure 2.2 Completely Decentralized P2P Network Topology

Gnutella Protocol Gnutella (v0.4) is a decentralized P2P protocol where each node can act a server or a client. Hence Gnutella2 nodes are called servents. The Gnutella protocol defines the way servents communicate with each other. Specifically the protocol dictates how network formation, maintenance and objects search mechanisms works.

Network Formation

A Gnutella servent connects itself to the network by establishing a connection with another servent currently on the network. A few techniques for finding the IP address of first servent are defined by the Gnutella protocol. After first connection is established, most host addresses are obtained from the network. .

Once a servent's address is obtained from the network a TCP connection is established and a handshake sequence ensues. The servents become neighbours on the network once the handshake completes.

Network Maintenance Once a servent has connected successfully to the network, it communicates with other servents by sending and receiving Gnutella protocol messages.

The following communication messages are defined by the Gnutella protocol (12):

Table 2.1 Gnutella Messages

Message	Meaning
Ping	Used to actively discover hosts on the network. A servent receiving a Ping message is expected to respond with one or more Pong messages.
Pong	The response to a Ping. Includes the address of a connected Gnutella servent, the listening port of that servent, and information regarding the amount of data it is making available to the network.
Query	The primary mechanism for searching the distributed network. A servent receiving a Query message will respond with a Query Hit if a match is found against its local data set.
QueryHit	The response to a Query. This message provides the recipient with enough information to acquire the data matching the corresponding Query.
Push	A mechanism that allows a firewalled servent to contribute file-based data to the network.
Bye	An optional message used to inform the remote host that you are closing the connection, and your reason for doing so.

Any new node that runs Gnutella software probes the network by sending a 'ping' request the neighbour Gnutella servents. The neighbours in turn forward this ping requests. The servents that receive the ping messages respond with a pong message that include the address of a connected Gnutella servent, the listening port of that servent. After the pong message is received a set of servers are selected by the Gnutella program running on the node to send a TCP connect request. A TCP connection is established once a handshake sequence of messages is exchanged and the servents add each other to their neighbour list.

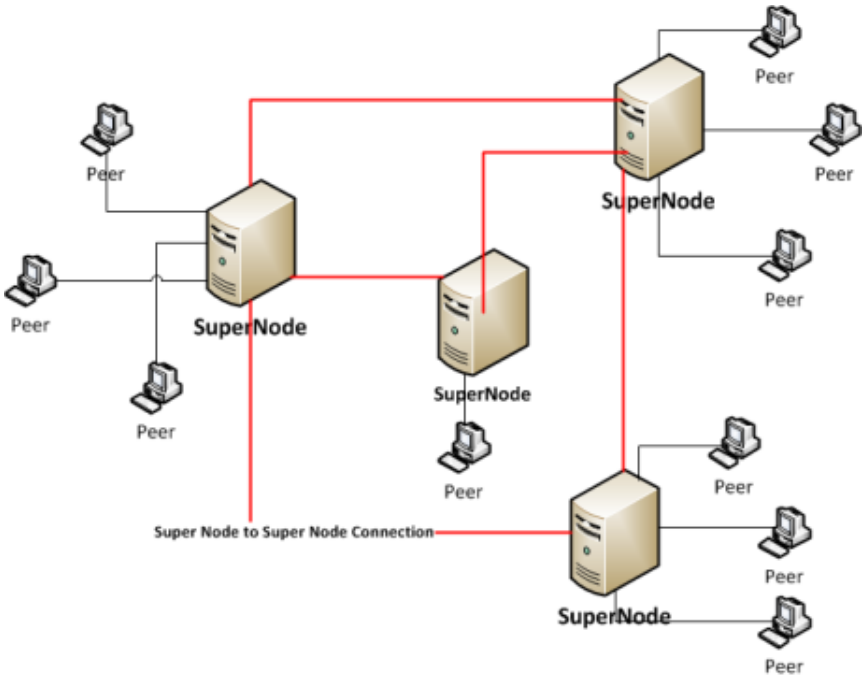
The servents periodically send messages to each other with information about their current status, list of files, neighbours etc.

Object Search Mechanism Gnutella servents broadcast a 'Query' packet described above to all their neighbours. The neighbour servents in turn forward the query to their own neighbouring servents, even when they have the target file. 'Query' packet has a time-to-live(TTL) field in its header that dictates the maximum number of times the packet can be forwarded. Each neighbour decrements the TTL field before forwarding the search requests to its own neighbours. Messages are forwarded to the neighbors within the limited number of hops defined by the Time to Live (TTL) in the message header. If the desired files were found, the servent sends back the matched result set to the neighbor through which it received the query. The servent initiatin the search selects the file from the result set and downloads the target file directly from the peer.Gnutella search mechanism is inefficient because the network is flooded with queries and many nodes might recive the same query multiple time. Newer versions of Gnutella protocol improved this mechanism.

2.2.3 Semi Centralized Networks

Multiple nodes on the network contain indexing information. A inherent hierarchy in the network is created by elevating a few nodes on the network to perform additional functions apart from those performed by regular peers. These additional functions might include maintaining an index of information that neighbours possess, routing queries or maintain/share routing information about 1 hop and 2 hop neighbours. Henceforth these nodes shall be referenced as 'SuperNode' or 'Hub' in this thesis.

Semi-centralized networks are more efficient in terms of routing and information exchange. In case of completely decentralized networks querying and routing mostly takes place though flooding mechanism. In semi-centralized network since the SuperNodes forwarw queries or route other important packets, flooding is reduced. Routing based on indexing information available to the SuperNodes is more efficient. Examples: Modified versions of Gnutella using UltraPeers and Distributed Hash Tables, Gnutella2, eDonkey/eMule and Kad are some of the P2P protocols that could be classified as semi-centralized networks. Gnutella2 protocol is described in detail in the followin sub-section. Though there are differences in these protocols the general concept remains the same throughout. Hence an understanding of one these protocols



centralized.png

Figure 2.3 Semi Centralized P2P Network Topology

would be sufficient to quickly understand the other due to the analogous nature.

2.2.4 Gnutella2

Gnutella2 (9) was inspired from Gnutella protocol but is not an official extension. Gnutella2 gained a lot of popularity notwithstanding the fact that it was an offshoot. The strength of Gnutella2 likes in its optimization of search mechanism. A hierarchy is introduced into the network and nodes were classified into Hub and Leaf networks. A network of highly available nodes called Hub network is responsible for metadata exchange and object search in Gnutella2 network.

2.2.4.1 Gnutella2 - Network Architecture

Gnutella2 network architecture follows the semi centralized P2P network topology. Following sub-sections describe the architecture and search mechanism in Gnutella2 networks:

Leaf Nodes Leaf nodes are common nodes type on the network. P2P nodes contribute through file sharing but do not have any special responsibility. Hence they are not critical to

the network infrastructure and can join or leave at anytime without affecting the network.

Hub Nodes Hub nodes are powerful nodes which contribute substantial resources to the network. Hub nodes play an important part in network maintenances.

Hub Responsibility Hubs are highly interconnected compared to leaf nodes, forming a hub network, with each hub maintaining a connection to other neighbouring hubs. This group that a single hub forms, along with the neighbours, is called a 'hub cluster'. Typically hubs are connected to 5-30 neighbouring hubs. Hubs also accept connections from leaf nodes described above. Leaf nodes are the terminal points of the network. Leaf nodes can connect to more than one hub at once. The entire network can view viewed as a grouping of hub-clusters. Information about the network is shared by hub cluster constantly.

Gnutella2 protocol(9) lists the following as hub responsibilities: Maintaining up-to-date information about other hubs in the cluster, and their neighbouring hubs, providing updates to neighbours. Maintaining a node routing table mapping node GUIDs to shortest route local TCP connections and UDP endpoints Maintaining query hash tables for each connection, including both leaves and hubs so that queries can be executed intelligently Maintaining a superset query hash table including local content and every connected leaf's (not hub's) supplied tables, to supply to neighbouring hubs Monitoring the status of local connections and deciding whether to downgrade to leaf mode, and keeping distributed discovery services such as GWebCaches updated

2.2.4.2 Basic Network Maintenance

Once TCP stream connections have been established forming the overall network topology, a set of basic packet types and behaviors are used to keep the network together and perform maintenance functions. These maintenance packets are documented in this section, each with name, payload, children and behaviours.

2.2.4.3 Query Hash Tables

Note: The following information was obtained from the Gnutella2 protocol description (9).

In the Gnutella2 network architecture, neighbouring nodes exchange compressed hash filter tables describing the content, which can be reached by communicating with them. Query hash tables or QHTs provide enough information to know with certainty that a particular node (and possibly its descendants) will not be able to provide any matching objects for a given query. Conversely, the QHT may reveal that a node or its descendants may be able to provide matching objects. QHTs are very efficient, both in terms of exchange and maintenance and lookup cost.

Queries can be discarded confidently when a transmission is known to be unnecessary without providing the forwarding node, any information about the searchable content. Neighbours know what their neighbours do not have, but cannot say for sure what they do have.

Nodes operating in hub mode maintain an aggregate or superset query hash table, consisting of their own searchable content combined with the QHTs supplied by all connected leaves. This aggregate table is supplied to neighbouring hubs, allowing them to completely filter traffic for the local hub and its leaves. When a change is detected in either the local content or a connected leaf node's QHT, the aggregate table must be rebuilt and patches dispatched to neighbouring hubs. This will happen often, so an appropriate minimum time between updates should be used. One minute is effective. An aggregate table representing hundreds of leaves will be much denser than a table representing one node. This means that all tables must be large enough that the aggregate table remains productively sparse.

2.2.4.4 Object Search Mechanism

Following is a step-by-step description of Gnutella2 Object search mechanism:

- A search client iteratively contacts known hubs with its query
- Hubs match the query against the cached hash tables of their neighbours
- Where a table hit occurs, the query is forwarded once only
- The single injected query thus effectively covers the logical hub cluster, which is the basic searchable unit

- Nodes which actually receive the filtered query process it and send results to the search client directly

When a search client wishes to execute a new search, the following events happen:

1. The search client selects an eligible hub from its global hub cache which has a recent timestamp, has not been contacted more recently than it allows, and has not yet been queried in this search.
2. If a query key is not yet available for this hub, a query key request is dispatched.
3. Once a query key is available, the search client sends a keyed query to the hub.
4. Upon receiving the query, the hub checks the query key for validity.
5. The hub then responds with a query acknowledgement packet, containing a list of neighbouring hubs which have now been searched and a list of 2-hop hubs which have not yet been searched.
6. The search client adds the list of searched hubs to its "don't try again" list, and adds the list of "try hubs" to the global hub cache for future selection.
7. Meanwhile, the hub examines the query to make sure that it has not received it before. Duplicate queries are dropped.
8. The hub then matches the query against the query hash table of all connected nodes, and sends it to those nodes which may be able to field a match.
9. While this is occurring, the hub processes the query locally and returns any results it may have.
10. Leaves directly attached to the hub which have a potential match will have received the query, and process it locally.
11. They may elect to return results directly to the search client, or may return their results to their hub for dispatch.

12. Other hubs in the hub cluster which received the query now examine it to ensure they have not processed it before. They do not send an acknowledgement.

13. Assuming it is new, the hubs match the query against the query hash tables of their leaves but not their neighbouring hubs. Potential leaves receive a copy of the query, and the hub processes it locally.

14. Once again, the hub returns its own results and may forward results for its leaves if they do not wish to dispatch them directly.

15. Meanwhile, the search client receives any search results generated by the hub cluster.

16. The search client makes a decision on whether it should continue its search. If so, the process starts again.

17. The search client will not re-query any of the hubs in the hub cluster, but it has a list of nearby hubs so that the crawl can continue.

3 Challenges And Research Problems

This chapter is divided into two parts. The first part discusses the legal challenges involved in evidence collection on P2P file sharing networks. The second part discusses the technical challenges involved in implementing a solution. Some of the challenges for evidence collection and forensics in P2P networks identified by existing body of research.

The second part analyzes the different monitoring methods for peer to peer networks, their advantages disadvantages based on past work in this direction. Further the relevance of each monitoring method for our purpose of of preliminary evidence collection in P2P networks is discussed.

3.1 Legal Challenges for Evidence Collection in P2P Systems

Forensic investigation of digital crimes involves several stages of evidence collection. Challenges faced at different stages of investigation could be broadly classified into legal and technical challenges described below.

Following is a gist of the legal issues of relating to possession of illegal files from the perspective of criminal investigations.

3.1.1 Jurisdiction

Only a few countries in the world have cyber-laws. Technology develops rapidly, but it takes sometime time for the law to identify and the illegal activities that are made possible by use of technology, a punishable crime. In cases where parts of key evidence is outside the jurisdiction of the law enforcement agency the effort require to carry-on the investgation would make if effectively infeasible.

3.1.2 Privacy and Fourth Amendment

According to the Fourth Amendement (16) "The right of the people to be secure in their persons, houses, papers, and effects, against unreasonable searches and seizures, shall not be violated; and no Warrants shall issue but upon probable cause, supported by Oath or affirmation, and particularly describing the place to be searched, and the persons or things to be seized."

Hence all the investigation conducted before a formal search warrant is issued must use public domain information. Data is exchanged freely by nodes on the P2P file sharing networks and the information collected from searches is voluntary - hence this could be considered as public data. The initial evidence collection tool should not perform any intrusive search/evaluation in order to abide by fourth amendment.

3.1.3 Uploading and Downloading of Illegal Content

Sometimes forensic investigation tools download the files from the P2P network for verification. Some clients that download files are configured to upload parts of files already downloaded. Tools used in investigation should not upload the files even accidentally during investigation. The evidence collection process might be questioned otherwise. Care must be taken while developing an evidence collection tool to disable uploading criminal content.

3.1.4 Digital Millenium Copyrights Act and Academia

Digital Millenium Copyright Act (17) is enforced very strictly on educational institutions. Educational institutions hence do not allow any Peer to Peer file sharing software to run on their network for the penalty if prosecuted for illegal file sharing, is huge. Hence using campus network to run a P2P even if it is only for research-purposes was not allowed.

3.2 Technical Challenges for Evidence Collection in P2P Systems

The number of technical challenges involved in creating a monitoring tool for P2P networks exceed the number of legal challenges. Following section lists a few of the technical challenges

3.2.1 Encryption

Use of encryption for communication between peers complicates monitoring P2P traffic at a network level. Even if network monitors are placed at several places on the internet, use of encruption makes if vritually impossible obtain any meaningful information from the packet dumps. The initial evidence collection tool should be able to perform its function in spite of encryption of network data.

3.2.2 Validation of Evidence

Prosecution requires validation of evidence collected at every stage. If there is a flaw in the initial stages of evidence collection the entire investigation could be invalidated. Therefore validation the initial evidence collected is of paramount importance. Our goal is to ensure that the initial data collected is validated sufficiently in the early stages of investigation so that credibility is maintained. Care should be taken while collecting information such that as much information is collected in a manner that is verifiable at a later stage. The information collected should not only help zeroing in on a particular file, but in cases where malicious programs installed on a users computer are responsible for downloading incriminating content, the data should be useful to exculpate the user of criminal charges.

3.2.3 Blacklisting Monitors on the Network

IP filter software like PeerBlock, PeerGuardian, Moblock (13) could be installed on nodes to prevent a list of known IPs from accessing nodes. A list of IP addresses belonging to the Government, corporations, machines flagged for anti-P2P activities or even entire countries could be obtained from services like iblocklist (15) and the IP filter software could be setup to block all IPs from this list. For the initial evidence collection to succeed if the tool monitors the network by being a part of the network, the IP address of the tool should not exist in such a block list. Also care must be taken that the IP address does not get added to a block-list. Apart from running the tool from an IP associated with a Govt Agency or known monitoring organization like RIAA or MPAA, other ways of getting noticed are aggressive querying of the

network.

3.2.4 Storage Space (Space Complexity)

As stated in 3.2.3 above it is important to collect as much relevant information as possible during initial stages of evidence collection. For examples if a network level packet logger is placed at the gateway, recording information such as the timestamps of packets along with capturing entire packets is more important. However nodes in the P2P networks share a large amount of metadata for maintenance and operational purposes. Logging every single incoming and outgoing packet would quickly add-up to a large amount of data and would require a lot of storage space. Though storage space is becoming cheaper everyday, network bandwidth and it's usage by P2P networks is increasing proportionately as well. Hence logging all the network traffic is not cost-effective. Hence it is important to filter-out irrelevant messages that do not provide any meaningful information for the purpose of investigation.

3.2.5 Processing Power(Time Complexity)

The amount of processing power required to build a system for monitoring and evidence collection has a direct impact on the cost. Due to the large size of P2P networks it is hard to imagine a solution where a single monitor is sufficient to obtain information about the entire network unless it probes every single node on the network. A monitor can be analogous to a node on the network or a device recording relevant network traffic at a lower layer or just a network tap that stores everything that it sees. The number of monitors required to effectively collect evidence and the placement of those monitors are factors that are of paramount importance for the design of an efficient system. A cost-benefit analysis should be performed and the processing power requirement should be optimized.

3.2.6 Network Access

P2P networks are overlay networks that are formed at application layer. Hence two nodes located thousands of miles apart in diametrically opposite ends of the world could be neighbours by adding each other to thier list of neighbours. Hence the geographical locations of application

level monitors is not important. However if network level monitoring is chosen, the placement of monitors becomes an important factor. If a graph of the P2P networks could be drawn, the placement of network level monitors could be viewed as a graph-covering problem. Also access to the underlying network level infrastrcuture is required to place network level monitors.

4 Literature Survey

This chapter presents a review of existing body of work related monitoring in P2P network. Though there are several research papers related to P2P network monitoring, the number of papers related to evidence collection are few in number. Section 4.1 discusses the possible monitoring approaches for a P2P network and reviews research work related to P2P monitors and traces. Section 4.2 reviews the available literature on evidence collection in peer to peer networks.

4.1 Monitoring Approaches for P2P File Sharing Networks

P2P networks are called overlay networks because they form their own networks of computers over the internet. Maintenance of these network infrastructure involves exchange of application level messages over the underlying physical network. As described in Chapter 2, P2P protocols mainly defined the network maintenance and search mechanisms. Monitoring a P2P network for evidence involves monitoring the network topology (architecture) and the messages exchanged between nodes to obtain information about digital contraband being shared. Therefore monitoring can take place at the TCP/IP layers or application layer or both.

Monitoring could also be further classified into active and passive monitoring. Passive monitoring involves recording information passively without querying or actively initiating information exchange with other nodes on the P2P network. Active monitoring on the other hand involves proactive information exchange with other nodes on the network. Hence, a monitoring tool designed to collect evidence from P2P networks could use one of the following monitoring techniques.

An analysis of the emperical monitoring studies on P2P systems is neccesary to obtain a better insight into different monitoring approaches and their applicability to evidence collection in P2P networks. Pitfalls and advantages of each monitoring approach could be enumerated

and thus a wiser design decisions could be made. Huges, Walkerdine and Lee (24) present an analysis of significant P2P traffic monitoring studies between 2000 and 2005. As described above, monitoring of P2P systems could be classified active and passive or network level and application level based on different criteria. (24) mainly classifies the monitoring studies into passive network level monitoring, passive application level monitoring and active application level monitoring. The following sections provide a brief description of each approach followed by an analysis of each approach based on analyis provided by (24) for studies before 2005. Analysis of P2P traffic monitoring studies conducted after 2005 is also presented.

4.1.1 Passive Network Level Monitoring

Passive network level monitoring involves recording TCP/IP packets by introducing agents at appropriate location on underlying physical network infrastructure. The location for where the agent for data collection is placed is critical. For example placing the agent at the network gateway of a large network would record all the traffic routed to nodes in the network. On the other hand, placing the agent on a network segment of switched LAN would record only traffic associated with a single node. Network level monitoring is transparent to the underlying network.

An important requirement in case of passive network monitoring is effective traffic classification. Apart from classifying non-P2P traffic from P2P traffic, the different types of P2P traffic should also be classified. A study of Napster traffic was conducted by (25) in 2000.

Plonka et al., (25), Saroiu et al., (26) and Gummadi et al., (29) studied network traces before 2005 analyzed by Huges et. al(24). It was observed that it was relatively easy to perform the first study by Plonka et al., (25) because of the centralized nature of Napster. (26)[11] and (29) identified traffic by port number and header data respectively. (25) also pointed out that P2P systems were increasingly becoming more difficult to monitor as they use non-standard ports and encrypted header data. To counter this problem , Subhabrata et al., (30) developed a system for real-time network-level identification of P2P traffic which was implemented as an extension to AT&T's Gigascope (31) high speed traffic monitor. They used 'application signatures' to classify realtime network traffic. Evaluation of this approach proved

that it was capable of identifying traffic from popular P2P systems like Gnutella (12), Kadmelia (10), eDonkey (8) and BitTorrent(3). Iliotofu et al., (27),(28) used a dispersion graph based approach to classify all network traffic in general. Based on these studies the following benefits and limitations of network level traces were identified.

Benefits and Limitations of Passive Network Level Monitoring Passive network level monitoring is transparent but introduces additional overhead of effective traffic classification. Even though techniques for traffic classification are invented, if payload is encrpted, not much could be done after the traffic is classified. Access to key segments of the underlying network infrastructure is required for this method. Obtaining access to gateway's of ISPs is not feasible. Gateways of educational institutions or large organizations could be monitored, but these networks do not accurately represent the activity in real private networks.

4.1.2 Active Network Level Monitoring

Active network level monitoring involves injection of packets onto the network or marking packets on the network near the source node and having a monitor placed near the destination node to detect the injected or marked packets. This method cannot be used for evidence collection since introducing content onto the network would raise questions about the validity of the evidence.

4.1.3 Passive Application Level Monitoring

Passive application level monitoring involves introduction of a modified client onto the network that participates in the regular message exchange with the neighbour and logs all the messages received from the neighbours. Regular network maintenance and search messages are logged by the modified client. Passive application level monitoring is transparent to the P2P network.

(22) was the passive application-level trace of P2P was performed on Gnutella network in 2000. Gibson et al., (32) and Coulson et al.(33) were other publications before 2006 that perfomed passive application level monitoring on P2P networks. Both these publications monitored Gnutella 0.6 network. Aidouni et al., (34) and Allali et al., (35) follwoed the approach

of passive monitoring on eDonkey for different periods on time. (34) used the technique of disconnecting and reconnecting randomly to increase the coverage whereas (35) used multiple modified clients.

Though all the work related passive application-level monitoring mentioned here was used to study application level monitoring in an internet wide-context some key insights are obtained about application level monitoring. Huges, Walkerdine and Lee (24) observe that, passive application-level monitoring is transparent, but, it does not require access to low-level network infrastructure. Unfortunately, in cases where a very large sample of network traffic is required, passive monitoring would be unsuitable because of the limited exposure it has, to the network.

Benefits and Limitations of Passive Application Level Monitoring Passive application level monitoring records all the data on the network without being intrusive. A modified client introduced logs all the incoming, outgoing messages from its neighbours and other informational messages received by the client. Thus passive Network level monitoring is suitable for cases where non-invasive approach is desirable. Ineherently this method is protocol specific. Unfortunately, due to the small-world properties of modern P2P networks as pointed out by Adar et al.,(22) a single client using this method does not collect pervasive information about all the nodes in the network.

4.1.4 Active Application Level Monitoring

Active application level monitoring involves deploying a modified client onto the P2P network similar to a class. In this case the client actively queries the P2P network to collect as much information from the network as possible. In passive monitoring at the application level, only the messages that the client exchanges with its neighbours or messages received from remote clients are logged. But in case of active monitoring the client sends queries to known nodes on the network to collect as much information about them as possible. Also the modified client connects to a large number of neighbours and repeatedly reconnects to the network to different nodes in order to cover as much network as possible.

The first known application level trace of Gnutella was performed by Ripeani et al., (48) with a gol of mapping the Gnutella network. A modified client called 'crawler' was created

to repeateldy reconnect to different nodes in it's list in order to build a map of the network. This approach is invasive and is not very scalable. A similar crawling method was followed by Sarious and Gummadi (50) for a month-long crawl of Gnutella network. This recorded both low-level data like IP addresses, bandwidth and high-level data like files being shared and file sizes. However this study just studied the properties in general like bandwidth bottlenecks and the average number of files shared. A study aimed at observing the availability of peers and files on Gnutella was conducted by (52). The method included a crawler that performed searched, intercepted the search-reponses and involed a 'tracking manager' to donwload a peer's file list. After a 40 day monitoring period Chu et al., (52) reported a strong correlation between time of day and node availability. Also Chu et al., (52) observed that file popularity is highly skewed with the top 10% of files accounting for more than 50% of shared data. Based on the work reviewed so far, it is clear that active application level monitoring is very intrusive in nature. If crawline, i.e., repeated disconnection and reconnection is used, the intrusiveness increases and it is not very scalable either. However in few networks searches automatically crawl since they are forwarded by peers.

Benefits and Limitations of Active Application Level Monitoring Active application level monitoring allows gathering a large amount of information about remote nodes on the network within a short time, but at the expense of transparency. Since it is very noisy, active monitoring at the application level would qualify as the best method if invasiveness is not a factor and global network information is required.

4.2 Monitoring P2P Networks for Evidence Collection

Banerjee A. et al., (36) analyze the P2P monitoring implemented by the RIAA and MPAA and concludes that almost all the nodes that do not use a IP block-list to prevent connections from decoy systems are subjected to monitoring. The paper concludes that the top five most prevalent blocklisted IPs contribute to nearly 94% of all blocklisted entities. By blocking the top-5 most prevalent blocklisted IPs, the users can reduce the chances of their being monitored to around 1%. Hence if active application level monitoring approach is chosen, care must be taken that the intrusiveness is as less as possible.

Very recently Latapy et al., (51) presented a work on identify keywords related to CP on P2P network. Their work is based on a large passive application level trace and extraction of keywords related to CP based on existing investigation material. This work could use these keyword information to identify the existence of contraband files very effectively. Also, Liberatore et al., (47) presented a work on monitoring of P2P networks like BitTorrent for forensic evidence collection and uses GUID information for validation of evidence similar to this work. However the initial stages of investigation related to obtaining information about new keywords and files on the network is non-existent. Also it does not examine the differences between popular files and less popular files while monitoring. In this work we propose an architecture that covers: identifying possible contraband files associated with a keyword, identifying nodes on a P2P network that share a particular file and collect sufficient evidence to prove suspicion in a court to obtain search warrant.

5 Design And Implemenetation Of An Evidence Collection Tool In Gnutella2 Network

This chapter illustrates the implementation details of the evidence collection tool. Section 5.1 describes the goals of this work. Section 5.2 describes the constraints and other considerations that are important to the design of the monitoring tool. Sections 5.3 and 5.4 present a framework and describe the components of the framework. Section 5.5 discusses why the proposed solution is feasible. Section 5.6 outlines the expected behavior of the prototype in realistic conditions. Section 5.7 presents the implementation details of the prototype for evidence collection in P2P networks.

5.1 Goals

The goal of this work is to create a working-prototype of a monitoring tool for forensic evidence collection in P2P file-sharing networks. Specifically the tool would be useful in collecting evidence related to possession and distribution of criminal content on P2P networks. The evidence collected would help in narrowing down on a suspect computer distributing criminal content and issuing a search & seizure warrant for it. The following section describes the design constraints and considerations.

5.2 Design Constraints and Considerations

Based on the challenges for P2P monitoring in Chapter 3 and the goals of evidence collection in digital forensics, the following constraints need to be satisfied for the tool to be effective:

- The tool must collect as much information as possible about the location of the criminal content and the time when it existed, for the sake of completeness of evidence.
- The tool must monitor the network as non-intrusively as possible

- The tool must not participate in uploading criminal content even by accident
- The tool must be able to collect information from the network in spite of encryption of messages
- It must be possible to validate the evidence collected from the tool at some stage of the investigation

The following challenges and goals were not applicable for the prototype, but will be important for a fully-functional evidence collection system:

- the information collected should be optimized so that the least possible storage space is required
- quick response time while processing messages logged
- it should be possible to automate the searches

Based on the neccessary constraints mentioned above for efficient evidence collection, the following section proposes a general framework for the same purpose.

5.3 Framework

Application level monitoring is the most suitable approach in order to satisfy the constraints discussed above. Active application level would be effective to collect large amounts of global data within a short period of time. However due to its blatantly invasive approach, it poses a significant risk of the nodes getting added to a block-list, thus rendering the monitoring tool useless. Passive application level monitoring is very effective for our purpose except that the data it collects contains is limited. This could be compensated by introducing more than one re-engineered nodes onto the network instead of one. Since at the application layer level, both approaches have complementing advantages, an ideal tool would use a combination of both approaches. The monitoring tool should intertwine both active and passive monitoring in such a way that the disadvantages of passive monitoring are offset by using active monitoring techniques and vice versa.

Figure 5.1 describes the stages of evidence collection where different monitoring approaches are used at different stages.

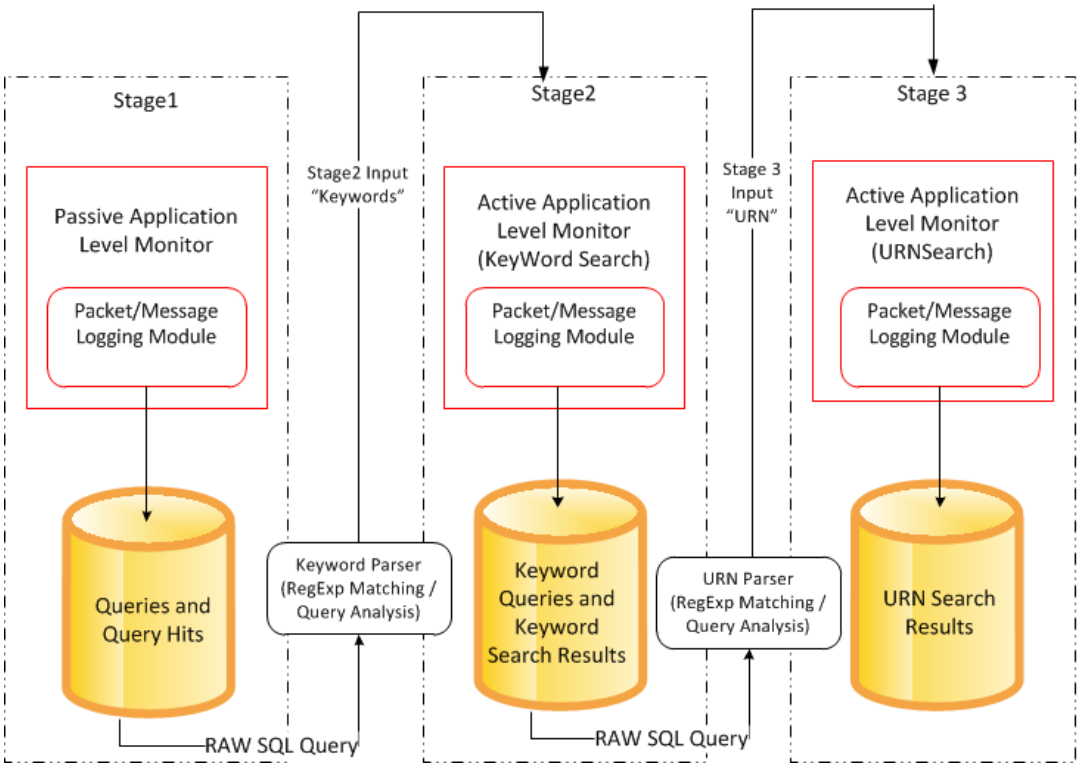


Figure 5.1 Monitoring Framework: Stages

Stage1: In S̈tage1ä monitor is deployed onto the network to log all application level messages passively. This yields to recording of all 'Query' and 'QueryHit' messages. The 'Query' messages could be parsed to obtain keyword related to illegal content on the network. The 'Query Hit' messages could be used to identify nodes on the network that contain illegal content. The keywords obtained as a part of the search act as input to stage two.

Stage 2: S̈tage2üses the keywords identified in stage-1 that could potentially list illegal files in the search results. Active keywords searches are performed in this stage and the URNs of all files obtained as search results for a particular keywords are logged.

Stage 3: S̈tage3üses the URNs obtained in Stage2. This is the crucial stage for accurately identifying the nodes of the network that possess and distribute criminal content. In this stage also, active searches are launched, but for the URNs instead of keywords. URNs are hash values that uniquely identify a particular file.

The following section contains a detailed description about the major components of a monitoring tool and a few implementation specific details.

5.4 Description of Major Components

The key idea behind the monitoring tool is the following: By introducing a modified client onto the network that logs all incoming and outgoing messages with timestamps, the IP addressed and unique identifiers of nodes possessing a particular file on the network can be recorded.

Passive monitoring could be used for collecting historical information and identifying previously unknown or unseen contraband. A database of criminal content could be created based on the passive monitoring of the network. When a specific file is identified by its hash value and investigative agencies need to track the nodes that possess and/or distribute the file, then active monitoring could be used. A search could be launched for the known file hash and all the 'Query Hits' could be logged. Targeted and limited searches reduce the risk of the monitors getting black-listed. Active network level monitoring would provide the required data if extensive information about one particular file or file-type was needed.

The major components of the monitor, as shown in figure 5.2, are the re-engineered P2P client, the message parsing tool, the database for storing passive application level monitoring information and the database querying tool.

A client that uses any one of the modern P2P protocols could be used. However, most of the modern P2P networks follow a semi centralized architecture and hence this work focuses on evidence collection in P2P network with semi-centralized architecture. Gnutella, Gnutella2, KAD etc., use supernodes or hubs on the network for efficient search. Supernodes or Hubs are powerful computers that run the client software and could contribute enough hardware resources and processing power to the network. Since the supernodes or hubs are responsible for forwarding the search queries, the number of search queries and search results visible to them is very large compared to a regular node. Hence for passive application level monitoring is effective in semi-centralized network when the monitoring tool is is super-node mode.

P2P Network/Protocol: The general search mechanism works in a similar fashion in

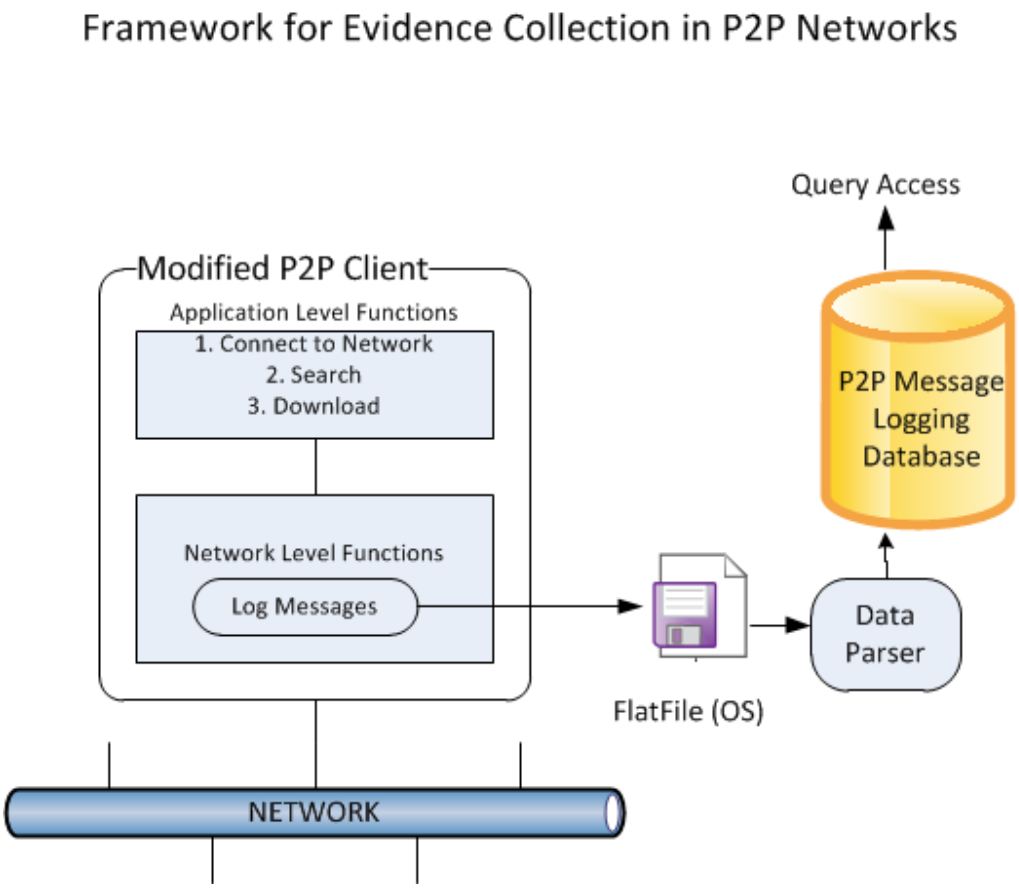


Figure 5.2 Anatomy of the P2P Monitor

case of GnutellaV0.6 and Gnutella 2. For the purpose of this work a Gnutella2 client was chosen for re-engineering. As noted in the 'Literature Survey', no significant monitoring or traffic analysis of Gnutella 2 could be identified prior to this work. Though Gnutella 2 is not as popular as Gnutella, a working prototype of a monitoring tool implemented for Gnutella 2 network could be extended to Gnutella v0.6 network, ED2K etc.

Monitoring at the application layer involves logging of several messages exchanged by the hub-cluster and by the leaves and hubs. Following section describes a few of the important messages of the Gnutella-2 network.

5.4.1 Important Gnutella2 Messages

Based on the Gnutella2 protocol specification, the following application level packets / messages yeilded key information for investigation:

Search Query(Q2) (37) message contains the Universal Resource Name(URN), Descriptive Name(DN) (Generic) Criteria, Metadata Criteria, Object Size Restriction Criteria.

Query Acknowledgement (QA) (38) message is used by a target hub to acknowledge the receipt of a query and to indicate that it is processing the query. The target hub also provides information to ensure that hubs are not searched more than once in a given query. Lists of 'Done Nodes' and 'Search Hub' nodes are sent to the querying client. These done nodes are all neighbours of the target hub which need not be sent the search queries since they were effectively searched by the target hub. The 'Search Hub' nodes are additional hub-nodes in the cache of the target hub, that were not effectively searched by the target hub.

Query Hit (QH) message (39) contains one or more search result descriptors, along with information about the node like GUID, node address(IP address + port number), neighbour hub address, descriptive name, metadata and timestamp when the result-set was generated. In response to a single query message, a single node could respond with multiple 'Query Hit' messages.

Local Node Information (LNI) (40) messages are sent only to neighbours of a node at regular interval. This message contains GUID and the local library information.

5.4.2 Leveraging Information Logged by the Modified Gnutella2 Client

Amongst the Gnutella2 packets described above each of the packets could be used to obtain information about the network and files distributed by the nodes of the network. It must be noted that these packets are essentially application layer protocol messages. Information within each packet can be leveraged to extract important information about nodes possessing illegal content.

Passive application level monitoring records all possible packets that the client receives. Search keywords that could possibly yield results of contraband or search-hits that are associated with illegal files could be extracted from the 'QUERY' packets. 'QUERY HIT' packets could be used to identify the nodes on the networks by IP address and port number. A 'Globally Unique Identifier' (GUID) could be recorded for each 'QUERY HIT' along with the time-stamp for the packet received. A 'QUERY ACKNOWLEDGEMENT' packet sent by the target hub

contains lists of 'Done Nodes' and 'Search Hub' nodes to the querying client. 'Done nodes' need not be sent the search queries since they were neighbours of target hub and were effectively searched by the target hub. Based on this information we know the list of neighbours for each hub that a query is sent to. Based on this information the hub-network could possibly reconstructed. Hub network is dynamic and changes, but since hubs have relativeley high uptime the restructuring or reorganization of the network should be slower than the leaf nodes.

5.5 Feasibility

(Why it is working) Deploying a re-engineered client onto the network to perform different kinds of monitoring requires one node. In initial stage i.e., during passive monitoring the modified client running as a hub has a limited network view and hence collects limited data. Deploying additional nodes onto the network is a possible solution. However information about illegal files could be found-out by directly starting at stage 2 i.e., active keyword searches. An important requirement in order to start at Stage2 is to know the important keywords related to the investigation. However looking at particular types of crimes, the investigating agencies already must have identified the most probable keywords relevant to the case they are investigating. Stage1 only helps in creating a consolidated list keywords and identify new keywords related to any investigation. Stage3 uses the same technique as stage two and therefore no additional required to carry-out stage3 of the framework. Hence in order to start the investigation at stage-2 of the framework, only one computer with enough processor and memory to run the modified Gnutella2 client and enough storage to store all packet information are required. Additional processing power and storage might be required for other components like the data parser and the relational database. However those requirements might require a maximum of a second computing unit.

5.6 Realistic Conditions

Under realistic conditions the major issues faced are network coverage in Stage1, unrelated and misleading search-results in stage2. Network coverage issues can be solved by disconnecting

and reconnecting to the network multiple times. Alternatively placing multiple monitors is a solution. However as mentioned in the previous sections, stage1 is less crucial to the investigation if important keywords are know. The problem with stage-2 i.e., keyword search is the noise in the search results. In case of an automated investigation, these irrelevant search results which can be considered as noise should be filtered out. An important challenge is identifying the noise without downloading the files. This could be possible done by looking at file size and other such factors. Also while deploying these behind a gateway or a NAT, port-forwarding should be enabled for other nodes on the network to communicate with the monitor. These are the major issues that need to be addressed in realistic conditions.

5.7 Prototype Implementation and Optimizations

Implementation details of the prototype, ie., the major components listed in figure 5.2 are presented in this section. The modified client was a re-engineered open-source Windows-based Gnutella 2 client called Shareaza (41). The re-engineered client was introduced onto the network to log the application-level messages(packets) sent and received. The messages were logged/recorded in regular flatfiles on the operating system. The re-engineered client was scheduled to stop executing twice a day. Perl scripts were scheduled to execute during the down-time to copy the flat files to an archive.

Note: Shareaza uses a GPL License. This modified version of Shareaza client developed as a part of this program was not distributed for public use.

Three flat-files were created and saved to the disk on the node - one file contained a list of all Gnutella2 messages sent and received by the client one file contained a list of all hosts in the host-cache of the client one file contained a list of neighbours the client had during its run-time

The data parsing and data querying tools were written in Perl. The PERL script for parsing the data and uploading the data into the database could be located in APPENDIX2. Data was extracted from the flat files and uploaded to a MySQL database. The MySQL database was queried for analysis purposes by the data querying scripts. Additionally for the sake of human readability while querying the messages recorded, a web-interface was developed.

5.7.1 Modifying Shareaza - Implementation Details

Shareaza v2.5.4 was the final version used for this work. While earlier versions of Shareaza contained the necessary code, the final implementation was a fork version 2.5.4. (43) has source-code of latest versions of Shareaza. Compiling Shareaza using Visual Studio 2010 is not straightforward. (42) contains detailed instructions on how to compile Shareaza. In order to compile shareaza we require Visual Studio, gzip, Boost library and Inno QuickStart Pack Setup. The details about using these could be found in (42).

Shareaza source contained over 400 C++ files that are responsible for maintenance of the network and operations. Out of these files a few important classes were identified and exploited to log the incoming messages to and outgoing messages from the Shareaza client. Packet, Neighbour and QueryHash table were a few of the more important classes that need to be understood.

Packets are important pieces of information that need to be processed quickly and efficiently by Shareaza. Shareaza supports eDonkey, Bit Torrent, Gnutella and Gnutella2 protocols. Hence the packet class is inherited in different levels. Packet.cpp, CG1Packet.cpp, CG2Packet.cpp, EDPacket.cpp, BTPacket.cpp were PacketPool.cpp were the important classes to understand. Classes CG1Packet.cpp and CG2Packet.cpp along with the modifications were included in Appendix-A

P2P handle neighbour information very frequently. In face maintenance of the network dependence on checking if a neighbour is alive and connecting to other neighbours on the network if no messages could be received from the neighbour. The 'Neighbour' class is important to understand since different types of packets are obtained from neighbours compared to other nodes. Neighbours typically send more information compared to other nodes. In Shareaza, the 'Neighbour' class has a long list of inheritances. The chain of inheritance from parent class to child classes is noted as follows: NeighboursBase, NeighboursWithG1, NeighboursWithG2, NeighboursWithED2K, NeighboursWithRouting and NeighboursWithConnect and CNeighbours. Code modified in a few of these classes is included in Appendix-A.

Query Hash Tables were maintained by hubs. Hubs maintained one aggregate query hash-

table for all the child tables and one query hash table for each of the neighbours. Whenever hubs encountered a query, the query hash table was checked to see the possibility of that query being fulfilled by any of their leaves or neighbours. 'QueryHashTable.cpp', 'QueryHashMaster.cpp' and 'QueryHashTable.cpp' are important classes related to QueryHash table.

Few windows in Shareaza power-mode displayed packet information, host-cache information and neighbout information. The code for these windows was tapped so that everytime an update or repaint operation occurs in these Windows, shareaza first writes the data onto OS flat-files before updating the windows. Everytime a packet is recieved, it's information is dumped onto the 'Packet Dump; windows and into a text-file in the windows 'Appdata' folder for the user under whose username Shareaza is running. 'WndPacket.cpp', 'WndHostCache.cpp' and 'WndNeighbours.cpp' were the windows whose code was modified to obtain and dump of different information. Copies of the modified code are included in Appendix-A.

In most of the cases, the modified code was commented and comments containing 'Modified by Teja' was added to the code. However a few trivial modifications might not contain comments. In case differences between original source and the modified source needed to be found, programs like vimdiff could be utilized.

In order to activate the network packet logging, once the modified code was compiled and the setup of Shareaza was completed - Shareaza needed to be started in 'PowerMode' where additional menu options are available under the view menu. The menu options 'PacketDump', 'HostCache' and 'Neighbours' are used to start recording the information into text files. The files with packet information dump, host-cache dump and neighbour-list dump could be located in the 'Appdata' folder of the logged-in user in Windows. These files were collected and parsed using the parser script. Before parsing the data was copied into folders with titles 'Data_yyyy_mm_dd'.

5.7.2 Testing and Optimization

The modified Gnutella 2 client was introduced onto the network and was elevated to a hub-status. The minimum memory, processing power and uptime requirements for a hub-node were met.

The tool collected 4-6 GigaBytes(GB) of packet data per day for the hub-node. It was observed that about 25% of the overhead was caused by the long delimiters in the data to assist parsing. Another opportunity to reduce the amount of data logged was by not logging unnecessary packets, depending on the purpose. Eliminating all packets other than 'Query' and 'Query Hit' packets, the size of the dump fell to about 0.7-1.2 GB.

6 Experimental Evaluation

This chapter describes the experiment setup for evaluating the tool and the findings based on both active and passive monitoring of the Gnutella2 network and presents an analysis of the findings.

6.1 Evaluation

The monitoring tool can be evaluated based on its ability to: (a) identify new criminal content on the network by passively logging search-hits and (b) identify the nodes on the network that posess a particular file along with their GUID(Globally Unique Idnetifier) values.

Three data-collection experiments were setup in order to evaluate the tool. The Information collected through both active and passive monitoring through these experiments was presented in the results section.

6.1.1 Experiment 1: Passive Monitoring Only

The goal of this experiment is to verify the working of the re-engineered Gnutella 2 client as designed. In other words, after recording 'Query', 'Query Hit' packets and the number of keywords extracted was observed. In this experiment the re-engineered Gnutella-2 client was introduced into the network and data was logged for five consecutive days. An analysis of the data collected is described in the results section of this chapter.

6.1.2 Experiment 2: A Comparision of Active and Passive Monitoring

In this experiment two nodes were introduced into the network one in leaf mode and one in hub-mode. The re-engineered client in the leaf mode was used to launch searches for specific keywords once a day and all the search results were logged. The messages logged were analyzed to obtain the number of nodes with unique IP addresses in posession of files containing a specific

keyword. Results are presented in the results section of this chapeter. Another re-engineered client in hub-mode ran for five consecutive days to collect all incoming and outgoing messages. The number of 'query hit' messages logged messages that contained the same keyword used in the active monitoring approache previously were extracted. A comparitive analysis of the number of 'Search Hits' recorded by the passive monitoring vs the total number nodes derived through active monitoring approach was performed. Results of the analysis are presented in the 'Results' section of this chapter.

6.1.3 Experiment 3: Actively Tracking a File on the Network

In this experiment a leaf-node was used to to query the network for a Uniform Resource Name (URN). A URN is a hash value that uniquely indentified a file. Two URN searches, one for a very popular file and one for a mildly popular file, were performed each day for five continuous days. The search results were logged by the node. The number of unique nodes that contained the file on Day 1 and the propagation of the file on the network could be observed using the data collected. There is no indication when the file was originally downloaded by a node. However if a node was previosuly a part of search results and it did not posess the file, then it is possible that the file was recently downloaded to the node. The changes to the unique-node population in posession of two files over a five day period was described in the 'Results' section fo this chapter.

6.2 Results

6.2.1 Experiment 1 - Results

The ip address, date, time and the message payload were logged by the re-engineered Gnutella2 client for all Gnutella 2 packets. Important information could be obtained by querying the payload of the packets described in section 6.2.1.

The number of records of relevant incoming messages by type is shown in fig 6.1 below:

A web-based application was created to query the database in order to identify search-queries and search hits with particular key-words that might indicate posession of or intent to

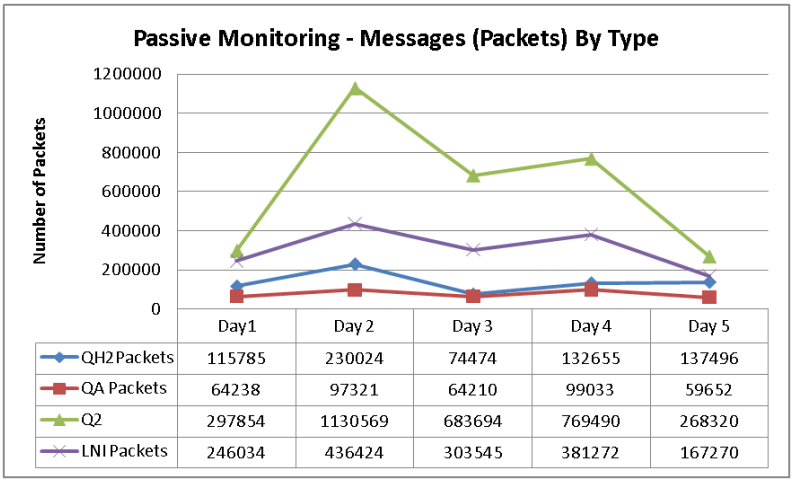


Figure 6.1 Number of Packets Recorded by Type

Table 6.1 Keywords Extracted - Five Day Data Collection

Day	Keywords Extracted from Q2	Keywords Extracted from QH2
Day1	2592	1317
Day2	4427	1908
Day3	3646	1756
Day4	3853	1689
Day5	2014	1352

obtain criminal content. Images X and Y below show the the search criteria and sample results to identify historical searches and historical search results for potential criminal content.

6.2.2 Experiment 2 - Results

In experiment two, six keywords were chosen - three very popular and three midly popular. A search was initiated in a re-engineered Gnutella2 client running in leaf-mode, once a day for each keyword. These searches were performed for five consecutive days. In parallel another re-engineered client in hub-mode was deployed on the Gnutella2 network to passively log messages. The number of 'Query Hit' messages that contained the chosen keywords were extracted from the data recorded.

Unfortunately, due to human error or less popularity of the keyword, the final data gathered contained information about only two of the six keywords for all five consecutive days - one

Figure 6.2 Query - Historical Search Information

Figure 6.3 Query Results - Historical Search Information

very popular keyword and one mildly popular keyword. For precision purposes the very popular keyword and the less popular keyword shall be referred to as 'Keyword A' and 'Keyword B' henceforth.

The graphs in figures 6.4 and 6.5 contain the total number of 'Query Hit' packets received in active and passive mode for different keywords. In order to get a better insight into the number of nodes on the network that posess the file, the number of unique 'IP address' and 'Port Number' combinations from which the query hits were received was deduced. The number of unique IP address and port number combinations from which a Query hit packet was received

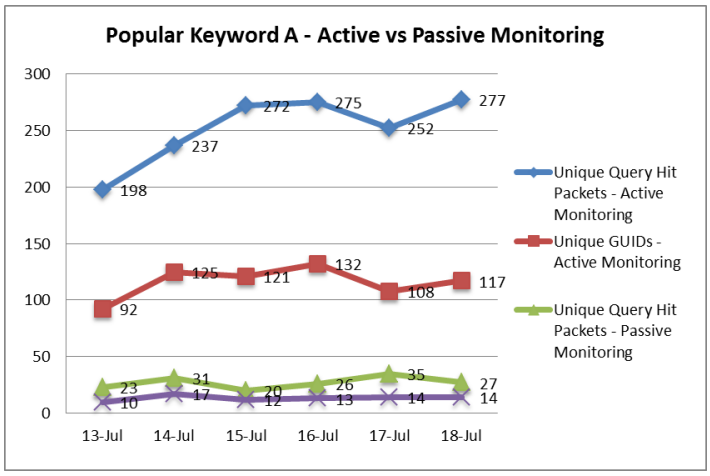


Figure 6.4 Popular Keyword - Active vs Passive Monitoring

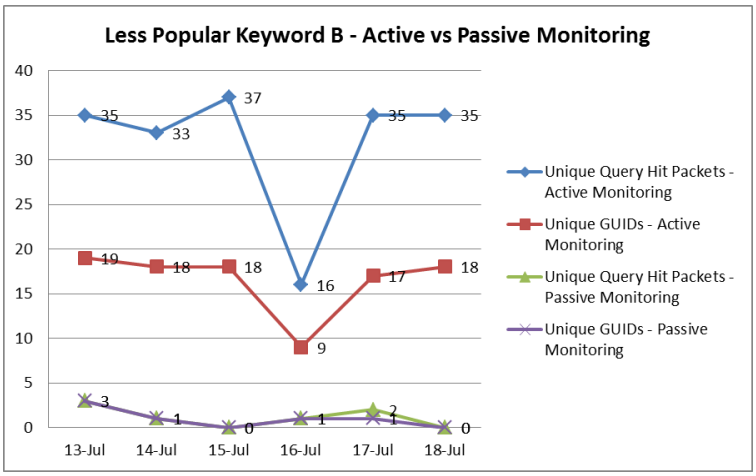


Figure 6.5 Less Popular Keyword - Active vs Passive Monitoring

was also deduced from the passive monitoring data.

Figure 6.4 contains the comparision of active mode and passive mode monitoring for the number of nodes that contain files matching Keyword-A. Figure 6.5 contains a similar comparision as figure 6.4 but for nodes containing files that match Keyword-B.

It was observed that for a popular 'keyword' the number of search-hits logged by the node monitoring the network passively were about 10 percent of the of the total search-hits logged by node searching for the keyword actively. After initiating an active keyword search, several files related to the keyword were a part of the search-results. Hence keyword based searches identify many different files related to a particular keyword.

6.2.3 Experiment 3 - Results

URNs for a popular and midly popular files were chosen from keyword search results and from messages logged during passive monitoring. URLs typically are hash values obtained from standard hashes like SHA1.

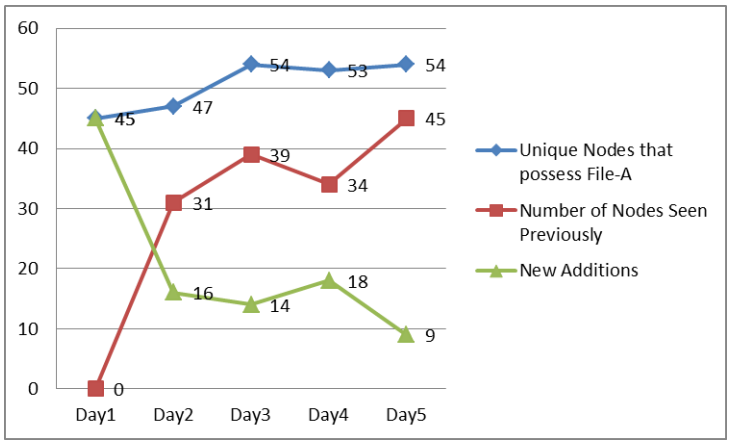


Figure 6.6 Very Popular File on Gnutella2 network - 5 Day Tracking Information

Table 6.2 Very Popular File - 5 Day Tracking Information

Day	Unique Nodes that posess File-A	Number of Nodes Seen Previously	New Additions
Day1	25	0	25
Day2	27	21	6
Day3	33	23	4
Day4	32	32	6
Day5	33	33	0

Figure 6.7 shows the trend of unique nodes identified by the tool 5 consecutive days that posess a popular file. The table 6.5 also contains the information about the number of new nodes identified each day. Figure 6.7 and Table 6.6 provide similar information as Figure 6.7 and table 6.5 but for a mildly popular file.

It was observed that case of a popular file that the number of new nodes that were identified to posess the file reduced drastically after 3-4 days. However for the midly popular files the number of new nodes identified to posess the file each day was almost zero.

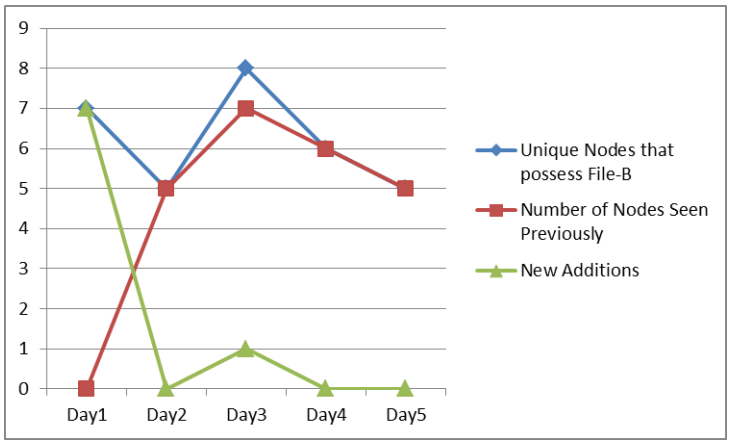


Figure 6.7 Less Popular File - 5 day Tracking Information

Table 6.3 Less Popular File - 5 Day Tracking Information

Day	Unique Nodes that posess File-B	Number of Nodes Seen Previously	New Additions
Day1	7	0	7
Day2	5	5	0
Day3	8	7	1
Day4	6	6	0
Day5	5	5	0

7 Discussion - Network Coverage

Active and Passive Monitoring involves introduction on a modified P2P client onto the network. In order to avoid being very intrusive and seem like an anomalous node, we proposed limiting of the number of searches performed. However steps must be taken in order to increase network coverage in the passive monitoring mode. It is also important to cover the nodes that continously join and leave this network. This chapter discusses the possible techniques to achieve network coverage.

7.1 Increasing Messages logged in Passive Application Level Monitoring

All searches in the Gnutella 2 network are performed over the hub-network. Each hub maintains a query hash table (QHT) for each of its neighbours and an aggregate query hash table for all its leaf nodes. Whenever a search query is received, it is forwarded only to the hubs who QHT contain a value for that keyword or URN. However, the query acknowledgement sent to the querying node would contain all the neighbours hubs in the 'ALREADY QUERIED' list. Since the hub checked the possibility of neighbouring hubs containing the file, this process is valid. Therefore the modified client introduced onto the network has limited visibility in terms of search queries it receives through forwarding from neighbouring nodes. Also it receives limited network coverage since it is connected to a few hubs.

Note: Searches could be keyword-based or URN based. URN is a hash value

To log maximum search queries on the network the following techniques might be used:

- Introduce multiple modified clients that connect to different parts of the Network
- Initialize the QHT so that all Queries record a Search Hit
- Introduce multiple modified clients that connect to a large number of leaves so that the QHT is dense.

7.1.0.1 Introduce multiple modified clients that connect to different parts of the Network

Introducing multiple client in different IP blocks and controlling the neighbour connections such that they do not have any common one-hop or two-hop neighbours could significantly increase the coverage of the network. During a search operation, the Shareaza client indicates the number of hubs searched for a particular keyword. It was observed that normally the number of hubs available were between 2500 and 3000. In order to cover as much of this hub network as possible the number of neighbours and the number of nodes introduced onto the network should be optimized by trying to connect to hubs with large number of neighbours.

Knowledge of network topology is an important factor for introducing multiple clients onto the network. The information obtained can be used to introduce hub-nodes into the network at different location so that the entire network topology is covered by as few monitors as possible. For a real-world investigation this could aid capturing as much network traffic as possible. Based on the information from a Query Acknowledgement packet that is sent by hub in response to a query, all the neighboring hubs for the given hubs could be obtained. Depending on the number of different sources that send this packet to out monitoring hub,neighbors and second-hop neighbors for many hubs on the network could be identified.

Active application level monitoring could help for obtaining an accurate topology of the network because each search request sent by the Hub receives a Query Acknowledgement packet. Hence if more searches originate from the HUB, more Query Acknowledgement packets are received in a shorter time. Hence a quicker and a more accurate snapshot of network topology could be obtained.

7.1.0.2 Initialize the QHT so that all Queries record a Search Hit

Each hub in Gnutella2 mantains 'Query Hash Table' information for neighbouring hubs. Queries in Gnutella2 get forwarded to neighbouring hubs only if there is a 'Query Hit' or in other words a possibbility of finding the file in the neighbouring hub-cluster. Hence if the source-code of the client is modified in such a way that the Query Hash table is initialized to

be full then the neighbour hubs will forward all search queries to our modified hub-node. This method would be effective to collect many search-hits and search results. However providing false search-hits would draw significant attention since the search hits do not provide any actual files for downloading.

7.1.1 Introduce Multiple Modified Hub-Nodes that allow Large Number of Leaf Connections

A large number of leaves generally increases the density of the Aggregate Query Hash Table. Instead of falsely initiating the Query Hash Table to be densely populated, in this case the actual files do exist with the leaves. Hence the download requests for these files will not make the node seem anomalous. Also since the Query Hash Table is densely populated, more search queries are forwarded to the hub-nodes. Thus more queries are covered by introducing a few hub-nodes in this case. However in order to serve a large number of leaf nodes significant processing power is required.

8 Summary And Future Work

The first part of this chapter summaries the results of the thesis. The subsequent section describes future work.

8.1 Conclusion

Based on the data collection and evaluation through experiments in the previous section it can be concluded that

- the prototype developed as a part of this work could be used to identify the a node on the Gnutella network in possession of criminal content
- active keyword searches could also be used for identifying content on the network provided they are not very intrusive
- solid investigation techniques like timestamping and collecting public domain information make the prototype developed, a practical investigation tool in the real world.

8.2 Future Work

Future work includes opportunities to maximize information collected in passive-monitoring mode and leveraging the ability to reconstruct a large part of the hub-network based on messages logged. From a more practical tool development point-of-view, optimizing the data-parsing techniques for parsing the data collected and optimizing the queries to obtain meaningful information from the database are equally important. Finally, in cases of crimes like distribution of Child Pornography, indentifying the initial uploader might be helpful in preventing child abuse. Future work includes the problem of identifying the initial uploader.

8.2.1 Possible Performance Optimizations

Currently the entire payload of all the packets received by the modified Gnutella2 client is stored in the database. A lot of packets do not provide any meaningful information. Hence identifying the important relevant information and storing only that information would reduce the storage space required and the response time for queries. Hence both space complexity and time complexity is reduced using measures like these.

APPENDIX A Shareaza Classes and Code Snippets

Attached in this appendix are a list of Gnutella2 classes that were modified by the author.

Re-engineered Gnutella2 Client

For most of the important modification to the code included search for 'Modified by Teja' in the comments section of the files included in the attachments.

Attached filed 'NeighbourClass.zip', 'PacketClass.zip', 'QHTClass.zip' and 'WndClasses.zip' contain the important C++ classes that either need to be understood or contain some modifications in order to record application-level messages using Shareaza.

The method 'SmartDump' contains the code for extracting the packet information before processing the packet.

The following function was added to the end of WndPacket.cpp:

```
void CPacketWnd::WritePacketToFile(CString record)
{

int iChars = 1 + record.GetLength ();

//Write the UTF-8 output string into the file.
CT2A outputString(record, CP_UTF8);
localFile.Write(outputString,::strlen(outputString));
}
```

APPENDIX B Data Parser - Perl Scripts

Custom delimiter strings are used to separate each packet logged by the modified Gnutella2 client. The parser perl-script uses these delimiters, the date-indicator on the file-name and the timestamps on packets to create an appropriate record in the MySQL database. Perl uses a database driver to connect to the MySQL database.

Attached file parse_script.zip contains the parser perl-script.

Bibliography

[1] *http://www.riaa.com/faq.php*

[2] There Goes a Neighbourhood: A Forensic Computing Case Study *http://web.interhack.com/publications/crimdef-casestudy.pdf*

[3] Bit Torrent Protocol *http://www.bittorrent.org/beps/bep_0003.html*

[4] Gnutella Protocol v0.4 *http://rfc-gnutella.sourceforge.net/developer/stable/index.html*

[5] Michael D. Mehta, Don Best, and Nancy Poon, 'Peer-to-Peer Sharing on the Internet' Canadian Journal of Law and Technology Vol 1, No 1.

[6] The Real Cost of Free Programs such as Instant Messaging and Peer-to-Peer File Sharing Applications Sigrun Grabowski GSEC Practical Assignment V. 1.4b Option 1 July 1, 2003

[7] Gnutella Protocol v0.6 *http://rfc-gnutella.sourceforge.net/src/rfc-0_6-draft.html*

[8] eDonkey/eMule Protocol Specification *http://www.cs.huji.ac.il/labs/danss/presentations/emule.pdf*

[9] Gnutella2 Protocol Specification *http://g2.trillinux.org/index.php?title=Main_Page*

[10] Kadmelia Protocola *http://pdos.csail.mit.edu/ petar/papers/maymounkov-kademlia-lncs.pdf*

[11] Sandvine (2011) Intelligent Broadband Networks. Global Internet Phenomena Report. *http://www.sandvine.com/downloads/documents/05-17-2011_phenomena/SandvineGlobalInternetPhenomenaReport.pdf*

[12] Gnutella Protocol version 0.4 *http://rfc-gnutella.sourceforge.net/developer/stable/index.html*

[13] *http://moblock.berlios.de/*

[14] Peer to Peer Client Investigation Software *http://www.p2pmarshal.com/*

[15] *http://www.iblocklist.com/lists.php*

[16] Fourth Amendment, Constitution of United States of America. *http://www.gpoaccess.gov/constitution/html/amdt4.html*

[17] The Digital Millennium Copyrights Act *http://www.copyright.gov/legislation/hr2281.pdf*

[18] Is Deviant Behaviour the Norm on P2P File Sharing Networks?, in IEEE Distributed Systems Online, volume 40 7, number 2, 2006

[19] Child sex crimes on the internet. Report for the State of Wyoming Attorney General.

[20] Child Pornography: an internet crime. Brighton: Routledge.

[21] Child pornography and the internet: perpetuating the cycle of abuse. Deviant Behavior, 23(4), 331-362

[22] Free Riding on Gnutella. Adar, E., Huberman, B., First Monday, October 2000. http://www.firstmonday.dk/issues/issue5 10.

[23] Is Deviant Behaviour the Norm on P2P File-Sharing Networks? Daniel Hughes, Stephen Gibson, James Walkerdine, Geoff Coulson Computing Department and Psychology Department, Lancaster University, Lancaster, UK

[24] Monitoring Challenges and Approaches for P2P File Sharing Systems, in the proceedings of the 1st International Conference on Internet Surveillance and Protection (ICISP06), Cap Esterel, France, 2006, page 18-18

[25] Napster Traffic Measurement, Plonka D., Univeristy of Wisconcin-Madison, available online at: *http://net.doit.wisc.edu/data/Napster, March 2000*

[26] An Analysis of Internet Content Delivery Systems Saroiu S., Gummadi K., Dunn R. J., Gribble S. D., Levy H. M., published in the proceedings of the 5th International Symposium on Operating Systems Design and Implementation (OSDI) Boston, USA,2002

[27] Network Traffic Analysis using Traffic Dispersion Graphs (TDGs): Techniques and Hardware Implementation by Marios Iliofotou et al., (2007)

[28] Graph-Based P2P Traffic Classification at the Internet Backbone, Iliofotou, M et al.,

[29] Measurement, Modeling and Analysis of a P2P File- Sharing Workload, Gummadi K., Dunn R. J., Saroiu S., Gribble S. D., Levy H. M., Zahorjan J., published in the proceedings of the 19th symposium on Operating Systems Principles (SOSP03), Bolton Landing, New York, October 2003.

[30] Accurate, Scalable Network-level Identification of P2P Traffic Using Application Signatures Subhabrata S., Spatscheck O., Wang D., published in the proceedings of the thirteenth international world wide web conference (WWW2004), New York, USA, 2004.

[31] Gigascope: *http://public.research.att.com/viewProject.cfm?prjID=129*

[32] Is Deviant Behaviour the Norm on P2P File Sharing Networks? Hughes D., Gibson S., Walkerdine J., Coulson G., in press in IEEE Distributed Systems Online, vol. 7, no. 2, February 2006. *http://csdl.computer.org/comp/mags/ds/2006/02/o2001.pdf*

[33] Free Riding on Gnutella Revisited: the Bell Tolls? Hughes D., Coulson G., Walkerdine J., published in IEEE Distributed Systems Online, vol. 6, no. 6, June 2005. *http://csdl2.computer.org/comp/mags/ds/2005/06/o6001.pdf*

[34] Ten weeks in the life of an eDonkey server, Aidouni, F.; Latapy, M.; Magnien, C.(2009)

[35] Measurement of eDonkey activity with distributed honeypots Allali, O., Latapy, M., Magnien, C., (2009)

[36] Anirban Banerjee, Michalis Faloutsos, Laxmi Bhuyan, The P2P war: Someone is monitoring your activities, Computer Networks, Volume 52, Issue 6, 24 April 2008, Pages 1272-1280, ISSN 1389-1286, DOI: 10.1016/j.comnet.2008.01.011. *http://www.sciencedirect.com/science/article/pii/S1389128608000091*

[37] Gnutella 2 Search Query Packet *http://g2.trillinux.org/index.php?title=Q2*

[38] Gnutella 2 Query Acknowledgement Packet *(http://g2.trillinux.org/index.php?title=QA)*

[39] Gnutella 2 Query Hit Packet *(http://g2.trillinux.org/index.php?title=QH)*

[40] Gnutella 2 Local Node Information Packet *(http://g2.trillinux.org/index.php?title=LNI)*

[41] *http://shareaza.sourceforge.net/*

[42] Compiling Shareaza *http://sourceforge.net/apps/trac/shareaza/wiki/Guides/Compiling%20Shareaza*

[43] Compiling Shareaza *https://shareaza.svn.sourceforge.net/svnroot/shareaza/trunk/*

[44] Danny Hughes. Monitoring challenges and approaches for P2P File-Sharing systems.

[45] The Recording Industry Association of America (RIAA) *http://www.riaa.com*

[46] [12] Availability and locality measurements of peer-topeer file systems Chu J., Labonte K., Levine N., published in ITCom: Scalability and Traffic Control in IP Networks. July 2002, vol. 4868 of Proceedings of SPIE.

[47] M. Liberatore, R. Erdely, T. Kerle, B. N. Levine, and C. Shields. Forensic Investigation of Peer-to-Peer File Sharing Networks. In Proc. DFRWS Annual Digital Forensics Research Conference, August 2010

[48] Mapping the gnutella network, Ripeani M., Iamnitchi A., Foster I., IEEE Internet Computing., vol. 6, no. 1, pp. 50-57, Jan./Feb. 2002.

[49] Why Gnutella Cant Scale, No Really Ritter, J, http://www.darkridge.com/ jpr5/doc/gnutella.html.

[50] Measuring and Analyzing the Characteristics of Napster and Gnutella Hosts Saroiu S., Gummadi K., Gribble S. D., published in Multimedia Systems 9, pp 170-184, 2003.

[51] Quantifying Pedophile Activity in a Large P2P System (2011) Latapy et al.,

[52] Availability and locality measurements of peer-topeer file systems Chu J., Labonte K., Levine N., published in ITCom: Scalability and Traffic Control in IP Networks. July 2002, vol. 4868 of Proceedings of SPIE.