P2P

Alex S.*

1 Introduction

The systems we will examine are known as Peer-To-Peer, or P2P systems, meaning that in the network, the primary mode of communication is between equally capable peers. Basically there is no clear-cut difference between which is a server and which is a client.

These systems are used for a variety of tasks. The popular ones we will look at are used for file distribution.

2 Napster

It is only fitting that we should start with Napster. It was the first widely used network, mostly setup to share music files in MP3 format.

2.1 Client/Server

As far as everyone was concerned, 'Napster' was the Napster client. A person would download a client program, which would traverse the user's directory looking for MP3 files, and report what it found to the server, along with the computer's address.

The server (or rather, one of great many servers operated by Napster) would index all the information each individual client has sent them. Whenever any client wants to search for something, the client sends a search string to the server, which then looks up relevant entries in the database—and returns them.

It is the client's job to then connect to that other 'peer' who is offering those files for download. In this system, the central server acts only as a tracker of data—of who is sharing which files.

2.2 Problems

Centralization is a bad thing. It is bad because if the main server dies, nobody can talk to anyone (and the network is effectively gone). This is precisely what happened when Napster

was shut down by RIAA for enabling easy free trading of music. (note that Napster itself wasn't hosting the music—it was only letting people easily search and find music that other people have).

Another problem is anonymity. In order to download music, you have to connect directly to the party offering music. That means that anyone who hosts music can easily be found, and they they track who donwloads music.

3 Gnutella

The centralization issue mentioned previously were overcome by Gnutella—unfortunately it also created a bunch of other more serious issues.

3.1 Decentralization

The major point of Gnutella is that there are no central servers. All peers are created equal. Upon starting, the client connects to a bunch of other clients. It also indexes the offered files.

Upon searching, a client sends the search query to all of its neighbors with a TTL (time-to-live) of seven to about sixteen. Every time a client gets one of these search messages, it searches its index for those files, and forwards that search query to its neighbors (with a decreased TTL). Eventually, the TTL expires, and the search completes.

Once the file is found (if it is found), the client can use plain HTTP to download the file from the hosting peer.

3.2 Problems

One of the major problems is scalability. The search algorithm is basically a flood algorithm—where it floods the network with the search query (it uses time-to-live, but it's still a flood). With many clients doing this at the same time, and with geometric growth rate of these types of searches, the network may spend most of its bandwidth just forwarding queries (ie: it's slow).

Another issue is the lack of anonymity—which after Napster died became more and more important in a file sharing application.

A less not-so-much-of-a-problem is the fact that it's 'open'. Usually, open means that it's 'good', but in this case, many people created a flood of different clients for different systems, many of which only partially operational—which made for some interesting interoperability issues.

4 KaZaA

The KaZaA network, being proprietary (and enterprising) is a bit obscure in its workings, so most of this explanation may be totally wrong.

4.1 Description

It appears that the network is a hybrid between centralization and decentralization. The network surely has central servers—these probably manage connections, and lists of users, etc.

The network also has "Super Nodes" which it seems are just regular peers that gained this 'special' status by having more bandwidth and/or CPU power than others. Instead of just being an average peer, these super-nodes can coordinate network activities (like searching, etc.) on behalf of the network.

Transferring of files happens via HTTP, directly between peers. The network doesn't 'store' files in any way—it just allows individuals to find each other to trade files.

4.2 Features

One of the interesting features is that the hybrid nature of the network makes it really hard to shut down. There isn't any single place to attack to shut it down (as it was with Napster).

Another interesting feature is that the network seems relatively reliable. You find a file, you download it, etc., no major issues.

4.3 Problems

This model lacks any anonymity. For one, KaZaA knows who you are, and anyone who downloads anything from you knows who you are[1].

Another problem is the fact that KaZaA network is proprietary. It is closed to (or at least used to be closed to) external clients. The only official available client is for Windows—and it comes loaded with spyware and adware.

5 BitTorrent

For most of the accurate information on BitTorrent, you should consult the official website[2]. The following provides a very brief description.

[1] When I say 'know who you are', I mean that they know your IP address

[2] `http://bittorrent.com/`

5.1 Preparation

The entire process starts with a `.torrent` file, that is created using a tool such as `maketorrent`. What it does is run through a file you wish to share and collect hash values for a certain number of parts of the file. You can specify several command line parameters; the important ones being the location of the *tracker*, and the number of chunks you'd like to have the file split into.

5.2 Tracker

The tracker does exactly what its name implies: it tracks. More specifically, it is a program, that runs on an Internet accessible computer (hopefully very reliable), that anyone can connect to.

A BitTorrent client connects, and announces itself as being interested in a particular `.torrent` file. The tracker maintains a list of clients who are interested in that file (along with some status information). Thus, clients can find out which other clients are interested in that file.

The tracker itself can be used for multiple files by multiple clients. It doesn't 'store' the file, nor does it constantly communicate with the clients. About every 15-30 minutes, every client will get back to the tracker to let it know of the updated status (finished downloading, etc.), but if the tracker goes down, all clients can still continue operating.

5.3 Clients

The clients have the `.torrent` file (which can be acquired via the web, e-mail, or simply copied via a floppy). They proceed to connect to the tracker which is mentioned in the `.torrent` file.

Using the tracker, they reveal themselves to the world, and find the list of other clients who are also involved in the sharing of that file. They then proceed to connect to various clients and announce what they have and what they'd like to get. They send things like "I have parts a, c, e, looking for parts b and d."

Note that if the client already has the file, then they effectively become the 'seed' for that file—in short, they're not downloading, they're hosting the file.

The system is designed such that a few initial requests are usually satisfied (clients start out being 'nice' to new-comers whom they haven't seen before). After those few 'nice' encounters, the system turns into a trading system. Where in order to get parts of files, you need to upload parts of files. If you don't upload, you won't be allowed to download. This tit-for-tat behavior ensures that people don't cheat the system (if you try to cheat, other clients will stop talking to you relatively quickly).

5.3.1 Connectivity

Notice that the system requires that clients be able to connect to other clients. This requires that clients disable firewalls on certain ports[3]. Running a client without opening up those few ports will result in a client that can only connect to other clients, but cannot be connected to—in short, it will miss a lot of opportunities to trade—and will not be contacted by new clients that are entering the system; all this will result in the client getting very slow download speeds.

5.4 Downloaded File

The download happens in relatively random chunks. Most clients allocate the whole file on disk, and then use that pre-allocated file to manage downloaded and served parts of the file. This may seem strange—since as soon as you start downloading, the client may create a few gigabyte file on disk, without having actually downloaded anything[4].

5.5 Problems

Probably the major problem to this system is that many users still manage to cheat the system. If you have high client turn-over rate, or many many clients, you can always find someone to send you 'nice' file chunks, without you actually sharing much. This is especially true for popular files.

Another problem is when the seed peer disappears. If clients have the whole file (in separate pieces), then there might not be a problem, but *usually* when the only seed leaves, it means that the file won't finish downloading. The seed *may* come back online later, but then again, it might not. Other clients, once finish downloading, may become seeds, but then again, they may just disconnect.

Another problem is when a tracker goes down. This basically means that you can't discover new clients (and new clients cannot start a download). This doesn't necessarily mean that the download won't finish—if you know the seeds and a few clients, then you can happily continue downloading from them—you only need the tracker to get the initial pool of clients, and possibly later to expand/update that peer pool.

Yet another problem is the fact that many people disconnect their client just as soon as their download ends. This has the effect of reducing the number of available seeds—not to mention it means that the file is only shared for some short amount of time (only while you're downloading). It is recommended (and considered good etiquette) to leave the client running even after download completes, and get at least two (or four—or even more) sharing ratio before disconnecting.

[3] A conservative range is: 6881-6889, but this range can be configured within the client—in fact, it is recommended that you don't use the default range.

[4] This can be alleviated by using a file-system like ReiserFS which will not actually 'use' that empty file space until it is actually written to.

The final problem, which isn't really a system problem but a usage problem, relates to the fact that most people cannot configure their computers right. The most common mistake is not to open the proper ports in the firewall/router. Another common mistake is to assume that bandwidth doesn't need to be monitored or controlled. BitTorrent will use all the bandwidth it can get—which includes interfering with itself! Slowing down BitTorrent just a bit actually speeds up downloads—and makes the network (besides BT download) relatively usable. A good tip for asymmetric connections is to set the maximum BitTorrent upload speed to 5KBps less than the maximum connection speed.

Another problem is anonymity—which is slightly lessened by the fact that you're not downloading the complete file, but a rather small (and likely meaningless-on-its-own) part of one.

6 Freenet

While the systems described so far lack any anonymity, Freenet was designed with anonymity in mind.

6.1 Decentralization

The network is setup to have no central server. Every node connects to other nodes. Unlike Gnutella, Freenet does no searches: every resource needs to have an 'id'. If a client wants to download a file, it sends (in depth first order) the 'id' to one of the neighbors, who then attempts to find it in their cache—and if not found, sends it to one of their neighbors. Basically upon receiving a request for a resource, you take on the quest to find and get that resource—so you can later send that resource to where it was requested from.

This has the effect of anonymizing requests and responses. For example, if I'm hosting some file, and someone is requesting it from me, it is almost certain that they're doing it on someone's behalf. Similarly, if I'm downloading the file from someone, it is almost certain that they got it from someone else on my behalf.

It also ensures that the network has multiple copies of popular (or often requested) resources—and that the resources are moved towards the part of the network where they're requested often.

6.2 Problems

Searching tends to take a while. This is understandable since the search travels in depth first order until it finds the resource you're looking for.

The resource that you're looking for may not be there. Resources are erased after they haven't been requested for a while. So if you're looking for something that's relatively unpopular (or old), then it might not be there.

7 Conclusion

There are many systems designed for different tasks. A few new trends are emerging that use encryption, and small 'trading' group (or friend-to-friend networks). Each network strives to optimize for something: be it flexibility, scalability, openness, robustness to attack, anonymity, speed, etc., take your pick.