Peer to Peer Systems

L Jean Camp*

Associate Professor of Public Policy

Harvard

Cambridge, MA 02138

www.ljean.com

Glossary

Cluster – a set of machines whose collective resources appears to the user a single resource

Consistency – information or system state that is shared by multiple parties

Domain name – a mnemonic for locating computers

Domain name system – the technical and political process of assigning, using, and coordinating domain names

Front end processor – the first IBM front end processor enabled users to access multiple mainframe computers from a single terminal

Metadata – data about data, e.g., location, subject, or value of data

* **This work was supported by the National Science Foundation under Grant No. 9985433 and a grant from the East Asia Center. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.**

Persistent – information or state that remains reliably available over time despite changes in the underlying network

Reliable –the systems maintains performance as a whole despite localized flaws or errors; e.g., file recovery despite disk errors, file transmission despite partial network failure. Alternatively a system (particularly a storage system) in which all failures are recoverable, so that there is never long term loss.

Scalable –a system that works within a small domain or for a small number of units will work for a number of units orders of magnitude larger

Swarm-a download of the same file from multiple sources

Servlet – software integrating elements of client and server software

Top level Domain Name – the element of the domain name which identifies the class and not the specific network. Early examples include "edu" and "org"

Trust – a component is trusted if

1. the user is worse off should the component fail, and
2. the component allows actions that are otherwise impossible and
3. there is no way to obtain the desired improvement without accepting the risk of failure. (Note that reliable systems do not require trust of a specific component.)

UNIX – a family of operating system used by minicomputers and servers, and increasingly on desktops. Linux is a part of the UNIX family tree.

Virus – a program fragment that attaches to a complete program in order to damage the program, files on the same machine, and/or infect other programs

Abstract

Peer-to-peer systems are bringing Windows-based computers onto the Internet as full participants. Peer-to-peer systems utilize the connectivity and capacity to share resources offered by the Internet without being bound by the constraints imposed by the Domain Name System.

Peer to peer systems (P2P) are so named because each computer has the same software as the other – every computer is a peer. In contrast, an application on a server is far more powerful than the client which accesses the server. Peer computers are fundamentally equals.

In the history of the network computation has cycled, from distributed on the desktop to concentrated in one centralized location. Peer to peer systems are at the decentralized end of the continuum. P2P systems utilize the processing power or data storage capacities at the end points of the Internet.

The fundamental basis of P2P is cooperation. Therefore P2P systems require trust. Cooperation is required to share any resource, whether it be two children splitting chocolate cake or two million people sharing files. Discussions of P2P therefore require some discussion of trust and security.

P2P systems are powerful because they are able to leveraging computers that are not consistently identified by domain name or IP address, are not always connected to the network, when connected have highly variable processing power, and are not designed to have particular server capabilities other than that provided by the peering network. The systems are built to withstand high uncertainty and therefore can accept contributions from anyone with even a modem.

After considering the generic issues of P2P systems specific systems are described: Napster, SETI @home, Gnutella, Freenet, Publius, Kazaa and Free Haven. Limewire and Morpheus are implementations of Gnutella. These specific systems are used to illustrate problems of coordination and trust. Coordination includes naming and searching. Trust includes security, privacy, and accountability. (Camp, 2001)

Clients, Servers, Peers

Peer to peer systems are the result of the merger of two distinct computing traditions: the scientific and the corporate. Understanding the two paths that merged to form P2P illuminates the place of P2P in the larger world of computing. Thus peer to peer computing when placed in historical context is both novel and consistent with historical patterns. This larger framework assist in clarifying the characteristics of P2P systems, and identifying the issues which all such systems must address by design. Recall that the core innovation of P2P is that the systems enable Wintel desktop computers to function full participants on the Internet, and the fundamental design requirement is coordination.

Computers began as centralized, hulking, magnificent creations. Each computer was unique and stood alone. Computers moved into the economy (beyond military uses) primarily through the marketing and design of IBM. When a mainframe was purchased from IBM it came complete. The operating systems, the programming, and (depending on the purchase size), sometimes even a technician came with the machine. Initially mainframe computers were as rare as supercomputers are today. Machines were so expensive that the users were trained to fit the machine, rather than the software being designed for the ease of user. The machine was the center of the administrative process as well as a center of computation. The company came to the machine.

Technical innovation (the front end processor and redesigned IBM machines) made it possible to reach multiple mainframes from many locations. Front end processors allowed many terminals to easily attach to a single machine. The first step was taken in bringing access to the user in the corporate realm. Processing power could be widely

accessed through local area networks. Yet the access was through terminals with little processing power and no local storage. The processor and access remained under the administrative control of a single entity. While physical access was possible at a distance, users were still expected to learn arcane commands while working with terse and temperamental interfaces.

In parallel with the adoption of computing in the corporate world, computing and communications were spreading through the scientific and technical domains. The arpranet (the precursor to the Internet) was first implemented in order to share concentrated processing power in scientific pursuits. Thus the LAN was developing in the corporate realm while the WAN was developing in the world of science.

Before the diffusion of desktop machines, there were so-called microcomputers on the desktops in laboratories across the nation. These microcomputers were far more powerful than concurrent desktop machines. (Currently microcomputers and desktop computers have converged because of the increase in affordability of processing power.) Here again the user conformed to the machine. These users tended to embrace complexity, thus they altered, leveraged and expanded the computers.

Because microcomputers evolved in the academic, scientific, and technical realm the users were assumed to be capable managers. Administration of the machines was the responsibility of placed on the individual users. Software developed to address the problems of sharing files and resources assumed active management by end users. The early UNIX world was characterized by a machine being both a provider of services and a consumer of services, both overseen with a technically savvy owner/manager.

The Internet came from of the UNIX world. The UNIX world evolved independently of the desktop realm. Comparing the trajectories of email in the two realms is illustrative. On the desktop, email evolved in proprietary environments where the ability to send mail was limited to those in the same administrative domain. Mail could be centrally stored and was accessed by those with access rights provided by a central administrative body. In contrast, in UNIX environments, the diffusion of email was enabled by each machine having its own mail server. For example addresses might be michelle@smith.research.science.edu in one environment as opposed to john_brown@vericorp.web. (Of course early corporate mail services did not use domain names, but this fiction simplifies the example.) In the first case Michelle has a mail server on her own UNIX box, in the second John Brown has a mail client on his machine which connects to the shared mail server being run for Vericorp. Of course, now the

distinct approaches to email have converged. Today users have servers that provide their mail, and access mail from a variety of devices (as with early corporate environments). Email can be sent across administrative domains (as with early scientific environments). Yet the paths to this common endpoint were very different with respect to user autonomy and assumptions about machine abilities.

The Internet and UNIX worlds evolved with a set of services assuming all computers were contributing resources as well as using them. In contrast, the Wintel world developed services where each user had some set clients to reach networked services, with the assumption that connections were within a company. Corporate services are and were provided by specialized powerful PC's called (aptly) servers. Distinct servers offer distinct services with usually one service per machine. In terms of networking, most PCs either used simple clients, acted as servers, or connected to no other machines.

Despite the continuation of institutional barrier that prevented WANs the revolutionary impact of the desktop included fundamentally altered the administration, control, and use of computing power. Stand-alone computers offered each user significant processing ability and local storage space. Once the computer was purchased, the allocation of disk space and processing power were under the practical discretion of a single owner. Besides the predictable results, for example the creation of games for the personal computer, this required a change in administration of computers. It became necessary to coordinate software upgrades, computing policies, and security policies across an entire organization instead of implementing the policies in a single machine. The difficulty in enforcing security policies and reaping the advantages of distributed computing continues, as the failures of virus protection software and proliferation of spam illustrates.

Computing on the desktop provides processing to all users, offers flexibility in terms of upgrading processing power, reduces the cost of processing power, and enables geographically distributed processing to reduce communications requirements. Local processing made spreadsheets, 'desktop' publishing, and customized presentations feasible. The desktop computer offered sufficient power that software could increasingly made to fit the users, rather than requiring users to speak the language of the machines.

There were costs to decentralization. The nexus of control diffused from a single administered center to across the organization. The autonomy of desktop users increases the difficulty of sharing and cooperation. As processing power at the endpoints

became increasing affordable, institutions were forced to make increasing investments in managing the resulting complexity and autonomy of users.

Sharing files and processing power is intrinsically more difficult in a distributed environment. When all disk space is on a single machine files can be shared simply by altering the access restrictions. File sharing on distributed computers so often requires taking a physical copy by hand from one to another that there is a phrase for this action: sneakernet. File sharing is currently so primitive that it is common to email files as attachments between authors, even within a single administrative domain. Thus in 2002 the most commonly used file-sharing technology is unchanged from the include statements dating from the sendmail on the Unix boxes of the eighties.

The creation of the desktop is an amazing feat, but excluding those few places which have completely integrated their file systems (such as Carnegie Mellon which uses the Andrew File System) it became more difficult to share files, and nearly impossible to share processing power. As processing and disk space become increasingly affordable, cooperation and administration became increasingly difficult.

One mechanism to control the complexity of administration and coordination across distributed desktops is a client/server architecture. Clients are distributed to every desktop machine. A specific machine is designated as a server. Usually the server is has more processing power and higher connectivity than the client machines. Clients are multipurpose, according to the needs of a specific individual or set of users. Servers have either one or few purposes; for example, there are mail servers, web servers, and files servers. While these functions may be combined on a single machine, such a machine will not also run single user applications such as spreadsheet or presentation software. Servers provide specific resources or services to clients on machines. Clients are multipurpose machines that make specific requests to single-purpose servers. Servers allow for files and processing to be shared in a network of desktop machines by re-introducing some measure of concentration. Recall that peers both request and provide services. Peer machines are multipurpose machines that may also running be running multiple clients and local processes. For example, a machine running Kazaa is also likely to run a web browser, a mail client, and a MP3 player. Because P2P software include elements of a client and a server, it is sometimes called *servlet*.

Peer to peer technology expands file-sharing and power-sharing capacities. Without P2P, the vast increase in processing and storage power on the less-predictable and more widely distributed network cannot be utilized. While the turn of the century sees

P2P as a radical mechanism used by young people to share illegal copies, the fundamental technologies of P2P are badly needed within administrative and corporate domains.

The essence of P2P systems is the coordination of those with fewer, uncertain resources. Enabling any party to contribute means removing requirements for bandwidth and domain name consistency. The relaxation of these requirements for contributors increases the pool of possible contributors by order of magnitude. In previous systems sharing was enabled by the certainty provided by technical expertise of the user (in science) or administrative support and control (in the corporation). P2P software makes end-user cooperation feasible for all by simplification of the user interface.

PCs have gained power dramatically, yet most of that power remains unused. While any state of the art PC purchased in the last five years have the power to be a web server, few have the software installed. Despite the affordable migration to the desktop, there remained a critical need to provide coordinated repositories of services and information.

P2P networking offers the affordability, flexibility and efficiency of shared storage and processing offered by centralized computing in a distributed environment. In order to effectively leverage the strengths of distributed coordination P2P systems must address reliability, security, search, navigation, and load balancing

P2P systems enable the sharing of distributed disk space and processing power in a desktop environment. P2P brings desktop Wintel machines into the Internet as full participants.

Peer to peer systems are not the only trend in the network. While some advocate an increasingly stupid network, and others advocate an increasingly intelligent network, what is likely is an increasingly heterogeneous network.

Functions of P2P Systems

There are three fundamental resources on the network: processing power, storage capacity, and communications capacity. Peer to peer systems function to share processing power and storage capacity. Different systems address communications capacity in different ways, but each attempts to connect a request and a resource in the most efficient manner possible.

There are systems to allow end users share files and to share processing power. Yet none of these systems have spread as effectively as have peer to peer systems. All of

these systems solve the same problems as P2P systems: naming, coordination, and trust.

Mass Storage

As the sheer amount of digitized information increases, the need for distributed storage and search increases as well. Some P2P systems enable sharing of material on distributed machines. These systems include Kazaa, Publius, Free Haven, and Gnutella. (Limewire and Morpheus are Gnutella clients.)

The Web enables publication and sharing of disk space. The design goal of the web was to enable sharing of documents across platforms and machines within the high energy physics community. When accessing a web page a user requests material on the server. The Web enables sharing, but does not implement searching and depends on DNS for naming. As originally designed the Web was a P2P technology. The creation of the browser at the University of Illinois Urbana-Champaign opened the Web to millions by providing an easy to use graphical interface. Yet the dependence of the Web on the DNS prevents the majority of users from publishing on the web. Note the distinction between the name space, the structure, and the server as constraints.

The design of the hypertext transport protocol does not prevent publication by an average user. The server software is not particularly complex. If fact, the server software is built into Macintosh OS X. The constraints from the DNS prevent widespread publication on the Web. Despite the limits on the namespace, the Web is the most powerful mechanism for sharing content used today. The Web allows users to share files of arbitrary types using arbitrary protocols. Napster enabled the sharing of music. Morpheus enables the sharing of files without constraining the size. Yet neither of these allows the introduction of a new protocol in the manner of http.

The Web was built in response to the failures of distributed file systems. Distributed files systems include the network file system, the Andrew file system, and are related to groupware. Lotus Notes is an example of popular groupware. Each of these systems shares the same critical failure – administrative coordination is required.

Massively Parallel Computing

In addition to sharing storage P2P systems can also share processing power. Examples of systems that share processing power are Kazaa and SETI @home. Despite the difference in platform, organization, and security, the naming and organization questions are similar in clustered and peering systems.

There are mechanisms to share processing power other than P2P systems. Such systems either run only on Unix variants, depend on domain names, or are designed for use only within a single administrative domain. Meta-computing and clustering are two approaches to sharing processing power.

Clustering systems are a more modern development, also related to peering systems. Clustering software enables discrete machines to run as a single machine. Beowulf, from the University of Virginia, provides access to the processor of multiple machines as if they were a single unit. DAISy (Distributed Array of Inexpensive Systems) from Sandia was an early provider of a similar functionality. Yet these systems are descended from the UNIX branch of the network tree. Each of these systems are built to harness the power of systems running Linux, as opposed to running on systems already loaded by the Windows operating system. (Linux systems are built to be peers, as each distribution includes, for example, web server and browser software as well as email servers and clients.)

Clustering systems include naming and distribution mechanisms. Consider Beowulf, an architecture enabling a supercomputer to be built out of a cluster of Linux machines. In Beowulf, the machines are not intended to be desktop machines. Rather the purpose of the machines is to run the software distributed by the Beowulf tree is as fast a manner as possible. Beowulf is not a single servlet but rather is a combination of multiple elements. Beowulf requires many elements, including message-passing software and cluster management software, and is used for software designed for parallel systems. Beowulf enables the same result provided by a P2P processor-sharing system: the ability to construct a supercomputer for a fraction of the price. Yet Beowulf assumes the clusters are built of single-purpose machines within a single administrative domain.

Examples of P2P Systems

In this section the general principles described above are discussed with respect to each system. For each system the discussion of design goals, and organization (including centralization) are discussed. Mechanisms for of trust and accountability in each system are described.

Given the existence of a central server there are some categorizations that place SETI @home and Napster outside of the set of P2P systems. They are included here for two reasons. First for theoretical reasons, both of theses systems are P2P in that they have their own name spaces and utilize heterogeneous systems across administrative

domains in cooperative resource sharing. Second, any definition that is so constrained as to reject the two systems that essentially began the P2P revolution may be theoretically interesting but are clearly deeply flawed.

P2P systems are characterized by utilization of desktop machines characterized by a lack of domain names, intermittent connectivity, variable connection speeds, and possibly even variable connection points (for laptops, or users with back-up ISPs).

Napster

Napster began as a protocol, evolved to a web site, became a business with an advertising-driven value of millions, and is now a wholly owned subsidy of Bertelsmann entertainment. Yet the initial design goal was neither to challenge copyright law nor create a business, the original goal was to enable fans to swap music in an organized manner. Before Napster there were many web sites, ftp sites and chat areas devoted to locating and exchanging music files in the MPEG3 format, yet Napster simplified the location and sharing processes. The goal of Napster was to allow anyone to offer files to others. Thus the clients were servers, and therefore Napster became the first widely known P2P system.

Before Napster sharing music required a server. This required a domain name, and specialized file transfer software or streaming software. The Napster client also allowed users to become servers, and thus peers. The central Napster site coordinated the peers by providing a basic string matching search and the file location. As peers connected Napster to search, the peers also identified the set of songs available for download.

After Napster the client software was installed on the peer machine and contacted napseter.com, Napster the protocol then assigned a name to the machine. As the peer began to collect files it might connect from different domains and different IP addresses. Yet whether the machine was connected at home or at work Napster could recognize the machine by its Napster moniker.

Thus Napster solved the search problem by centralization, and the problem of naming by assignment of names distinct from domain names.

When a peer sought to find a file, the peer first searched the list of machines likely to have the file at the central Napster archive. Then the requesting peer selects the most desirable providing peer, based on location, reputation, or some other dimension. The connection for obtaining the file was made from the requesting peer to the providing peer, with no further interaction with the central server. After the initial connection the

peer downloads the connection from the chosen source. The chosen source by default also provides a listing of other songs selected by that source.

Accountability issues in Napster are fairly simple. Napster provides a single source for the client, therefore downloading the client is not an issue of trust. Of course, the Napster web site itself must be secure. Napster has been subject to attacks by people uploading garbage files but not by people upload malicious files.

In terms of trust, each user downloads from another peer who is part of the same fan community. Grateful Dead fans share music as do followers of the Dave Matthews Band. Each groups of fans shared music within their communities. It is reasonable to assert that Napster was a set of musical communities, as opposed to a single community of users.

Kazaa

The widely-installed Kazaa software has always been a business first. Kazaa was created by a team of Dutch programmers and then sold to Sharman Networks. In 2002 Kazaa was downloaded by more than 120 million users. Kazaa is downloaded by users so that they can access music and video files on remote machines. Kazaa has always sold advertising, charging to access the customers' attention span.

Kazaa also installs up to four types of additional software for the revenues. First, and most importantly for Kazaa, the software installs an ad server. Kazaa's business model depends on advertiser revenue. Kazaa installs media servers to enable high quality graphics in its advertising.

Second Kazaa installs software to use processing resources on the users' machine. Sharman Networks has teamed with Brilliant Networks to develop software that enables processing power to be shared. With a centralized command the Brilliant Software owners can utilize the processing power of all Kazaa users. As of the close of 2002, the system is not being used to resell processing power. Company statements suggest it is being used to allow machines to serve adds to others. (Borland, 2002).

Third Kazaa installs media servers that allows complex video advertisements.

Fourth, Kazaa alters affiliate programs. Many companies get percentages of purchases. Affiliate programs are used by businesses, not for profits, and individuals. Kazaa intercepts affiliate messages and alter the flow of revenue to Kazaa.

In some versions, Kazaa includes a shop-bot which compares prices while the user shops using a browser. The shop-bot identifies sites with better prices when the user seeks an identifiable good.

Kazaa also offers New.Net – an alternative domain name root. By enabling an alternative root New.Net allows users to choose domains names other than the original top level domain names and allows domain name registrants to maintain their own privacy. (The governing body of the original top-level domain names increasingly requires identifying information whenever a domain name is purchased. Anonymous domain names, and thus anonymous speech, are increasingly disallowed in the top-level domains controlled by ICANN.)

In terms of trust the user must trust Kazaa, and trust other users.

In order to encourage users to cooperate Kazaa has a *participation level*. According to a competitor (K-lite) participation level measures the ratio of downloads to uploads. Depending on this ratio the speed of downloads is altered. A user who offers popular content is allowed higher access speeds than users who download but do not upload.

According to Kazaa the participation level only matters if there is competition for a file. If two or more users seek to access a file then the user with the higher participation level has priority. According to Klite there are controls on download speeds for all access attempts.

Besides offering uploads, another way to increase a participation level is to increase the detail of meta data available about a file. Integrity is a measure of the quality of the descriptors of the data. Meta data describes the content, including identifying infected or bogus files by rating them as "D". "Integrity level" is another trust mechanism implemented with Kazaa. This means that the descriptors may be good regardless of the quality of the data. Other descriptions include content and technical quality.

Kazaa implements mechanisms to enables users to trust each other and trust the content downloaded. Kazaa does not implement technical mechanisms to encourage the user to trust Kazaa itself. Kazaa offers stated privacy policies for all the downloaded software. However, the difference between the descriptions of participation level at Kazaa and K-lite suggests that there is distrust. In addition, the prominent declaration on Kazaa's site that there is no spy-ware in October 2002 in Kazaa suggests that there is indeed concern. This declaration directly contradicts media reports and the description of competitors describing the installation of spyware by Kazaa. (See K-lite and Lemos, 2002)

Search for Intelligent Life in the Universe

SETI @home distributes radio signals from the deep space telescope to home users so that they might assist in the search for intelligent life. The Arecibo telescope sweeps the sky collecting 35Gbyte of data per day.

To take part in this search, each user first downloads the software for home machine use. After the download the user contacts the SETI @home central server to register as a user and obtain data for analysis. Constantly connected PCs and rarely connected machines can both participate.

There are other projects that search for intelligent life via electromagnetic signals. Other programs are limited by the available computing power. SETI @home allows users to change the nature of the search, enabling examination of data for the weakest signals.

SETI @home is indeed centralized. There are two core elements of the project – the space telescope at Arecibo. Each user is allocated data and implements analysis using the SETI software. After the analysis the user also receives credit for having contributed to the project.

SETI tackles the problem of dynamic naming by giving each machine a time to connect, and a place to connect. The current IP address of the peer participant is recorded in the coordinating database.

SETI @home is P2P because it utilizes the processing power of many desktops, and uses its own naming scheme in order to do so. The amount of data examined by SETI @home is stunning, and far exceeds the processing capacity of any system when the analysis is done on dedicated machines. SETI is running 25% faster in terms of floating point operations per second at 0.4% of the cost than the supercomputer at Sandia National Laboratories. (The cost ratio is .0004). SETI @home has been downloaded to more than 100 countries. In July 2002 there were updates to the SETI software in Bulgarian, Farsi and Hebrew.

The software performs Fourier transforms – a transformation of frequency data into time data. The reason time data are interesting is that a long constant signal is not expected to be part of the background noise created by the various forces of the universe. So finding a signal that is interesting in the time domain is indicative of intelligent life.

The client software can be downloaded only from SETI @home in order to make certain that the scientific integrity of code is maintained. If different assumptions or

granularity are used in different Fourier analyses, the results cannot be reliably compared with other result using original assumptions. Thus even apparently helpful changes to the code may not, in fact, be an improvement.

SETI @home provides trustworthy processing by sending out data to different machines. This addresses both machine failures and malicious attacks. SETI @home has already seen individuals altering data to create false positives. SETI @home send data to at least two distinct machines, randomly chosen, and compares the results. Given the number of machines this is difficult. Note that this cutes the effective processing rate in half, yielding a cost/processing ratio of 0.002 as opposed to a 0.004. However, the cost per processing operation remains three orders of magnitude lower for SETI @home than for a supercomputer.

SETI @home has also had to digitally sign results to ensure that participants do not send in results multiple times for credit within the SETI @home accounting system. (Since there is no material reward for having a high rating the existence of cheating of this type came as a surprise to the organizers.) SETI @home can provide a monotonically increasing reputation because the reputation is the reward for participation. In addition to having contributions listed from an individual or a group, SETI @home lists those who find any promising anomalies by name.

Gnutella

Gnutella was developed as an explicit response to the legal problems of Napster. The developers of Gnutella believed that the actions labeled theft by the owners of copyrights were in fact sharing. Philosophical and economic arguments (qualitative and quantitative) have been made that Napster encouraged the purchase of compact discs. Some argue that the sharing of songs on Napster was more advertisement than substitute for a purchased work. The creators of Gnutella had observed the expansion of rights of trademark holders, and observed the ability of censors to use copyright law to prevent critical speech. (The Church of Scientology has had particular success in this legal strategy.)

Based on concepts of fair use and ideological commitments to sharing, Gnutella enables sharing of various types of files. Gnutella allows users to share their disk space for storage and search by integrating the search into the client.

Gnutella searches works on the basis of local broadcasts. Each peer is connected to n other peers in a search pattern, and so down the line. If a peer receives a query that it

can answer, it responds affirmatively. If the peer does not have the requested content then the receiving peer resends the query to its immediate peers. Because Gnutella is built in this modular fashion, shutting down a single peer will not prevent sharing. Gnutella applications can exist in a large networked tree, or as independent cells.

The broadcast model of searching is considered to be a weaknesses with respect to the ability to scale. (Ritter, 2002) However, Gnutella search technique allows local cells to survive without broader connections and implements a very through search. Gnutella enables scaling through segmenting the network. Gnutella creates a small world network, where there is a network of closely connected nodes and few connections between the networks. The design is based on the six degree of separation concept (sometimes familiar as the Kevin Bacon game).

In Gnutella the searches are made anonymous, yet downloads are not. Thus there is the assumption that the server contacted by a requester will not log the request. Yet this assumption has not held up in practice. Gnutella requires that requestors trust providers.

The trust assumptions has been used to entrap criminals. In particular, some users work to defeat the use of Gnutella to trade child pornography. Using a tool to generate fake files names combining explicit words and young ages and logging file, it is fairly simple to post deceptively named files and create a “Wall of Shame” publicly showing the IP address of those who request the files. In this case the lack of anonymity enabled social accountability. Of course, the same techniques can be used to bait those interested in files about Chinese Democracy or open source software, yet in 2000 there is no record of the practice. The example of the Wall of Shame illustrates the complexity of the issue of accountability in distributed anonymous systems.

Limewire & Morpheus

Limewire and Morpheus are implementations of the Gnutella protocol. In 2002 Limewire is most popular as a Macintosh servlet while Morpheus dominates the Wintel world. Morpheus is available for the Macintosh platform. Limewire is written in Java and is available for all platforms. (As of October 2002, Limewire is available for twelve platforms.) The source of Limewire is available, theoretically preventing some of the revenue-capturing methods of Kazaa. (Of course, Limewire could make the same arrangement with new.net, as described below.)

Limewire offers a version without advertisements for $9.50 and with advertisement for free. (Note that Opera uses the same strategy.) The version with ads installs

ClickTillUWin.com - a bit of adware that pops windows up as long as the program is active.

Limewire has developed a two-tier network. There are coordinating peers (called Ultrapeers) who assist in searching and organizing downloads. These are used to optimize the system for all users. The standard peers connect to one or two Ultrapeers. The Ultrapeers do not host the files, but rather organize downloads. Each Ultrapeer is associated with a subnet, and the Ultrapeers are themselves tightly-connected.

In order to increase the speed of downloads and distribute the load on peer providing files Limewire uses swarming transfers.

Limewire implements accountability by allowing a source to obtain information about the number of files shared by a requester. If a peer requesting a file does not offer many files to others, the peer receiving the request is may automatically refuse to share any files with the requestor.

Morpheus similarly offers source code availability. Morpheus bundles its code with adware, as with Limeware. Morpheus also installs software to resell disk space and CPU cycles. Early on Morpheus redirected affiliation programs to Morpheus, however this appears to have ended on later versions.

Freenet and Free Haven

Freenet was designed before Free Haven, and thus Free Haven is both informed by and built on the assumption of Freenet. That is, Free Haven is optimal when Freenet also exists. Thus Freenet is described first. Free Haven is optimal for distribution while Freenet provides long-term persistent reliable storage. Therefore Freenet provides trustworthy content. Thus the designs of Free Haven and Freenet are focused respectively on availability and persistence. One without the other is not useful.

Freenet is meant to prevent censorship. In that design goal it is similar to Gnutella and Publius.

The goals of Freenet are anonymity for information producers, anonymity for information consumers, plausible deniability information archives, efficient storage and routing of information. A goal of Freenet is to enable all these functions in a completely decentralized manner. These goals exacerbate the problem of accountability. Recall that Gnutella does not seek plausible deniability not through encryption but rather through routing.

Freenet encrypts content and then places the encrypted content on servers based on location. Therefore more popular content will be placed closer to the requestor. Content will be distributed, deleted or stored based on demand. With the current client-server network content that is widely desired becomes less available as servers are overwhelmed. For example, Google mirrored CNN.com on September 11 and 12 when demand was very high. The moderated selection and commentary provided by Slashdot.org is so popular as to create a "Slashdot effect", which means that selected sites suffer from server download and crash. Thus the implicit argument in the design of Free Haven – that routing and storage demand the kind of flexibility provided by a fully connected P2P system – can be supported by observation.

When a Freenet host chooses to delete an object, it keeps a record of the encrypted contents and a pointer to the closet known location. The record of the encrypted document is called the document key. Freenet uses routing based on these document keys. A search can be expressed in terms of document keys. When receiving a query, if neither document nor document key is found the node forwards the request to adjacent nodes based on the history of data provision. This is similar to how IP routing is currently implemented; yet, recall the basis is the cryptographic thumbprint of each file (the document key)

In Freenet the goal of high levels of anonymity requires secure, encrypted communication. The replication of data across multiple nodes allows users to obtain information from multiple nodes if a first copy obtained is noticeably corrupt. In order to protect the provider of data, the identity of the provider is reset in the reply. Reader anonymity is protected because when a provider receives a request for information there is no source information except that it was forwarded from the immediately adjacent node. There is no way to distinguish between an original request and a forwarded request. (Notice that in IP routing the source is identified throughout the entire path.) Finally the encryption of the content during the initial storage and the storage of data signatures during the routing process makes inserting altered copies of files cryptographically infeasible. In future versions of Freenet the authors suggest that a type of payment may be required before documents are inserted. For example, publishing a document would require storing documents in advance of performing some calculations.

Freenet provides availability. Free Have provides persistence. Free Haven is intended to provide long term storage instead of easily available storage. Freenet was

built in part because of an observation on the ease of denial of service under Free Haven.

Documents can be inserted into Free Haven but not deleted. Documents are broken into some number cryptographic shares so that some subset of that number can be used to create the entire document. When storing a document each peer stores only a share. The storing peer then selects a buddy and sends a copy of the share to that buddy. Peers intermittently re-arrange shares by trading them. No peer selects a permanent buddy and shares all documents, thus when one machine goes down no documents are lost. If a buddy goes off the network, after some time, a peer selects a second buddy. When a buddy is available but a share is lost, the peer initiates the sharing protocol with another peer and records the first peer as unreliable.

In Free Haven persistence is the critical issues. Thus servers are rated based on their record in keeping shares. Recall that documents are broken into shares. When one buddy loses a share or is rarely reliable the buddy(s) whose shares have been unavailable will note this is the buddy's reputation rank. Each peer maintains its own database of the reliability of other peers, so there is no single universal ranking for a particular peer.

Mojo Nation

Mojo Nation implements a shared P2P system to enable reliable publishing at minimal cost. Mojo Nation solves the problem of availability and uses micro currency to address the problem of persistence. Mojo Nation is designed to prevent free riding. Mojo Nation implements a micro-payment system and a reputation system using a pseudo-currency called Mojo. Mojo is not convertible.

Mojo Nation software combines the following functions: search, download, search relay and publishing content. Search relay is used to support the searches of other users.

The Mojo Nation is designed to provide reliability. Mojo Nation provides reliability by using a swarm download. Swarm downloading entails downloading different elements of file available on multiple low bandwidth connections to obtain the equivalent service a single broadband connection. Swarming prevents concentration of downloads from a single server as well. Essentially swarm downloading provides decentralized load balancing.

Any participant in Mojo Nation is pseudonymous. Mojo identities are public keys (RSA) generated by the users' own computer. The pseudonym can then be moved with the machine, and is associated with its own Mojo balance.

Mojo Nation is not optimized for a particular type of file. Swarm downloading enables downloads of large files, while small files present no particular problem Examples of large files include video, while MP3 files are smaller and more easily managed.

Mojo is designed neither to enable (as with Publius) nor disable (as with Free Haven) file deletion. Files may be published individually or as an explicit collection. Publication of collections enables the creation of user lists, and collaborative filtering, as with Napster and Amazon.

Mojo peers can search for content, download content, support others' searches, offer content, or relay search information. The relay function enables users who are behind firewalls or otherwise blocked to continue to use Mojo Nation.

Searching and downloading cost Mojo while supporting others' searches and providing content earns Mojo. Each downloaded peer software package begins with some Mojo, so pseudo-spoofing assaults on the system are possible. Pseudo-spoofing is the creation of many accounts in order to built the account or reputation of some single account.

Conclusions

Peer to peer software is currently a topic of hot debate. The owners of high value commodity content believe themselves to be losing revenue to the users of peer to peer systems. In theory all downloaded music may be lost revenue. An equally strong theoretical argument is that peer to peer systems now serve as a mechanism for advertising, like radio play, so that music popular on peer to peer networks is music that will be widely purchased. There is no empirical data that can answer the question with respect to lost revenue from peer to peer systems.

There are strong lobbying efforts to prohibit peer to peer software. Some ISPs prohibit peer to peer software by technical and policy means.

There is debate within as well as about the peering community. By bundling software for ads, peer to peer systems are creating innovative business models or alternatively committing crimes against users. In one case the reselling of processing power by the software creators is seen an innovative way to support the peer to peer network. The targeting of ads and selling of user data directly echo the business plans of the Internet boom, when a joke column about giving away automobiles by covering

them with ads spawned a company. From the other perspective the peers bring the value to the community and bundled software illegitimately exploits that value. Thus some decry the bundled software that is installed with P2P code as parasitic or spyware. Installing advertising, software that records user actions, or software that redirects affiliate programs is seen by users as a violation of the implied agreement. (The implied contract is that users share their own content and in return obtain content provided by others.)

There are significant research issues with respect to digital networked information, including problems of naming, searching, organizing and trusting information. Because peer to peer systems required downloading and installing code as well as providing others with access to the users machine, the problem of trust is particularly acute. The vast majority of users of peer to peer systems are individuals who lack the expertise to examine code even when the source code can be downloaded and read.

Peer to peer systems in 2002 are at the same state as the web was in 1995. It is seen as an outlaw or marginal technology. As with the web, open source, and the Internet itself the future of peer to peer is both in the community and in the enterprise.

Peer to peer systems solve (with varying degrees of success) the problem of sharing data in a heterogeneous network. Just as no company is now without an intranet using web technologies, in a decade no large enterprise will be without technology that builds on today's peer to peer systems.

Peer to peer systems bring the naïve user and the Wintel user onto the Internet as full participants. By vastly simplifying the distribution of files, processing power, and search capacity peer to peer systems offer the ability to solve coordination problems of digital connectivity.

Press coverage of peer to peer systems today is not unlike press coverage of early wireless users at the turn of the last century, both admiring and concerned about radical masters of frightening technology bent on a revolution. In a decade or so, peer to peer systems within the enterprise will be as frightening and revolutionary as radio is today. Yet without breakthroughs in the understanding of trust in the network, peer to peer across administrative domains may founder on the problems of trust and thus become, like Usenet and gopher, footnotes for the historical scholar.

Related readings and sites of reference

J. Borland (2002) "Stealth P2P network hides inside Kazaa", CNET Tech News, April, 2002. **http://news.com.com/2100-1023-873181.html**

Camp(2001), *Trust and Risk In Internet Commerce*, MIT Press, Cambridge, MA.

Electronic Freedom Foundation, "Peer to peer Sharing and Copyright Law After Napster", (2001)
www.eff.org/Intellectual_property/Audio/Napster/20010309_p2p_exec_sum.html

Lemos (2002) "AIM+ Creators delete spyware" CNET Tech News, June 6, 2002.
http://news.com.com/2100-1040-933576.html

M Pahfl, (2001) "Giving Music Away to Make Money, *First Monday*, Vol 6 No 8.
www.firstmonday.org/issues/issue6_8/pfahl/index.html

Oram, ed. (2001) *Peer-to-Peer Harnessing the Power of Disruptive Technologies*, O'Reilly and Associates, Cambridge, MA.

B Sniffen, (2000*) Trust economies in the Free Haven Project*, MIT Technical report. Massachusetts Institute of Technology, Cambridge, MA.

J. Ritter. Why Gnutella Can't Scale. No, Really, February 2001. Available from http://www.monkey.org/dugsong/mirror/gnutella.html.

W.A. Wulf, C. Wang, D. Kienzle, (1995) "A new model of security for distributed systems", University of Virginia, Computer Science Technical Report CS-95-34, August 1995

Beowulf
www.beowulf.org/

Mojo Nation
url:sourceforge.net/projects/mojonation

The Napster Protocol
opennap.sourceforge.net/napster.txt

SETI @home
setiathome.ssl.berkeley.edu

Free Haven
www.freehaven.net

Freenet
freenet.sourceforge.net

Gnutella,
www.gnutella.org

Kazaa:
www.klite.com
Kazaa Lite
www.k-lite.tk
a version of Kazaa without 'spyware' or limits on download speed