The Role and uses of Peer-to-Peer in file-sharing

Computer Communication & Distributed Systems EDA 390

Jenny Bengtsson jenben@dtek.chalmers.se

Prarthanaa Khokar prarthan@dtek.chalmers.se

Gothenburg, May 2006

Computer Communication & Distributed Systems
The Role and Uses of Peer-to-Peer in file-sharing

1. Introduction

Peer-to-Peer (P2P) file-sharing is becoming a very popular network application amongst other applications such as the e-mail, instant messaging and the web. With our growing desire to instantaneously access software, music, movies etc. P2P file-sharing provides an inexpensive and quick (and many a times illegal!) means of access to such commodities. It is also noted that P2P file-sharing contributes to a major fraction of traffic in today's Internet. This report aims to define how P2P file-sharing functions and also compare already existing and newly emerging means of P2P file-sharing.

One of the most familiar architectures is the client-server architecture. In the client server architecture there is an always-on host. This host is referred to as the server. The server services the requests from numerous other hosts known as clients. The client on the other hand can be always-on or sometimes-on. The web application is a typical example of the client-server architecture. In this application the always-on web server services requests from browsers running on client hosts.

P2P architecture differs from the client-server architecture in the sense that there isn't an always-on server at the centre of the application. In a P2P architecture pairs of hosts referred to as peers communicate directly with each other. Thus, none of the participating hosts are required to be always-on a peer can go offline without this affecting the general performance of the system. In a sense each peer can be seen as both a client and a server at the same time. Each peer can initiate contact with another peer - thus taking on the role of a client and it can also perform some kind of service to other clients - thus acting as a server. In this manner peers can collaborate to perform some task without using a centralized server. The reason for this collaboration is often to share some kind of joint resource. In P2P file-sharing the resource is data, but it can also be other things such as computing power or the sharing of knowledge. Peer to peer technology is used in such fields as file-sharing, instant messaging, and distributed computing.

The following schematics illustrate the difference between client-server and P2P architectures:

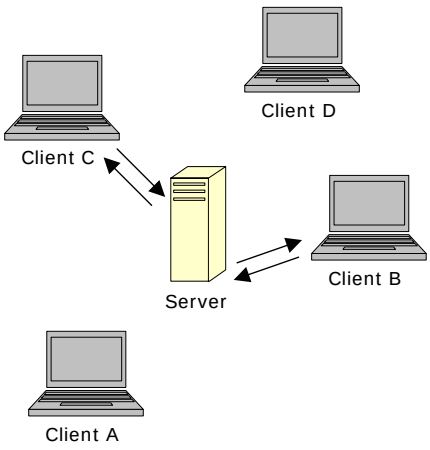

Fig. 1a Client-Server architecture

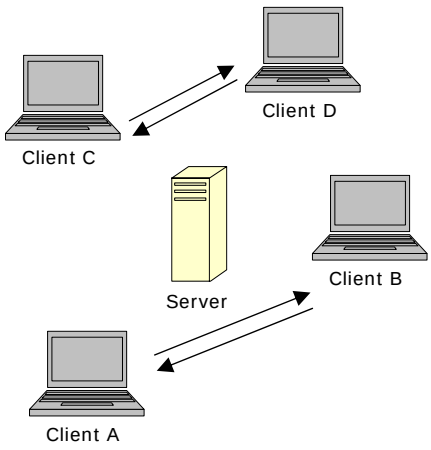

Fig. 1b Pure P2P architecture

P2P systems can be classified as either being pure P2P or a hybrid of peer-to-peer and the client-server model. In a pure P2P system all peers are equal and the system is fully decentralized, in a hybrid system there are peers communicating with each other but

some kind of centralized server is also used to perform some specific task. The file-sharing protocol of Gnutella is an excellent example of a pure P2P system, while the file-sharing protocol used in Napster is a good example of a hybrid model. In Napster a central server(s) keep track of which users have which files but the actual sharing of files is done P2P. Both Napster and Gnutella will be discussed later in the report along with some of the newer systems used for file-sharing such as Bittorrent and DC++.

Scalability vs. Decentralization

With P2P file-sharing numerous peers participate in the file-sharing community. Thus, each peer functions as a server, contributing resources to the community, as well as functioning as a client – requesting files. Therefore, each peer not only generates workload by requesting files but each peer also increases service capacity by responding to the requests of the other peers. For this reason the P2P architecture is said to be scalable - each additional peer not only increases demand but also increases service capacity. Hence, the greatest strength of the P2P architecture is scalability.

There is however a drawback to the P2P architecture. Consider the following situation: there is one peer that holds the only copy of an important file and that peer drops out of the community. This demonstrates the highly distributed and decentralized nature of the P2P application which makes the P2P architecture difficult to manage.

2. P2P File Sharing

In P2P file sharing a user – Alice – requests a file from one of the peers Bob. A TCP connection is established directly between Alice's computer and Bob's computer, and the requested file is transferred to Alice's computer. While Alice is downloading the requested file from Bob's computer, another peer Charlie may request to download a file from Alice's computer. This exemplifies the fact that each peer is a consumer as well as a contributor to the community.

However, prior to downloading a file a peer needs to determine which of the peers in the community have the desired file. There are a large number of connected peers in a P2P file-sharing system with numerous files to be shared. A peer interested in obtaining a particular file needs a way of determining the IP addresses of the connected peers that hold copies of the desired file. There are three architectures commonly employed to locate a file: centralized directory, query flooding and exploiting heterogeneity. These three architectures are discussed in the text below.

2.1. Centralized Directory Architecture

A commonly used approach to locate files is using a centralized directory. This approach was employed by the popular MP3 file-sharing application Napster. A larger central server, or a server farm, is used in this architecture to provide the directory, or "look-up", service. This architecture is best explained with the aid of a schematic.

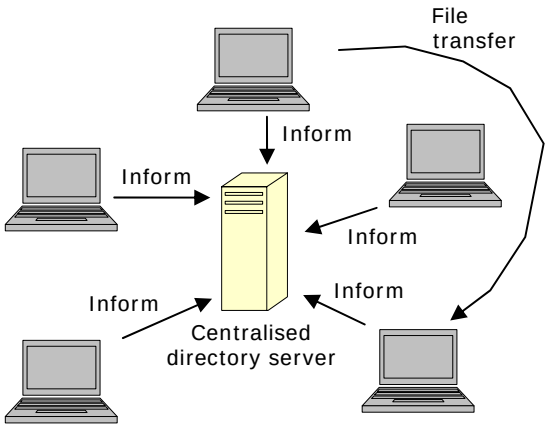

Fig. 2.1 Decentralised directory architecture

When a peer launches the P2P application, the application contacts the directory server. The application informs and updates the directory of its IP address and names of the files that it is making available for sharing. By collecting this information from all the peers becoming active the directory becomes a centralized database that maps each object name to a set of IP addresses. The directory is kept up to date by updating the database every time a peer obtains a new file or removes a file. In order to keep track of which peers are connected messages are sent periodically to the peers to see if they respond. If the directory server determines a peer to be no longer connected, that peer's IP address is removed from the database.

There are, however, several drawbacks with the centralized directory architecture. If the central directory crashes this causes the entire P2P application to collapse – this is referred to as single point failure. Yet another drawback is bottleneck performance – as there are numerous peers the centralized server needs to maintain the huge database and respond to thousands of queries every second.

Hybrids: Instant Messaging and Napster

Centralized directory P2P file-sharing is a hybrid of the P2P and client-server architectures.
As mentioned earlier a hybrid model differs from a pure P2P application. A hybrid model is a combination of both the P2P and client-server architectures. Examples of such hybrid models are instant messaging and the popular MP3 file-sharing application - Napster.

Napster is P2P in the sense that the MP3 files are exchanged directly amongst the peers. Moreover, Napster is client-server in the sense that the peers query an always-on central server to determine which currently-up peers have the desired file.

Instant messaging is hybrid in its architecture in the sense that when Alice launches her instant messaging application, such as Yahoo! Messenger, MSN etc., she registers herself at a central server. When Bob wants to chat with one of his friends on his buddy list, his messaging application contacts the central server to find out which of Bob's buddies are currently online. Once the buddies have been established Bob and Alice can chat with each other exchanging messages and files in a P2P fashion.

2.2. Query Flooding Architecture

The Gnutella client, a pure P2P architecture, employs a query flooding approach. Gnutella is a public domain file sharing application and is differs significantly from Napster. In Gnutella any peer can request for and send files, respond to and forward queries.

In Gnutella the peers form a logic network called an overlay network. The overlay network is defined by the edges, TCP connections, connecting peers. There may be thousands of participating peers in a Gnutella overlay network but a given peer is only connected to fewer than 10 nodes. Gnutella works as follows. Peers send messages over existing TCP connections to neighbouring peers in the overlay network – these messages are called Gnutella Query messages. When Alice wants to locate a file she sends out a Gnutella Query message to all of her neighbours and all of Alice's neighbours forward the query message to their neighbours – this is referred to as query flooding. When a peer receives a Query message, it checks to see if it has the file available for sharing. If the peer does have the file it sends a QueryHit message back to Alice. The QueryHit message follows the reverse path of the Query message making use of the existing TCP connections.

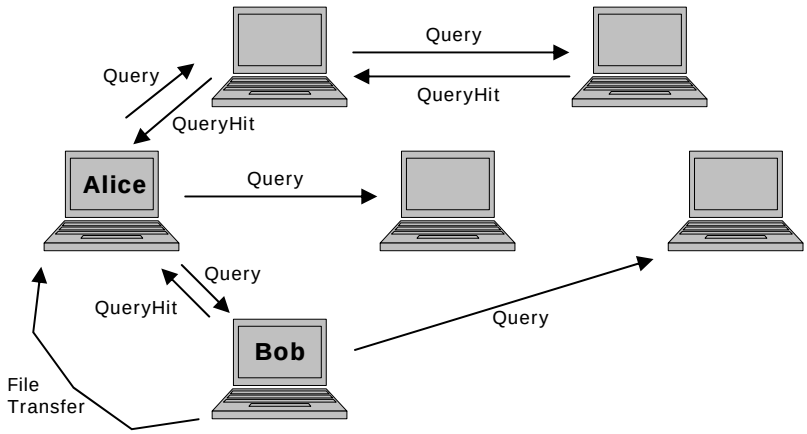

Fig. 2.2 Gnutella – File search and transfer

Alice may receive numerous QueryHit messages from many peers – she then selects a peer from the set of possible peers. Supposing Alice chooses to download the file from Bob then her Gnutella application sets up a direct TCP connection with Bob's Gnutella application. Alice's process then sends into the connection an HTTP GET message that contains the file name. Bob's process sends the file to Alice within an HTTP response message. Once Alice has received the entire file the Alice-Bob TCP connection is terminated.

There is however a major drawback with Gnutella – it is non-scalable. Due to query flooding whenever a peer initiates a query, the query propagates to every other peer in the overlay network – this dumps a significant amount of traffic onto the Internet. Limited scope query flooding provides a solution to this problem. A peer-count field is maintained in the Query message and each time the query reaches a new peer the peer count is decremented by one before forwarding the message to a new peer – this way the flooding is localized to one region of the overlay network. Limited scope query flooding reduces the query traffic that is dumped on the Internet. There is however a drawback with limited scope query flooding, there is a chance that the desired file is not located because it is present in an outside-of-scope peer.

For a peer, e.g. Alice, to join a Gnutella network, Alice must find some other peer already in the overlay network. This accomplished through Alice's Gnutella client maintaining a list of peers that are often in the network or Alice could contact a Gnutella site that maintains such a list. Once Alice has gained access to such a list she can attempt to set up a TCP connection with the peers on the list until a connection with some peer e.g. Bob is created. Alice's application sends a Ping message containing the peer-field count to Bob. Upon receiving the Ping message Bob forwards it to all of his neighbours in the overlay network. The Ping message continues to be forwarded in the network until the peer-count field is zero. As soon a peer, Charlie, receives the Ping message it sends a Gnutella Pong message, containing information such as the IP address and names of files

available for sharing, through the overlay network back to Alice. Thus, in this manner when Alice receives Pong messages from various peers she knows their IP addresses. Alice can set up TCP connections with these peers hence creating edges from itself into the overlay network.

The Gnutella architecture differs from the Napster architecture in the sense that there is no centralised directory. In comparison Gnutella is a simple, distributed P2P system that allows a peer to query for files that are located at nearby peers.

2.3. Heterogeneity Architecture

Kazaa makes use of both the decentralised directory and query flooding architecture. Kazaa is therefore a blend of Gnutella and Napster – it therefore exploits heterogeneity. Kazaa is similar to Gnutella as it does not make use of a centralized server for tracking and locating files. Unlike Gnutella, however, the peers in Kazaa are not all equal. There is a hierarchy of peers, powerful peers that have high bandwidth connections and high internet connectivity – these peers are called group leaders and have higher responsibilities. When a peer launches Kazaa the peer establishes a TCP connection with one of the group leaders. The peer informs the group leader of the files it is making available for sharing – this allows the group leader to maintain a database of the files and corresponding IP addresses of its children. In this manner, each group leader in effect becomes a mini Napster-like hub. Group leaders connect with one another over TCP connections creating an overlay network. This network created resembles a Gnutella network. In summary, the Kazaa architecture forms a hierarchical overlay network that is built upon the exploitation of the heterogeneity of the peers.

3. The new generation of P2P file-sharing

Many of the file-sharing protocols we have described this far are no longer used, or at least they are not as popular as some of the newer file-sharing applications. The original version of Napster was shut down in the year 2001 due to legal issues with copyright. This was mainly due to the fact that Napster used a centralized server listing mp3 music files that could be downloaded from its users. Since most of these songs were copyright protected the music industry wasn't too happy, and many law suites followed.

Furthermore Napster was never a general purpose file-sharing protocol it could only be used to download music files. The Gnutella file-sharing protocol is still used to some extent and many Gnutella clients are available for download, but Gnutella is not as commonly used as the other two file-sharing protocols we will describe in this section, these are BitTorrent and Direct Connect. Both of these are general purpose file sharing protocols which are easy to use. Files such as music, software, games and movies can be easily downloaded. These to protocol take on two completely different approaches.

3.1 Direct Connect

The original direct connect client was written by Jon Hess at the company NeoModus and released in November 1999. It has been less popular since the release of other clients implementing the Direct Connect protocol. Most popular is the open source client DC++. The idea behind Direct connect is very straightforward. Clients connect to a specific kind of central servers called hubs. Clients connected to a specific hub can search each other for files, exchange files and chat with each other. The hubs themselves dosen't store any files, they simply provide the software for routing chat messages and search requests/results. All file exchange is done strictly peer to peer and not through the hub. Many hubs have a share limit, i.e. a user has to share at least a given amount of data in order to connect to a hub. In this manner users are forced to share files in order to be able to download files from other peers, the share limit is different for different hubs and is decided by the hub owner. Anyone can start their own hub as long as they have

downloaded the required software, and it's easy both to download and upload files using the client software. Lists of available hubs can be found either through the client or by searching the web.

Since the hubs doesn't store any files nor do they store any file lists, the legal issues like the ones with Napster can be avoided, i.e. the owner of a hub is not responsible for whatever type of material the peers decide to share. Most hubs also have rules prohibiting copyright protected material. Even though these rules are ignored by most users they prevent the hubs from being closed down.

For users who have a high speed connection or users who wants to download relatively small files such as mp3 music Direct Connect works very well and without much effort. The problem arises when downloading large files with a slow connection, using Direct Connect a file such as a movie can take hours if not days to download. Since files generally are downloaded from only one person at a time, if that person goes offline the download is cancelled. The download can be resumed when and if the peer owning the file comes online again. This problem can be solved by simultaneously downloading the same file from many persons who has the file. There is only limited support for this in the Direct Connect protocol and therefore other software such as BitTorrent is better suited for this, BitTorrent is described in the next section

3.2 BitTorrent

One of the most popular file-sharing protocols of today (May 2006) is BitTorrent. This protocol is especially suited for downloading large files. What makes Bittorrent different from the other file-sharing protocols we have described this far is the ability to simultaneously download the same file from multiple peers. This is illustrated by figure 1. In figure 1a eight users are currently downloading the same large file from one other peer. In figure 1.b the same eight users are downloading the same file this time using BitTorrent. There are two major differences between a and b. Firstly as already mentioned, with BitTorrent the same file can simultaneously be downloaded from multiple peers. Secondly using BitTorrent the load on the peer uploading the file is reduced, why this is the case will be explained later.

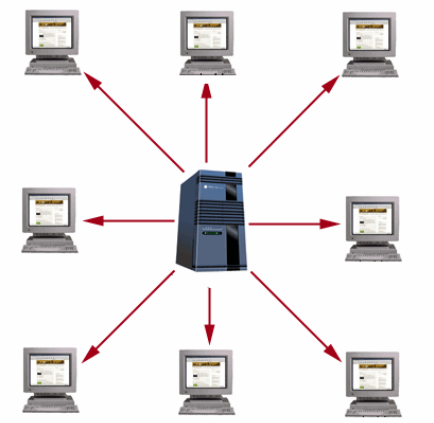

Fig 1a – Downloading of a file using Conventional File-sharing software

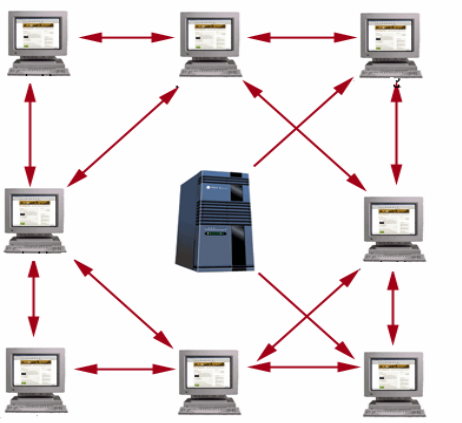

Fig 1b - Downloading of a large file using BitTorrent

BitTorrent like direct connect is not a pure P2P protocol, the file-sharing between clients is peer-to-peer but special servers called trackers are used to keep track of which users have which files.

To make a file available for other users to download a user first publishes a file with the extension .torrent on an ordinary web server. The file is uploaded using ordinary HTTP. This file is not the actual file the user want to distribute but a file containing information about the file, its length, name, and hashing information, and the URL of a tracker who knows the IP-addresses of people holding the file.

When a user wants to download a file it first downloads a .torrent file from a webpage holding it. Special client software which implements the Bittorrent protocol is then used to download the file that the torrent file points to. The client software uses information contained in the .torrent file to contact a tracker, this tracker sends back a list of contact information of users holding that file.

Each peer downloading a file reports to the tracker which file it is downloading, along with other information such as which port it listens on, this is done using a simple protocol layered on top of HTTP. The tracker responds by sending a list with contact information for other users downloading that same file.

Once a client has this information it can initiate P2P TCP connections to other peers. All peers that download a file is at the same time uploading that file to other peers (as long as someone else is interested in downloading that file), this is built into the protocol so users can not choose to only download files. This makes the spreading of newly uploaded files fast since everyone who is downloading a file also is distributing it!! To keep track of which users have what, files are divided into fixed size parts, typically 0.25 MB. A user can therefore download different parts of the same file from different users simultaneously. Special algorithms are used to determine which piece that should be fetched next in order to optimize performance, the details of these algorithms is outside the scope of this article. Let's illustrate this with an example.

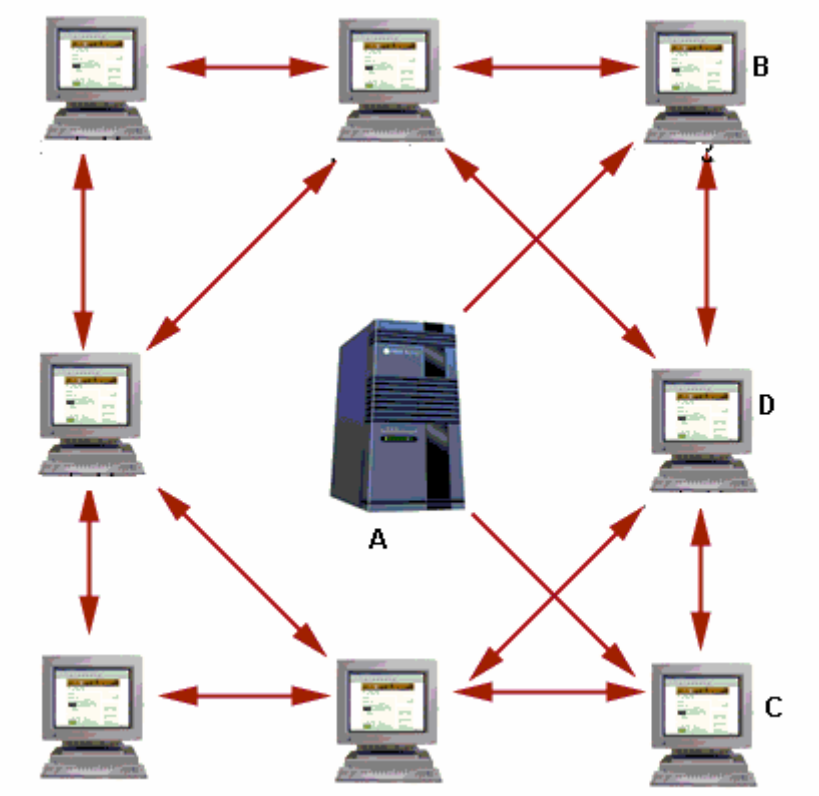

Fig 2

Consider figure 2, here a user A is distributing a file (this user is commonly known as the seeder of the file). Another user B wants to download this file so he starts downloading

the file directly from user A. User C also wants the file; he starts to download it from A as well. Most likely he downloads a different part of the file than the part user B is downloading. Now user D also is interested in the same file, but instead of downloading it directly from A he starts two peer to peer connections with users B and C. Hence user D can download from B and C even though none of them has finished downloading the entire file. In this manner a file can spread fast too many users without putting a heavy load on the originating user.

4. Conclusion

In this paper we have made a summary of the different architectures used for P2P file-sharing, we have also described some common P2P file-sharing applications and protocols, both standard text book examples and the two most popular file-sharing applications of today Bittorrent and Direct connect.

The field of P2P file-sharing is evolving very quickly and we would not be very surprised if someone reading this article in two years time would think that all the applications we have written about are really old and out of date. That was exactly what went through our mind when we read about P2P architecture in a course book about data communication and there was nothing mentioned about either Direct Connect or Bittorrent. This book was only a couple of years old from this date.

The field of P2P file-sharing is also a controversial one, in most countries sharing copyrighted material is against the law but this doesn't stop millions of people worldwide from sharing copyright protected software, movies and music. This has also had an impact on the design of new file-sharing applications. Without the strict laws on copyright, maybe Napster would still be the number one file-sharing application for sharing music?

5. References

[1] Computer Networking, Third Edition – James F. Kurose, Keith W. Ross
[2] Gnutella, http://en.wikipedia.org/wiki/Gnutella
[3] Napster, http://en.wikipedia.org/wiki/Napster
[4] BitTorrent, "Incentives Build Robustness in BitTorrent", article by Bram Cohen, http://www.bittorrent.com/bittorrentecon.pdf (webpage was valid 2006-05-01)
[5] Direct Connect, http://en.wikipedia.org/wiki/Direct_Connect_%28file_sharing%29