

DEPARTMENT OF
APPLIED PHYSICS AND ELECTRONICS
UMEÅ UNIVERISTY, SWEDEN



DIGITAL MEDIA LAB

The analysis of security, protocol and other issues on P2P networks

Ahmad Naeem, Sheikh Amar Shahzad,
Apostolos A. Georgakis
Dept. Applied Physics and Electronics
Umeå University
SE-90187, Umeå Sweden
e-mail: apostolos.georgakis@tfe.umu.se

DML Technical Report: DML-TR-2005:01

ISSN Number: 1652-8441

Report Date: February 6, 2005

Abstract

This technical report is a compilation of material readily available on the web in regard with P2P networks and their vulnerabilities.

Keywords

Peer-to-peer, anonymity, privacy, vulnerabilities.

Contents

1	Peer-to-Peer networking	3
2	Security issues on P2P	4
2.1	Overview	4
2.2	Dangers and vulnerabilities	4
	Theft	5
	Bandwidth clogging and file sharing	5
	Bugs	5
	Trojans, Viruses and Sabotage	5
	Confidentiality	6
	Authentication	6
2.3	Internal threats	6
	Interoperability	6
	Adding and removing users	6
	General security	6
	Distributed Dangers	6
	The People Problem	7
	What is Unethical and Sort of illegal?	7
2.4	Existing security standards for P2P networks	7
	Secret Key Techniques	7
	Public Key Techniques	7
	Asymmetric Key pairs	7
3	Security mechanisms	7
3.1	Secure Sockets Layer protocol	8
3.2	IPSec technologies	8
3.3	Public Key infrastructure	8
3.4	Quantum Key cryptography	8
4	File Sharing and Identity	8
5	Monitoring web traffic	9

6	Secure P2P networks	11
6.1	ANTs network	11
	Query in ANTs network	11
6.2	MUTE	12
	Encryption in MUTE	12
	MUTE and privacy	13
	How MUTE Routes Messages	14
	Encryption	15
	Routing tables	15
	Searching	16
6.3	A Comparison between MUTE and ANTs	16
6.4	Legal protection	17
6.5	FreeNet	17
	FreeNet's current routing mechanism	18
	Next generation routing mechanism	18
	Handling DataNotFound messages	19
	Inherited knowledge	19
	Adapts to network topology	20
	Performance analysis of FreeNet	20
	Advantages	20
	Disadvantages	20
6.6	WASTE network	21
	Network architecture	21
	Security	21
	Projects	22
7	Future of anonymous P2P	23
8	Conclusion	23

1 Peer-to-Peer networking

When accessing the Internet, you access websites through an application known as a browser. Common browser applications are Internet Explorer, Netscape, Mozilla etc. Peer-to-Peer networking, also known as P2P, is similar in concept to a browser. It is an application that runs on your PC and allows sharing of files. Napster used to be one of the most popular P2P application programs for sharing music files until it was shut down by the U.S. Justice Department. Today programs like Limewire, Gnutella, Morpheus, Bearshare, Kazaa, MUTE and BitTorrent share center stage.

Figure 1 depicts the Napster network which uses centralized topology. On the other hand Fig. 2 depicts a second generation P2P architecture, namely the Gnutella network. The differences between a centralized and its decentralized counterpart can be seen between these two figures.

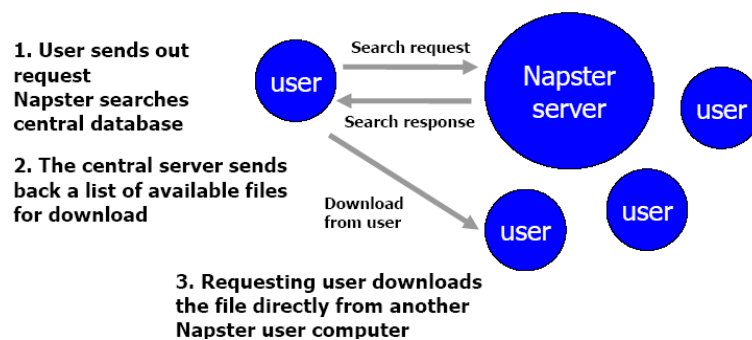


Figure 1: The centralized nature P2P topology in Napster.

The concept of P2P networking is to allow computers to communicate directly with each other, rather than through a central server like a website. Once you have a P2P application installed, you can allow anyone in the world to copy files from your home PC. This can be a single file, an entire directory, or your entire hard-drive. If care is not exercised, your entire hard-drive, including any confidential documents, may be wide-open to anyone in the world.

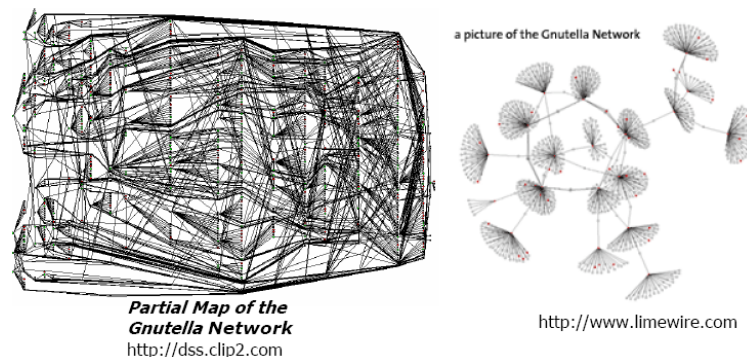


Figure 2: Decentralized P2P networks.

With traditional P2P, file sharers make a direct connection to the computer that is hosting the file they demand. Both parties in the transaction know what file is being transferred. Along with each other's IP addresses. This provides users of the network with no privacy and ultimately makes them vulnerable to lawsuits in many countries.

But lately there has been an increasing interest in anonymising networks. These use clever routing protocols to route data through the network in such a way that nobody knows who released the file onto the network. Or who has requested it. The only information available is the direction it needs to travel to get to its destination. Although a user on the network know the IP addresses of its neighbors, when those neighbors send a file, the

	Research	Commercial	
Business	JXTA .NET Jabber Open Cola Folders	NextPage WorldStreet Groove Jibe Endeavors Entropia DataSynapse	Business
Consumer	Cancer Research Genome Seq Free Haven SETI@home FreeNet Publius Mojo Nation	CenterSpan OpenCola SwarmCast ICQ AIM Kazaa, Morpheus Gnutella LimeWire Bearshare	Consumer
	Research	Commercial	

Figure 3: Classification of P2P applications according to function and audience. (Source: Burton Group)

user does not know if the neighbor is the file's seeder or just a proxy from another user. This is oversimplified in comparison to the complexities of these networks.

From the above is evident that P2P networks can be sub-divided in categories depending on their functionality. Figure 3 depicts a classification of the P2P networks depending on their functionality.

2 Security issues on P2P

2.1 Overview

Security is an essential component of any computer system. It is especially essential for P2P systems. The main points of this are connection control, access control, operation control, anti-virus and of course the protection of the data stored on our machines. The connection, access, and operation control are the priority issues in this report. If one can guaranty secure over the previous issues then the rest of them can follow as well. Figure 4 illustrates all the points that a secure P2P system must address.

2.2 Dangers and vulnerabilities

P2P networking allows a network to be open to various forms of attacks, breaks-in, espionage and malicious mischief. P2P does not bring any novel threats to a network. Just the familiar threats such as worms and virus. P2P networks can also allow someone to download and use copyrighted material in a way that violates intellectual property laws, and to share files in a manner that violates an organisations security policies. Applications such as Napster, Kazaa, Grokster and others have been popular with music-loving Internet users for several years, and many users take advantage of their employers' high-speed connections to download files at work. This presents numerous problems for the corporate network such as using expensive bandwidth and being subject to a virus attack via an infected file download.

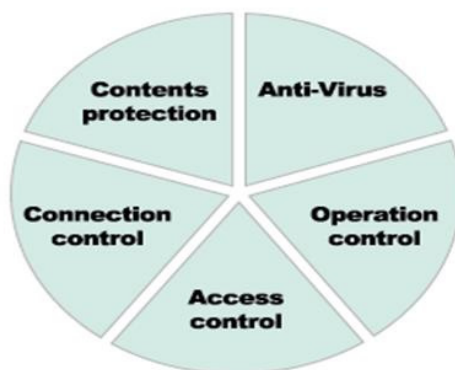


Figure 4: Different issues that needs to be addressed in P2P networks.

Unfortunately, P2P networking circumvents enterprise security by providing decentralized security administration, decentralized shared data storage, and a way to circumvent critical perimeter defenses such as firewalls and NAT devices. If users can install and configure their own P2P clients, all the network managers' server-based security schemes are out of the window.

Theft

Companies can lose millions of euros worth of property such as source code due to disguising files using P2P technologies. P2P wrapping tools, such as Wrapstar (freeware utility) [4], can disguise an archive file like ZIP, containing a company's source code, as an MP3 file.

Bandwidth clogging and file sharing

P2P applications such as Kazaa [3], Gnutella [2] and FreeNet [1] make it possible for one computer to share files with another computer located somewhere else on the Internet. A major problem with P2P file-sharing programs is that they result in heavy traffic which clogs the institution networks.

Bugs

In order for P2P file-sharing applications to work, the appropriate software must be installed on the users system. If this software contains a bug it could expose the network to a number of risks e.g. conflict with business applications or even crash the system.

Trojans, Viruses and Sabotage

A user could quite possibly download and install a booby-trapped P2P application that could inflict serious damage. For example a piece of code that looks like a popular IM or file-sharing program could also include a back door to allow access to the user's computer. An attacker would then be able to do serious damage or to obtain more information then they should have.

P2P software users can easily configure their application to expose confidential information for personal gain. P2P file-sharing applications can result in a loss of control over what data is shared outside the organization. P2P applications get around most security architectures in the same way that a Trojan horse does. The P2P application is installed on a "trusted device" that is allowed to communicate through the corporate firewall with other P2P

users. Once the connection is made from the trusted device to the external Internet attackers can gain remote access to the trusted device for the purpose of stealing confidential corporate data, launching a Denial of Service attack or simply gaining control of network resources.

Confidentiality

Kazaa and Gnutella give all clients direct access to files that are stored on a user's hard drive. As a result it is possible for a hacker to find out what operating system the peer computer has and connect to folders that are hidden shares, thus gaining access to folders and information that is confidential.

Authentication

There is also the issue of authentication and authorization. When using P2P you have to be able to determine whether the peer accessing information is who they really say they are and that they access only authorized information.

2.3 Internal threats

Along with the external threats previously described there are a few internal issues that have to be dealt with.

Interoperability

Interoperability is a major security concern within P2P networks. The introduction of different platforms, different systems, and different applications working together in a given infrastructure opens a set of security issues we associate with interoperability. The more differences in a given infrastructure, the more compounded the security problems.

Adding and removing users

There must be a feasible method to add/delete users to/from the network without increasing vulnerability. The system is under the most threat from users and former users who know the ins and outs of the system e.g. the existence of trapdoors etc.

General security

P2P shares many security problems and solutions with networks and distributed systems e.g. data tampering, unreliable transport, latency problems, identification problems etc.

Distributed Dangers

When using distributed processing applications the user is required to download, install and run an executable file on their workstation in order to participate. A denial of service could result if the software is incompatible or if it contains bugs.

The People Problem

There will always be malicious users who are intent on gaining clandestine access to corporate networks. And no matter what security protocols are put into place a skillful attacker, given enough time, will find a way around them. So all that the security buffs need to do is to keep ahead of the hackers by creating bigger and better protocols. But that's easier said than done!

What is Unethical and Sort of illegal?

There is the violation of *Trade Related Intellectual Property Rights* (TRIPS). The writer, the artist, the company or the person who created a work has the intellectual property of. So sharing it without authorization is always illegal. More over there can be secret data or files. For these issues the supreme court of USA has decided to hear a case from a company's owners facing the threats of leakage of intellectual property.

2.4 Existing security standards for P2P networks

All security mechanisms deployed today are based on either symmetric/secret key or asymmetric/public key cryptography, or sometimes a combination of the two. Here we will introduce the basic aspects of the secret key and public key techniques and compare their main characteristics.

Secret Key Techniques

Secret key techniques are based on the fact that the sender and recipient share a secret, which is used for various cryptographic operations, such as encryption and decryption of messages and the creation and verification of message authentication data. This secret key must be exchanged in a separate out of band procedure prior to the intended communication.

Public Key Techniques

Public Key Techniques are based on the use of asymmetric key pairs. Usually each user is in possession of just one key pair. One of the pair is made publicly available, while the other is kept private. Because one is available there is no need for an out of band key exchange, however there is a need for an infrastructure to distribute the public key authentically. Because there is no need for pre-shared secrets prior to a communication, public key techniques are ideal for supporting security between previously unknown parties.

Asymmetric Key pairs

Unlike a front door key, which allows its holder to lock or unlock the door with equal facility, the public key used in cryptography is asymmetric. This means just the public key can encrypt a message with relative ease but decrypt it, if at all, with considerable difficulty.

Besides being one-way functions, cryptographic public keys are also trapdoor functions- the inverse can be computed easily if the private key is known.

3 Security mechanisms

Mechanisms for establishing strong, cryptographically verifiable identities are very important. These are industry standard authorization protocols that allow peers to ensure that they are speaking with the intended remote system.

3.1 Secure Sockets Layer protocol

For protection of information transmitted over a P2P some networks employ the industry-standard *Secure Sockets Layer* (SSL) protocol. This guarantees that files and events sent will arrive unmodified, and unseen, by anyone other than the intended recipient. Moreover, because both peers use SSL both sides automatically prove who they are to each other before any information is transferred over the network. The protocol provides mechanisms to ensure tamper proof, confidential communications with the right counterpart, using the same, well-proved techniques used by all major website operators to protect consumer privacy and financial information transmitted on the Internet.

3.2 IPSec technologies

Most *Virtual Private Networks* (VPNs) use IPSec technologies, the evolving framework of protocols that has become the standard for most vendors. IPSec is useful because it is compatible with most different VPN hardware and software, and is the most popular for networks with remote access clients. IPSec requires very little knowledge for clients, because the authentication is not user-based. This means a token (such as Secure ID or Crypto Card) is not used. Instead, the security comes from the workstation's IP address or its certificate (e.g. X.509), establishing the user's identity and ensuring the integrity of the network. An IPSec tunnel basically acts as the network layer protecting all the data packets that pass through, regardless of the application.

3.3 Public Key infrastructure

A full-featured X.509 *Public Key Infrastructure* (PKI) over a Secure Sockets Layer (SSL) network backbone - the combination of X.509 PKI authentication and SSL transport encryption is the established cryptographic standard for Internet e-commerce.

Use of X.509 PKI authentication allows security certificates from any other recognized X.509 certificate authority to be used to establish the true identity of any peer device when it comes on-line. Use of SSL point-to-point security encryption enables each pair of peers that communicate with each other to have a unique key for that pairing. The advantage of SSL encryption is that when a peer goes off-line from a community, all its unique pairing keys become invalid, but no pairing keys between other members of the community are affected.

3.4 Quantum Key cryptography

Quantum encryption uses photon state as the key for encoding information. According to the Heisenberg uncertainty principle, it's impossible to discover both the momentum and position of a particle at any given instant in time. Therefore, in theory, an intruder can not discover secret keys based on particle state information. The intruder would need the actual particle to decipher any data encrypted with a key.

Unfortunately this concept is, for the moment, incredibly complex to implement. IBM scientists constructed the first working prototype of a *Quantum Key Distribution* (QKD) system in the late 80's. Back then they could transmit quantum signals just under half a meter through open air. Today, fiber optic cables can transmit the signal up to 50 Km. This still is not very far, but it is definitely good progress. And although we might not see QKD come to market for quite some time, the technology sounds incredibly promising.

4 File Sharing and Identity

Following the death of Napster, all of the file sharing networks that rose to main-stream popularity were decentralized. The most popular networks include Gnutella (which powers Limewire, BearShare, and Morpheus) and

FastTrack (which powers KaZaA and Grokster). The decentralization provides legal protection for the companies that distribute the software, since they do not have to run any component of the network themselves: once you get the software, you become part of the network, and the network could survive even if the parent company disappears.

All of these networks operate as a web or mesh of neighboring node connections. Your node connects to a few other nodes in the network, and those nodes connect to a few other nodes, which in turn connect to a few other nodes, and so on. This layout is similar to a real-life social network: you know people, and those people know other people, who in turn know other people, and so on. A portion of one of these networks can be seen in Fig. 5.

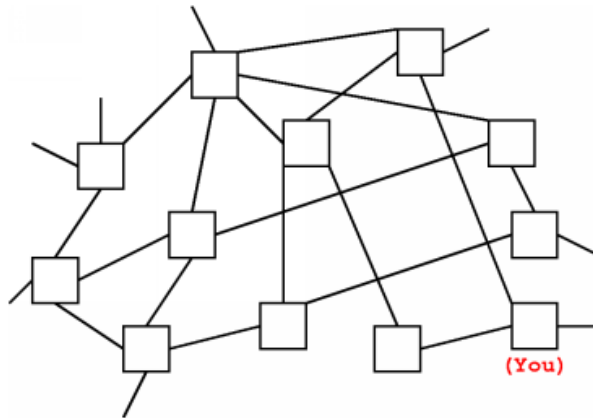


Figure 5: Mesh topology.

When you search for files in the network, you send a search request to your neighbors, they send the request on to their neighbors, and so on. Eventually, your request reaches many nodes in the network. For example, you might send out a search for `metallica AND mp3`. Lots of nodes receive your request but only a few of them are actually sharing any metallica music. Those that do have matches send their results back to your node. These results look something like this:

<i>My Address</i>	<i>My File</i>
128.223.12.122	Metallica_Enter_Sandman.mp3
128.223.12.122	Metallica_Unforgiven.mp3
128.223.12.122	Metallica_Everywhere_I_Roam.mp3

Suppose that the blue node in Fig. 6 is the node at 128.223.12.122 that returned the three Metallica results. To download a file from the blue node, your node makes a direct connection to it using the address 128.223.12.122. After your node connects to the blue node, the blue node knows your computer's Internet address as well, say 113.18.92.15 (going back to the phone analogy, this is like the blue node using caller ID). The blue node sends the Metallica file to you over this connection, and then you close the connection. This connection, which is separate from the other neighbor connections in the network can be seen in Fig. 7.

5 Monitoring web traffic

Your *Internet Service Provider* (ISP) provides you with your IP address. An ISP knows who is using each IP that it gives out. In general, your ISP will keep your identity private. So, though the person sharing Metallica might contact your ISP and ask, who is using 113.18.92.15?, your ISP will likely keep its lips sealed.

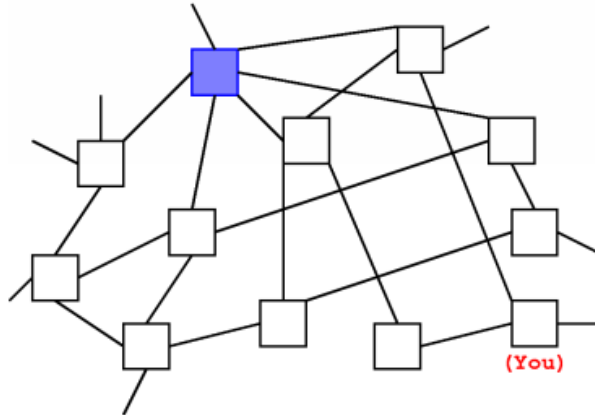


Figure 6: The node that holds the file under consideration.

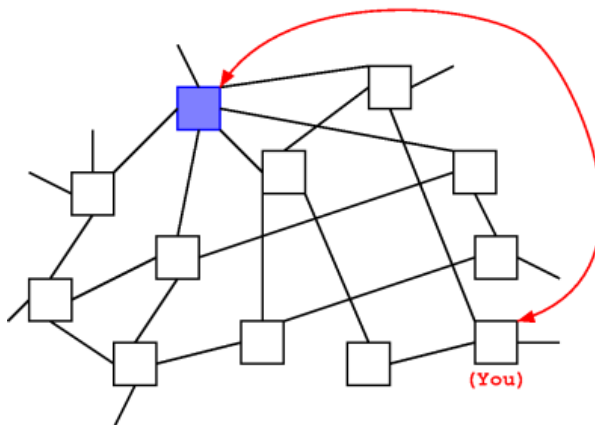


Figure 7: Direct link between two nodes.

Suppose that you are sharing a large collection of your favorite music, and assume that your collection contains more than 1000 songs. Also, suppose that most of the songs in this collection are owned by record labels that are represented by the RIAA. When someone searches for mp3 in your file sharing network, your node returns a lot of results. Now suppose that one of the nodes in the network happens to be owned by the RIAA as seen in Fig. 8.

The RIAA performs a search in the network for songs that it cares about. Since RIAA record labels own the vast majority of music that is published in throughout the world, we can simplify things by assuming that the RIAA cares about most songs. Thus, the RIAA performs a search for mp3, and your node returns over 1000 results, which look something like this:

113.18.92.15	Madonna_Holiday.mp3
113.18.92.15	Fleetwood_Mac_Dreams.mp3
113.18.92.15	Journey_Faithfully.mp3
113.18.92.15	Bonnie_Raitt_Something_To_Talk_About.mp3
113.18.92.15	Poison_Unskinny_Bop.mp3

Though an average file sharing user would usually initiate a download in response to this gold mine of search results, the RIAA has all the information that it needs, so it stops right here. With the list of 1000+ infringing songs

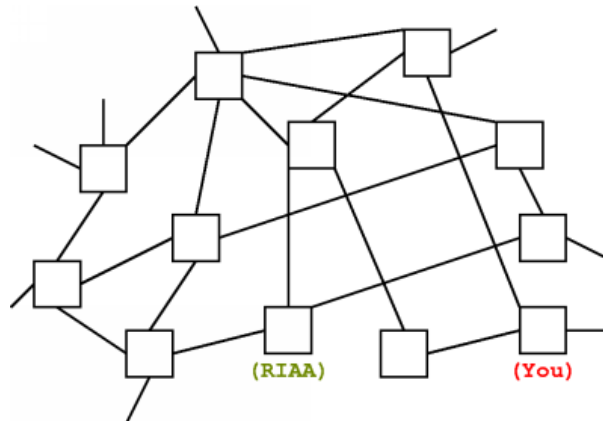


Figure 8: A RIAA node is listening.

in hand, it files a subpoena against your ISP and demands that your ISP hand over your personal information. Once the RIAA has your personal information, it is ready to file a lawsuit against you for copyright infringement.

6 Secure P2P networks

6.1 ANTs network

In ANTs a connection is not anymore point to point in the strict sense. The peers are virtual peers over a virtual net, so when we are requiring a resource over the net, our request is routed through many points until it reach our peer. The peers are not anymore identified by IP. They have a unique ID produced hashing infos from their time and their location (this yields a unique hash). So a client now knows only the IPs of its neighbors (the other peers directly connected to it), but it does not know their ID, as only the same node knows its ID.

So what about routing, how can a node route a message if it does not know where the destination is? The answer is simple. A node will know the “best” direction to route a message to, but it will not know where precisely another node is. The routing protocol has been developed over studies on ants behavior.

Ants do not know the precise location of their hive; they simply follow a track. The same happens in this system. So the more messages follow a track the more that track will be strong, if a track produces many failures it'll fade out and it will not be followed anymore. This way we can achieve privacy over our identity, but what about the information sent? They have to route through many peers so how can we protect them? The protection is realized at two levels. In low level (against man-in-the-middle) by encrypting communication between each couple of directly linked points of the net and in high level (against internal threats) encrypting the communication between the two end points. At both level the security is granted using a DH-KA and DES or AES (negotiated at the beginning). Figure 9 depicts the main configuration panel of ANTs.

Query in ANTs network

Each query is distributed (in a non deterministic and sequential way) over a part of the net. It is processed by each node it passes and at last it is returned to the source following the shortest path. Each node can process operations more complex than the simple text-matching.

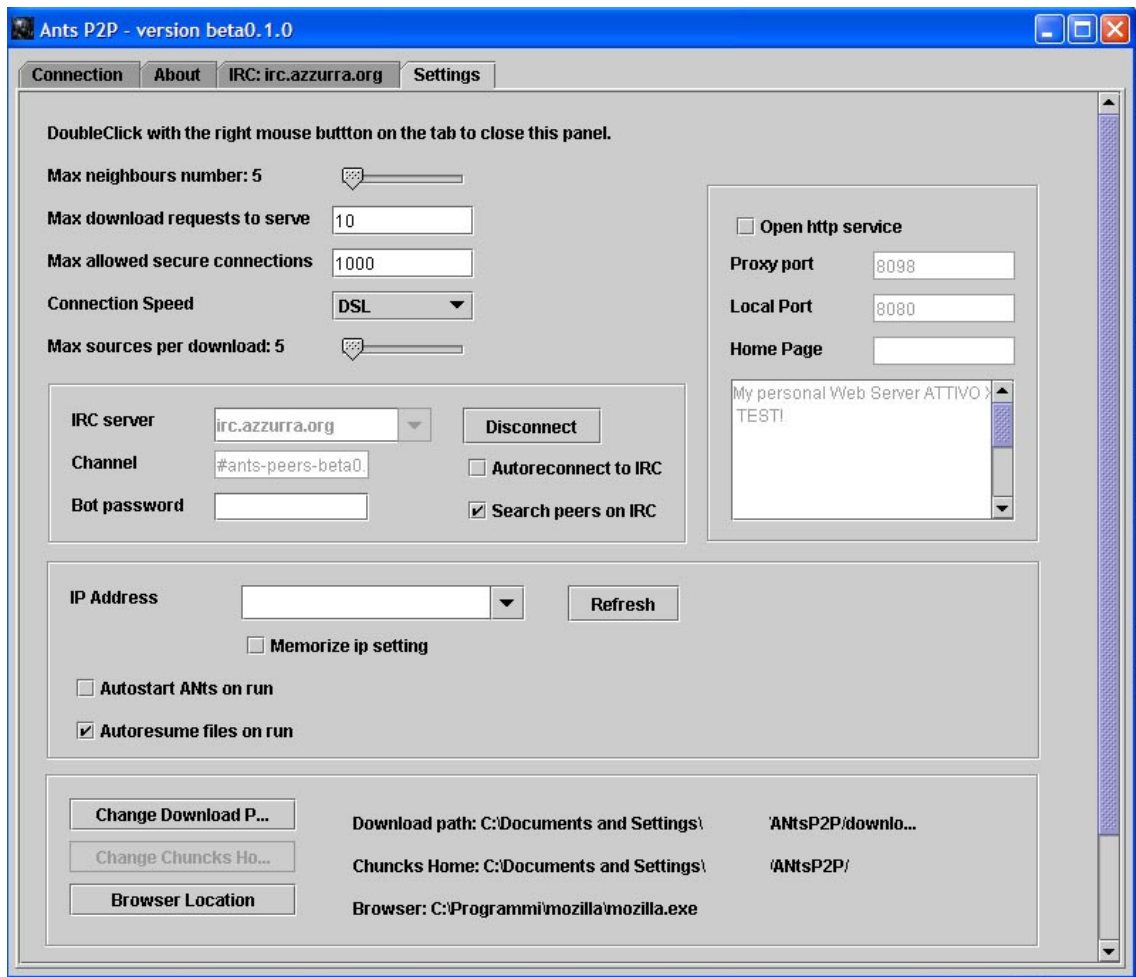


Figure 9: The main configuration panel of ANTs.

6.2 MUTE

MUTE is a new file sharing network that provides easy search and download functionality while protecting your privacy. It does this by routing all messages through a network of neighbor connections, using virtual addresses and encrypting all the traffic using RSA for public/private keys and AES for the actual encryption. MUTE implements the same idea as of ANTs using different routing protocols and discovery systems. Also the security policy is different as no endpoint secured connections are used (it uses only point to point secured connections). Main problems of MUTE are the lack of multiple downloads and of the resume system. Sometimes it can also result too slow even if it is used with a broadband connection (probably due to wrong routing policies).

Encryption in MUTE

MUTE nodes use encryption, but only to secure the direct connections to their neighbors. This might be described as link encryption. For a given MUTE route, each end of the route is encrypted separately. When a node receives a message from one of its neighbors, it essentially decrypts the message, reads it, and then re-encrypts it to route it onward through another neighbor.

For a network that might provide privacy along with anonymity, this seeming lack of security sounds like a

major problem. How can we fix this problem? In an anonymous network like MUTE it is impossible to fix this problem in any real way. Over the course of MUTE's development, various people have suggested (and even demanded) an end-to-end encryption to MUTE. The basic idea is that a sender would encrypt the contents of a message with the receiver's key before routing it through the MUTE network. Since only the receiver could decrypt the message contents, this kind of encryption would thwart any attempts by intermediary nodes to read or modify the message in-route.

There are many possible variations in terms of schemes for enabling end-to-end encryption, but the fatal flaw affects all of them. To establish a secure channel, you must exchange keys (or other information) somehow. As long as you are passing this channel-setup information through the MUTE network, intermediary nodes can interfere and defeat the security of the resulting end-to-end encryption. Of course, proponents of end-to-end encryption in MUTE are insistent, and new schemes pop up from time to time that supposedly solve the man-in-the-middle problem.

MUTE and privacy

The main way that MUTE protects your privacy is by avoiding direct connections between downloaders and uploaders. By using the network to route search requests, these networks deliver a particular request to many nodes without making direct connections to any of them. Of course, when it comes to transfer a file, these networks use direct connections.

MUTE routes all messages, including search requests, search results, and file transfers, through the network of neighbor connections. Thus, though you know the Internet addresses of your neighbors, you do not know the Internet address of the node you are downloading from.

A map of a MUTE network looks identical to the maps of standard networks shown earlier. If you perform a search for `metallica AND mp3`, you still might receive back three results, but these results look a little different:

My Address	My File
7213D...2DCA5	Metallica_Enter_Sandman.mp3
7213D...2DCA5	Metallica_Unforgiven.mp3
7213D...2DCA5	Metallica_Everywhere_J_Roam.mp3

Notice that the `My Address` portion of these responses no longer contains an IP address. The address shown, is `7213D29781593840CF00CDD1E9A7A425AE16DCA5`. This is a MUTE virtual address. Each node in the MUTE network has a virtual address that it generates randomly each time it starts up. Your neighbors in the network (those nodes that actually do know your Internet address) do not know what your virtual address is, so no one in the network can connect your virtual address to your Internet address, and thus no one can obtain your real-world identity.

MUTE uses virtual addresses to route messages through the network using the technology behind ANTS. Thus, to download a `metallica` file, your node would send a download request through the network to `7213D...2DCA5`, and your node would mark that request with your own virtual address, say `D1E9A59380CD425AE16D40CF0CA57A7213D29781`. The node sharing `metallica` would send the requested file back to you using your virtual address. The entire transfer is routed through the network and can be seen in Fig. 10.

Though the transfer is routed through a node owned by the RIAA, all the node sees are the virtual addresses of you and your file sharing partner. The RIAA can send out a search for `mp3`, and it will still get back 1000 results from you, but these results would look like this:

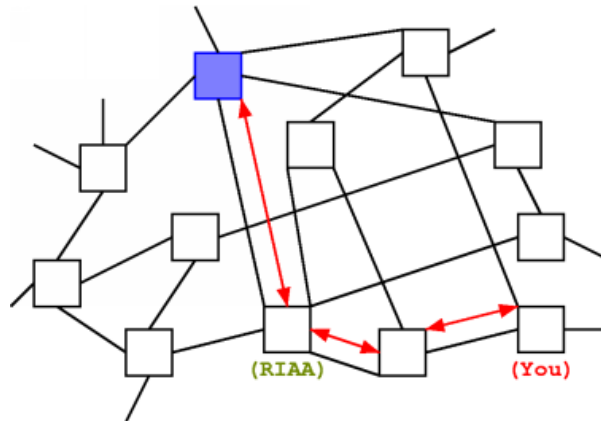


Figure 10: The RIAA node is monitoring traffic.

My Address	My File
D1E9A...29781	Madonna_Holiday.mp3
D1E9A...29781	Fleetwood_Mac_Dreams.mp3
D1E9A...29781	Journey_Faithfully.mp3
D1E9A...29781	Bonnie_Raitt_Something_To_Talk_About.mp3
D1E9A...29781	Poison_Unskinny_Bop.mp3

How MUTE Routes Messages

In the network we have messages that need to travel from a sender to a recipient. Since MUTE users are anonymous, none of the nodes in the network know exactly where to find a particular recipient. Like ants that are unaware of the overall environment layout, MUTE messages must be directed through the network using only local clues.

Each MUTE node maintains connections to several neighbors in the network, and these neighboring connections are used for message passing. Suppose that MUTE node *X* receives a message from *Alice* to *Bob* through node *Y*. *X* may have no information clues about where *Bob* is in the network. However, upon receiving this message, node *X* learns something about *Alice*: it learns that messages from *Alice* come through node *Y*. In the future, if node *X* ever receives a message to *Alice*, it can send it back through node *Y* using this clue.

Regardless of what *X* learns about *Alice*, it still has no information about *Bob*. The best strategy here, using ants as inspiration, is to send ants in all directions, or to send a copy of the message on to each one of *X*'s neighbors. One of the neighbors may have more information about which direction *Bob* is in. If none of the nodes in the network have clues about *Bob*'s location, they will all broadcast the message to their neighbors. If *Bob* exists in the network, this technique will eventually find him.

Notice that throughout the search for *Bob*, the message has been leaving a trail of clues about *Alice*. If the message reaches *Bob*, and then *Bob* sends back a response, the response can follow these clues on a rather direct path back to *Alice*.

As the response is routed to *Alice*, it leaves a trail of clues that can be used to route future messages from *Alice* back to *Bob*. Other nodes can make use of these clues too. For example, if the owner of node *X* sends a message to *Bob*, the message will travel on a rather direct route using the existing clues.

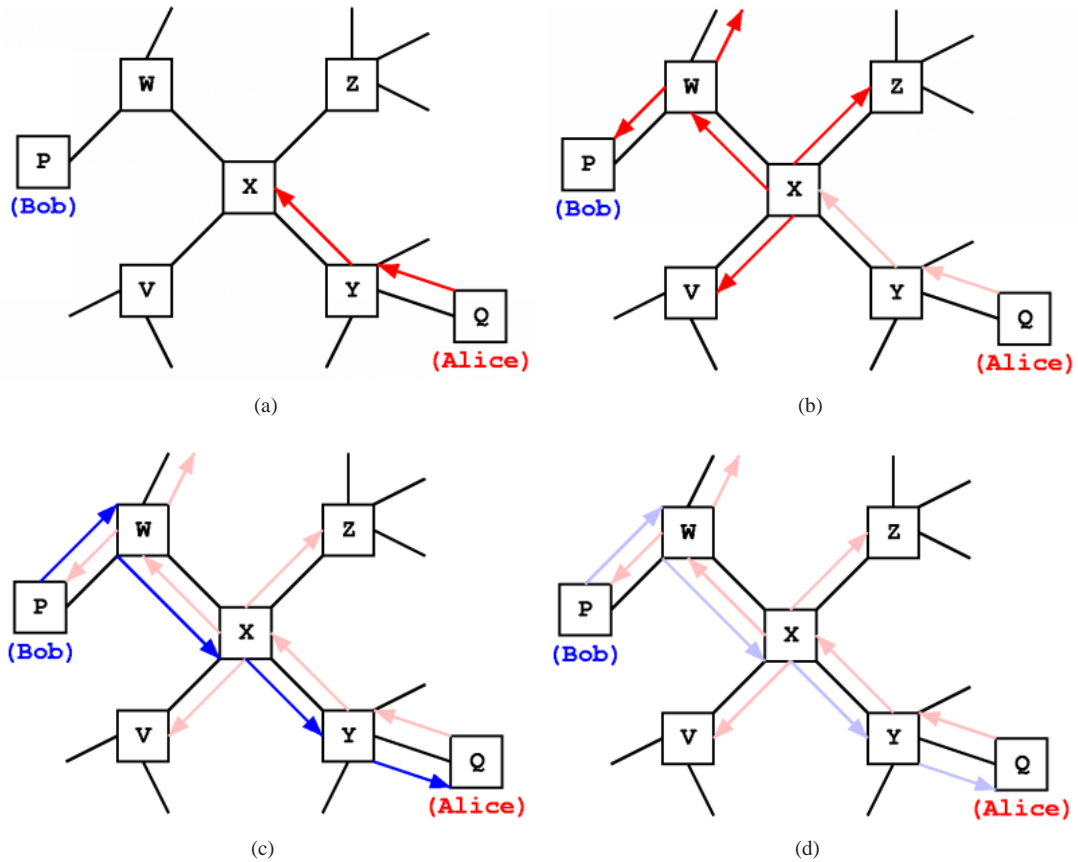


Figure 11: Routing protocol in MUTE. (a) Alice sends a query to the network, (b) The query is being propagated, (c) Bob replies to Alice query, (d) Communication between Bob and Alice is finally established.

Encryption

Neighbor connections in the MUTE network are built on top of secure streams that use:

- RSA public/private keys (size selected by user at runtime) to exchange secret keys.
- AES 128-bit secret keys used in CFB mode with all-zero initialization vectors to encrypt stream data.
- Separate AES keys used for each stream direction
- Fresh AES keys used each time a new stream is established

Routes in the MUTE network are not encrypted end-to-end. Because of person-in-the-middle attacks during a routed key-exchange, secure end-to-end encryption is impossible in an anonymous network.

Routing tables

MUTE’s routing algorithm uses routing tables to track which neighbor connections are associated with particular sender addresses. For example, we might receive a series of messages from *Alice*, and these messages might be sent to us through three out of our five neighbor connections. When we receive messages addressed to *Alice*, we should route them back through the neighbors associated with messages from *Alice*. Given that we might receive

messages from *Alice* through more than one neighbor, various algorithms can be used to decide which neighbor to send a particular message through.

MUTE uses a probabilistic algorithm for back route selection. For each recently seen sender address, MUTE keeps a queue of pointers to recent neighbor connections on which we have received messages from that sender. For example, we might track the last 100 neighbors that sent us messages from *Alice*. If we are tracking three sender addresses (*Alice*, *Bob*, and *Eve*), and we have five neighbors (numbered 1 to 5), our routing table might look like this:

From	Alice	Bob	Eve
1	5	4	
2	1	4	
2	2	4	
1	1	3	
4	1	4	
1	5		
1	5		
2			
2			
1			

Looking at the pattern in this table, we can see that most messages from *Eve* came to us through neighbor 4. Our best bet when routing messages back to *Eve* would be to send them through neighbor 4. Briefly neighbor 4 has the strongest scent from *Eve*.

If we receive a message addressed to *Alice*, we select *Alice*'s column from the table, select one of the neighbor entries uniformly at random, and route the message through that neighbor. This scheme prefers to route messages through the most popular paths, but occasionally routes messages through less popular paths.

Searching

MUTE uses distributed search based on controlled flooding to locate files by name based on free-form query strings. While many other P2P networks control flooding using a TTL scheme, MUTE introduces a more effective and scalable mechanism called Utility Counters.

6.3 A Comparison between MUTE and ANTs

MUTE provides very low-level privacy. Although the information traveling between individual nodes is encrypted, each node has full access to the data traveling through the network.

This contrasts with ANTs, which also encrypts data sent between the source node and the destination node. This is called endpoint encryption. Theoretically, endpoint encryption means proxy nodes cannot identify the data they proxy.

To obscure the files being transferred so that proxies cannot identify them AES is used. To be able to do this, the two file sharers must have a secret code (or "key") that only they know. For them both to know the key, the Diffie-Hellman (DH) Key Exchange Protocol is used. However there are some questions over the implementation of DH in ANTs. The security of the system relies on this key exchange, but anyone could potentially launch a man-in-the-middle attack. This attack would allow an adversary to obtain the secret key and see the file being transferred.

This is usually combated with trusted third party involvement using "certificates". These certificates are not implemented in the ANTs network. To a determined node, ANTs therefore provides no extra information security.

However, there are still some concerns: Implementing a certificate system is not a simple task. You need to be able to create, issue and revoke certificates, and you need a party that you trust to do this securely. You also remove anonymity, and may even introduce the concept of non-repudiation. This means that after sharing a file, you can not later deny that you were the source.

Ultimately this raises further questions of the use of an anonymising system between trusted peers. Indeed, if you trust them, why not make a direct connection? However, there are many different methods of employing certificates. To discuss and analyze even one system would be a report in itself.

Whatever structure is used for trading amongst trusted peers, we know this is not how mass scale P2P networks operate. Users mainly communicate with untrustworthy peers, existing only as an IP address. Therefore, ANTs cannot promise security between these majority transfers, due to lacking certificates.

6.4 Legal protection

Legal protection is increasingly what file sharers are caring about when choosing a file sharing application. Numerous music lovers have fallen victim to lawsuits against file sharers. With over 4500 such cases in America, along with a mass of cease and desist letters worldwide, protection from the prying eyes of organizations who think they are the law becomes ever more important.

In anonymising networks, it is impossible to detect who seeded a file. The same routing mechanism that provides anonymity also provides legal protection. As files make their way through the network, no peer can know if the file they are receiving has come from the original source or a proxy.

This system destroys the current targeting method used by organizations that object to the sharing of certain files. In traditional file sharing systems, file sources are tied to an IP address. But it is this attachment of IP address and file that makes targeting users sharing copyright works, for example, such an easy process. In anonymising networks, files are tied only to a virtual address.

Any two computers that are directly connected in a network are called neighbors. Neighbors must know each other's IP addresses in order to keep the connection. As already established, when neighbors send each other files, the receiver does not know if their neighbor is the original seed, or a proxy. However, they do know their IP address and what has been sent.

ANTs offers an additional element of security due to proxying. Proxies are therefore in a weak position, as any peer can catch their neighbor proxying an objectionable file. The legal implications of being a proxy in a file sharing network are unclear. No court has ever considered whether you can be held liable for copyright infringement simply for proxying data in a network. EFF strongly believes that the answer should be that, assuming you have no knowledge of what the packets are that you're passing, you should not be liable for the contents of the packets. That is, after all, the rule for ISPs. The same rule should apply for individuals.

If the law favors the EFF view, then neither MUTE nor ANTs peers can be held liable for proxying objectionable files. Although ANTs provides endpoint encryption, the average user of MUTE would not know where to begin discovering what they are proxying. The overhead of endpoint encryption provided by ANTs would therefore be wasted in a legal context.

6.5 FreeNet

FreeNet is free software which lets you publish and obtain information on the Internet without fear of censorship. To achieve this freedom, the network is entirely decentralized and publishers and consumers of information are anonymous.

Communications by FreeNet nodes are encrypted and are routed-through other nodes to make it difficult to determine who is requesting the information and what its content is. Users contribute to the network by giving bandwidth and a portion of their hard drive (called the data store) for storing files. Unlike other P2P file sharing

networks, FreeNet does not let the user control what is stored in the data store. Instead, files are kept or deleted depending on how popular they are, with the least popular being discarded to make way for newer or more popular content. Files in the data store are encrypted to reduce the likelihood of prosecution by persons wishing to censor FreeNet content. The network can be used in a number of different ways and is not restricted to just sharing files like other P2P networks. It acts more like an Internet within an Internet.

FreeNet's current routing mechanism

Every node in the FreeNet network is required to provide a simple service to other nodes in the system. When a node receives a request for a particular key, it should do its best to retrieve the data corresponding to that key as quickly as possible and send it back to the node that requested it.

In the simplest case, the recipients of the request will already have the data cached locally, and can immediately send it to the requester. In most cases, however, the node will need to request the data from another node in the network, which it thinks, will be better able to retrieve the data. The way FreeNet makes this decision forms the core of the FreeNet algorithm. At present, the algorithm used to select which node should be consulted for a given key is relatively simple.

In short, when a node forwards a request for a particular key to another node in the network, and that node is successful in retrieving the data is called the *DataSource*. The requester makes a note of the requested key, and the *DataSource* node passed back with that reply. It is assumed that the upstream node is a good place to route future requests for keys closest to the previously requested key. It is analogous to deciding that since your friend *Bob* successfully answered a question about *Alice*.

Despite its simplicity, this approach has proved surprisingly effective, both in simulation and in practice. One of the expected side effects of this approach was that nodes would tend to specialize in the retrieval of some keys to the exclusion of others. This can be seen is somewhat analogous to the way that people specialize in particular areas of expertise. This effect has been observed in actual FreeNet nodes deployed in the FreeNet network. Figure 12 represents the keys stored by an actual FreeNet node. The X axis represents the key space, at the left is key 0, across to the right which is key 2^{160} . Dark areas indicate where the node has better knowledge. The dark strips indicate areas in which the node has detailed knowledge about where requests for those keys should be routed.

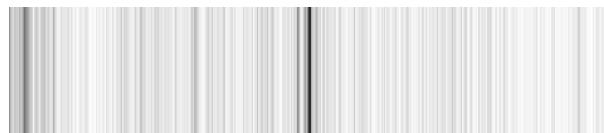


Figure 12: The keyspace in FreeNet.

Note that when the node was first initialized keys would have been evenly distributed across the key space. This is a good indication that FreeNet's current routing algorithm is performing correctly. Nodes specialize like this due to an emergent feedback effect, when a node successfully responds to a request for a given key - it increases the likelihood that other nodes will route requests for similar keys in the future. Over time this effect causes the specialization that can be seen very clearly in the diagram above.

Next generation routing mechanism

The fact that it works, of course, does not mean that it cannot be improved upon. At the simplest level, note that the current algorithm does not distinguish between a slow FreeNet node sitting at the end of a slow modem line in the remote Australian outback, and a powerful node connected to a T3 in downtown Los Angeles. Also note that while the current algorithm works, the only real way to test improvements to the algorithm is to see how they affect a large scale network, either in simulation, or in the real world. By making more effective use

of the information available to a FreeNet node, we can dramatically improve a node's ability to route requests in a manner likely to result in the fastest response for that request. These goals are achieved by Next Generation Routing.

The core idea behind the Next Generation Routing design is to make FreeNet nodes much smarter about deciding where to route information. For each node reference in its routing table, a node will collect extensive statistical information including response times for requesting particular keys, the proportion of requests which succeed in finding information, and the time required to establish a connection in the first place. When a new request is received, this information is used to estimate, which node is likely to be able to retrieve the data in the least amount of time, and that is the node to which the request is forwarded.

Handling DataNotFound messages

When a request has visited the number of nodes specified in its hops to live field (similar to TTL), a *DataNotFound* (DNF) message is returned to the requester. This indicates that the data could not be found within the time limit specified by the requester. There are two reasons that a DNF can be returned for some data, either the data exists but was not found, or the data did not exist at all. In the former case, a DNF would indicate a shortcoming in the routing ability of whichever node the request was routed to. In the latter case, it would not - but the difficulty is that for a given DNF - there is no easy way to tell what type of DNF it is, whether it is legitimate or not.

Let, us, for a moment, assume that there was a way to identify illegitimate DNFs. In this case, the cost in terms of the time required for such a DNF would be the time required to receive the DNF plus the time required to request the same data from somewhere else. We can estimate the former by looking at how long previous DNFs took for requests sent to this node. We can estimate the latter by looking at the average amount of time it takes to successfully retrieve some data.

Now, imagine a FreeNet node with perfect routing, the only DNFs it returns would be legitimate - since if the data were in the network, it would find it. The proportion of DNFs this perfect node returned would be the same as the proportion of DNFs for which the data simply did not exist in the network. Now - such a node (we assume) could not exist, however we can approximate it by looking at the proportion of DNFs for the node with the lowest proportion of DNFs in our routing table.

So now, we can determine the time cost of DNFs, and we can also approximate what proportion of a node's DNFs are legitimate - and therefore should not incur a time cost. We can therefore add an estimated routing time cost for each node to account for DNFs.

Inherited knowledge

One of the problems observed in the current FreeNet is the time required for a FreeNet node to establish sufficient knowledge about the FreeNet network to route efficiently. This is particularly damaging to FreeNet's usability as people's first impressions of software is critical, and the first impressions of FreeNet are generally the worst impressions of FreeNet as they are formed before the FreeNet node can route requests effectively.

The solution is to employ some qualified trust between FreeNet nodes, allowing them to share the information they have collected about each other, albeit in a rather untrusting way. There are two ways that a FreeNet node finds out about new FreeNet nodes. The first is when they first start up - they load a `seed nodes.ref` file, which contains the routing expertise of another experienced node.

The other way nodes learn about new nodes is in the *DataSource* field of successful replies to requests for data. The *DataSource* field will contain one of the upstream nodes in the request chain. The simple approach would be to allow this *DataSource* node to attach statistical information concerning its own performance to the reply - but clearly this would be open to abuse. A refinement would be to say that any node passing back a reply which has collected its own statistical information about the node in the *DataSource* will replace the statistical

data in the reply with its own. This will mean that even if a node does put misleading statistical information in the reply - it will probably be replaced as it is passed back to the requester.

Adapts to network topology

In the old FreeNet routing algorithm, FreeNet ignores the underlying Internet topology as all nodes are treated equally. In contrast, next generation routing bases its decisions on actual routing times, this means that a node will tend to prefer routing messages to faster nodes, on better Internet connections that are geographically closer - unless those nodes become overloaded, which will decrease the incentive to use them and have a load balancing effect.

Performance can be evaluated locally with the old approach to FreeNet's routing; the only concrete way to evaluate its performance was by trying it. With next generation routing we have a simple metric of how effectively it is performing - namely the difference between estimated routing times, and actual routing times. If a modification to the algorithm results in closer estimates, then we know it is better. If not, we know that it is not. This will dramatically accelerate the development cycle of future improvements.

Performance analysis of FreeNet

Problems which affect the performance of FreeNet are:

1. In network communication, connection speed dominates processor and I/O speed as the bottleneck. This problem is emphasized by the highly parallel nature of FreeNet.
2. As there is no central master index maintained, messages must be passed over many hops, in order to search through the system to find the data. Each hop not only adds to the total bandwidth load but also increases the time needed to perform a query. If a peer is unreachable it can take several minutes to time out the connection.
3. P2P communities depend on the presence of a sufficient base of communal participation and cooperation in order to function successfully.

Advantages

1. FreeNet is solving many of the problems seen in centralized networks. Popular data, far from being less available as requests increase, becomes more available as nodes cache it. This is the correct reaction of a network storage system to popular data.
2. FreeNet also removes the single point of attack for censors, the single point of technical failure, and the ability for people to gather large amounts of personal information about a reader.
3. FreeNet's niche is in the efficient and anonymous distribution of files. It is designed to find a file in the minimum number of node-to-node transactions. Additionally, it is designed to protect the privacy of the publisher of the information, and all intervening nodes through which the information passes.

Disadvantages

1. It is designed for file distribution and not fixed storage. It is NOT intended to guarantee permanent file storage, although it is hoped that a sufficient number of nodes will join with enough storage capacity that most files will be able to remain indefinitely.

2. FreeNet does not yet have a search system, because designing a search system which is sufficiently efficient and anonymous can be difficult.
3. Node operators cannot be held responsible for what is being stored on its hard drive. FreeNet is constantly criticized because you have to donate your personal hard drive space to a bunch of strangers that may be very well use it to host content that you disapprove of.
4. FreeNet is designed so that if the file is in the network, the path to the file is usually short. Consequently, FreeNet is not optimized for long paths. Long paths can therefore be very slow.
5. Self-organizing file sharing systems like FreeNet are affected by the popularity of files, and hence may be susceptible to the tyranny of the majority.

6.6 WASTE network

WASTE is an anonymous, secure, and encrypted collaboration tool, which allows users to both share ideas through the chat interface and share data through the download system. WASTE is RSA secured, and has been heralded as the most secure P2P connection protocol currently in development.

WASTE is a software product and protocol that enables secure distributed communication for small (on the order of 10-50 nodes) trusted groups of users. It is designed to enable small companies and small teams within larger companies to easily communicate and collaborate in a secure and efficient fashion, independent of physical network topology.

WASTE creates a network of hosts, making whatever connections possible, and typically routes traffic via the path of lowest latency (which effectively ends up as load-balancing, though it is far from ideal). With at least one host outside of firewalls (or behind a firewall but having one incoming port open), a WASTE network can enable all supported services (including chat and file transfer) between any two hosts.

WASTE currently provides the following services:

- Instant Messaging (with presence)
- Group Chat
- File browsing/searching
- File transfer (upload and download)

Network architecture

WASTE uses a distributed architecture that allows for nodes to connect in a partial mesh type network. Nodes on the network can broadcast and route traffic. Nodes that are not publicly accessible or on slow links can choose not to route traffic. This network is built such that all services utilize the network, so firewall issues become moot.

Security

WASTE secures the links of the WASTE network by using RSA to exchange session keys and authenticate the other end of the connection. Once the hosts have authenticated each other and both have the correct session keys, the connection is encrypted using Blowfish in PCBC mode.

Both sides exchange session keys and challenge-response tokens encrypted with each others public keys. Both sides decrypt and verify the challenge-response tokens, and begin encrypted communication (a stream of messages, each message is verified using an MD5). There's a lot more to it than that, but that's the basic idea.

The reality of it is that there is also a Network ID/Name feature that allows you to easily keep networks from colliding, as well as efforts to obfuscate the whole process (to make WASTE connections difficult to detect). Another unique feature is the way session keys are exchanged and combined so that in order to decrypt past (recorded) traffic, both private keys of a connection need to be recovered.

Since WASTE requires a small trusted network to function efficiently, it benefits greatly from cryptography. Using public-key encryption for session key negotiation and user authentication allows both the prevention of unknown users from joining the network as well link data security to prevent unknown users from “sniffing” network traffic.

WASTE also provides for an additional “network name or ID” that can be used to secure a network against people who do not have the name or ID. This can be useful if you wish to easily prevent multiple networks from merging, or change it to easily remove access of user(s) without having to make everybody ban those user(s) public keys.

Projects

There have been many projects inspired by WASTE and modify WASTE in some way. The following list is just an example:

- modWASTE - The modWASTE project is used by WASTE developers to experiment with modifications that may be added to the main WASTE source tree. These modifications are experiments for WASTE 2. modWASTE may eventually contain features not available in WASTE.
- jWASTE - WASTE is a software protocol that enables secure distributed communication for small trusted groups of users. This project will extend the original development release and maintain a Java-based client.
- PHPWasteSrv - This projects aims to create a web administration front end for Waste. Allowing configuration of all aspects of a WasteSRV. Key Management, Host Management, System Config.
- FreeWASTE - The freewaste project aims to create a cross platform library that will help a developer create applications that take part in a mesh-like distributed network. This network is a set of nodes communicating in RSA (using openssl) public key encryption.
- WASTE-Gtk2 - A GTK2 Implementation of the WASTE protocol - encrypted chat, file sharing and instant messaging.
- MUTE plus WASTE - This program combines mute and waste as a hybrid.
- WASTED- A new portable implementation of WASTE.
- phWASTE - phWASTE is a User-Management-System inspired by WASTE: A user can save all informations required to connect on his/her WASTE-Client. Other users can see those infos and decide to connect or not. phWASTE uses PEAR, MySQL and the SmartyTemplateEngine.
- taint - Implementation of the WASTE P2P client that focuses on cross platform capability with a more robust trust model.
- TheRing - TheRing is a P2P tool inspired by and compatible with but not based on Nullsoft Waste featuring secure chats and file-exchange for closed user groups.

Some Self Experienced Conclusions

1. The above P2P networks seem to be secure but no one is taking care of the security holes so they are not much secure even.

2. The messages wander in the networks and any one can read it or sniff it.
3. The messages are not exactly and managed pointed to some definite route, so the inside of the network is not efficient enough.
4. The protocols are not strong and refined, as lets see in BGP (Border Gate way Protocols) to communicate between diff AS (Autonomous Systems) of the world, there are definite ways and costs of the paths and the lowest cost paths are always preferred. More over there are route reflectors and confideration schemes to reduce the mesh in side the network and not every router is communicating with the other one. The P2P networks are clearly illegal and that's why they always talk for the anonymity of the points.
5. These P2P networks are clearly against the copyright laws and the laws of TRIPS, trade related Intellectual Property Rights. Where the owner of a music or video is actually paying the artist and then becomes the owner of the art or performance of the artist. So getting these videos before paying the owners is the clearly a theft.
6. P2P networks make other secure networks insecure, by penetrating in them and many times on the account of the interest of the person sitting inside the secure network and to get the files and downloading them.

7 Future of anonymous P2P

File sharers swapping the latest pop-track is one thing, but less savory characters may use the system to commit crimes. Pedophiles for example could use the system to hide their activities. Users of the network would unknowingly be proxying child pornography. As networks become more anonymous, further problems will arise. Anonymity is also provided to those fighting the development of P2P technology. Data and information can be corrupted in transit.

8 Conclusion

The attacks detailed above illustrate that is it is currently possible to defeat all of the security claims designers have made about FreeNet. Of course, the system is a work in progress and many enhancements remain to be implemented. FreeNet designers are strongly encouraged to widely publish their documentation of the protocol and future enhancements to encourage security reviews. The highest priority should be given to authenticating connections between nodes and more thought must be given to mechanisms for node discovery. To defeat traffic analysis, designers can implement a limited form of onion routing or concentrate on building gateways to existing anonymous communication networks designed with these goals in mind.

References

- [1] Freenet. <http://freenet.sourceforge.net>. (Accessed 051210).
- [2] Gnutella. <http://gnutella.wego.com>. (Accessed 051210).
- [3] Kazaa. <http://www.kazaa.com>. (Accessed 050810).
- [4] Wrapster. <http://members.fortunecity.com/wrapster>. (Accessed 051210).