P2P File Sharing System

Group 7

Ishan, Kumar Aniket, Jagadeesh Killi, Aman Bansal, Ayush Mahajan

March 24, 2021

1. Introduction

Peer-to-Peer(p2p) system is a distributed application architecture that allows peers to share their workloads and partitions the task. It enables peers to share the network resources, computational power and data storage, without relying on a central authority. In the network all peers are equally privileged. Information is distributed among the member nodes instead of concentrated at a single server.

Peer-to-peer networks are most commonly used for real-time data streaming, file sharing, and computationally intensive tasks. The primary design concept of these system files is that they are distributed across nodes. This is in stark contrast to conventional client/server systems, in which all files are stored on a single central machine and all transfers are limited to the individual clients and that machine.

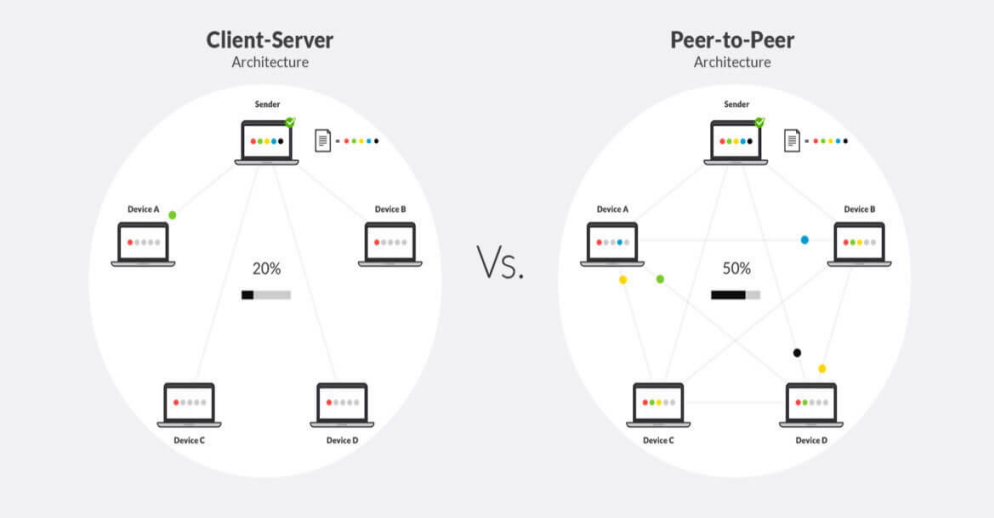

Figure 1.1 : Client-Server vs P2P Architecture [3]

Peers make a separate compartment of their resources, such as disk storage, processing power, which all can be used directly by all the other nodes in the network without any

centrally managing servers or hosts. In this system peers play the role of both clients as well as servers, which is different from the general client-server based models in which the task of suppliers and consumers are divided. The emerging collaborative P2P systems are moving ahead with the traditional p2p systems in which all the peers do similar jobs while sharing their resources, and are connecting the peers of different sectors which can provide some unique resources which is available to them only and benefits to the virtual community by making potential to do a greater task which cannot be easily done by individual peer.

While P2P systems had previously been used in many applications, the concept was popularized by file-sharing systems such as the music-sharing application Napster. File sharing is the most commonly used p2p application on the Internet. It allows peers to share their files with all the peers in the network and peers can easily search and obtain files from other peers in the network.

P2P file sharing architectures can be classified by their "degree of centralization", i.e. to what extent they rely on one or more servers to facilitate the interaction between peers. A P2P system should be highly scalable, available, distributed and more or less symmetric. Furthermore, highly dynamic p2p networks of peers with complex topology can be differentiated by the degree to which they contain some structure or are created ad hoc. By structure we refer to the way in which the content of the network is located: Is there a way of directly knowing which peers contain some specific content, or does one need to "randomly" search the entire network to locate it. They can be categorised in three categories: Unstructured, Structured and Hybrid network architecture. [1]

Another characteristic of P2P systems is that they have an interesting self-organizing capacity, in that the topology of a P2P network must change as nodes (i.e. users) enter or leave the system, in order to maintain its connectivity and performance. The main issues in a P2P system are mostly different than in traditional distributed systems, However, P2P systems provide certain advantages over conventional file systems that justify their usage in building distributed file systems.

For example, compared to the client-server model, P2P systems provide inherent scalability and availability of resources. They take advantage of the redundancy of the resources and construct a coherent view of the system using decentralized, independent components. The diverse nature of P2P systems and the large-scale distributed structure ensure the fault tolerance and resolute nature of P2P systems as compared with client-server systems. The sheer number of nodes participating in P2P architecture contributes to advantages such as being adaptable, scalable, and self-organizing. [2]

Peer-to-peer system research has exploded in popularity in recent years. Due to the phenomenal popularity of Peer-to-Peer (P2P) services like Napster, Gnutella, Freenet Bittorrent etc., using a large-scale distributed network of machines has become an important aspect of distributed computing.

2. Design Issues

We will have a look at some common design issues in a P2P file system

2.1 Fast Resource Location

The method for resource location is one of the most critical P2P design problems. As resources are distributed across multiple peers, an effective mechanism for object location becomes the deciding factor in how well such systems perform. The mechanism must be able to respond to changes in network topology.

An important location strategy used in several systems is Distributed Hash Table (DHT). It uses hashing of file or resource names to locate the object.[2]

2.2 Availability

In P2P file sharing systems, there are a variety of different node and resource availability requirements. Each node in a P2P system, in particular, should be able to receive messages from other nodes and communicate with them in order to provide access to the resources that it contributes to the network.
A denial-of-service (DoS) attack attempts to make a node and its resources unavailable by overloading it. By overloading a node, the attacker tries to make it and its resources inaccessible. The most common DoS attack aims to consume all of a node's bandwidth. If a node's available bandwidth is consumed by transferring useless messages generated directly or indirectly by a malicious node, all of the node's other resources (such as CPU and storage) become unavailable.

2.3 Load Balancing

Load balancing is a critical component of constructing reliable P2P file systems. Based on the functionality and availability of node resources, the system should be able to distribute resources optimally. The system should have mechanisms in place to prevent hotspots from forming, which are areas where the load is disproportionately high. Also, the system should be able to be rebalanced based on usage patterns.[2]

2.4 File Authenticity

In these systems, file authenticity can be a significant problem. A file authenticity mechanism responds to the following question: given a query and a set of documents that are responses to the query, which of the documents are "authentic" responses to the query?

For example[5], if a peer issues a search for \Origin of Species" and receives three responses to the query, which of these responses are \authentic"? One of the responses may

be the exact contents of the book authored by Charles Darwin. Another response may be the content of the book by Charles Darwin with several key passages altered. A third response might be a different document that advocates creationism as the theory by which species originated.

2.5 Access control

For the majority of the p2p file sharing protocols, the peers are free to share everything. This poses threat to the intellectual property distribution over the network demanding access control for such shared data. This would mean restricting the accessibility of data with only certain peers. The problem gets more complex due to assumptions taken over other peer systems architecture. Further adding restrictions to the protocol leads to a reduction of peer's utility. This introduces a tradeoff between the two decisions to let everyone share everything and maximizing each peer's utility or to put restrictions on the data being shared and reducing each peer's utility. This tradeoff stands important to keep a check by preventing illegal data from being shared and hence promoting security.

3. Basic Architectures

Peer-to-peer networks that are highly dynamic and have a complex node arrangement. This configuration creates an overlay network that may or may not be connected to the physical network that connects the nodes. The degree to which these overlay networks have some structure or are formed ad-hoc distinguishes P2P systems. We primarily categorise them in the following:

3.1 Unstructured network architecture

This also includes pure P2P and centralized P2P systems. In a pure P2P system, the software running at each node is equivalent in functionality. Example – "Gnutella Protocol" that protocol is capable of finding and locating all files over that network.

Unstructured peer-to-peer networks, on the other hand, do not enforce any fixed structure on the overlay network by design, but are instead created by nodes connecting to each other at random. In unstructured network topology, the storage of files is purely unrelated to the overlay topology. Due to this there's no information about the file that where it is stored in the network, so for obtaining the data we have to do large number of searches, essentially searching amounts to random search, in which so many of different nodes are probed and asked if they have the data files matching the query. These kind of networks are simple to construct and allow for localised optimizations to different regions of the overlay because there is no global structure imposed on them.

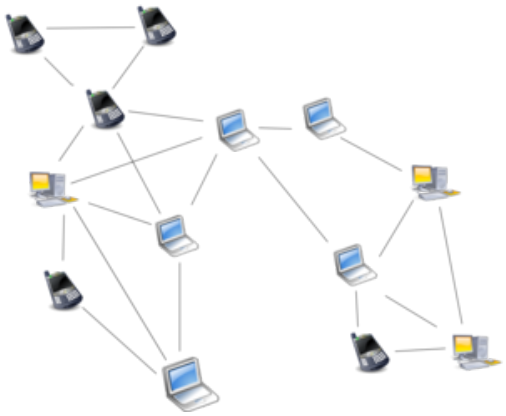

Figure 3.1 : Unstructured P2P Network[4]

Unstructured systems are different in the way in which they are connected to each other and in the way in which they are searching for the data file node-to-node which is being queried. since large numbers of peers are frequently joining and leaving the network so the Unstructured networks are highly robust towards this, i.e. they can accommodate a highly transient node population. Whereas the disadvantage of such a system is that it is hard to find the desired files without distributing queries widely. This is the reason that these systems are unscalable.

However, the lack of structure is also one of the main limitations of unstructured networks. When a peer tries to find a specific piece of data in the network, the search query must be broadcast throughout the network in order to find as many peers as possible who share the data. Flooding generates a large amount of signalling traffic in the network and consumes more CPU/memory (due to the increased demand). Furthermore, since there is no link between a peer and the content it manages, floods cannot be guaranteed to find a peer with the desired data. Popular content is likely to be available from most peers, and any peer looking for it is likely to come up with the same results.But if a peer is looking for rare data shared by only a few other peers, then it is highly unlikely that search will be successful

3.2 Structured network architecture

Structured networks arose primarily as a response to the scalability problems that plagued unstructured systems. The overlay is organised into a particular topology in structured peer-to-peer networks, and the protocol ensures that any node can effectively search the network for a file/resource, even if the resource is extremely rare.

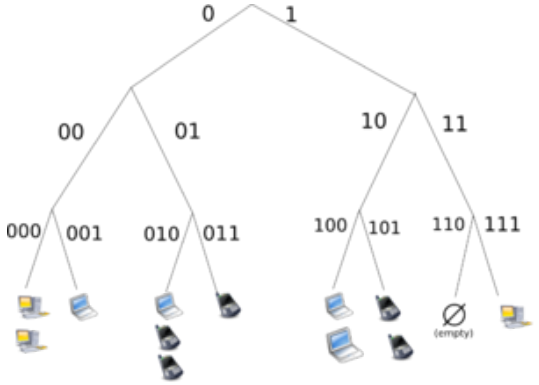

Figure 3.2 : Structured P2P network, using a distributed hash table (DHT) [4]

Structured systems were proposed, in which the overlay network topology is tightly controlled and files (or pointers to them) are placed at precisely defined locations, because the random search techniques used by unstructured systems appear to be intrinsically unscalable. These systems use a distributed routing table to map the file identifier to its location, allowing queries to be efficiently routed to the node that has the desired file. Exact-match queries, in which the full identifier of the requested data object is known, have a scalable solution in structured systems.

Nodes in a structured overlay, on the other hand, must keep lists of neighbours who meet particular requirements in order to route traffic effectively through the network. This makes them less resilient in networks with a high rate of churn, or when a large number of nodes join and leave the network regularly.

3.3 Hybrid network architecture

It's one that uses an index server for search and has an index server at the centre that contains information on the locations of resources. Because it includes one centralised peer that performs all operations as a server, such as keeping all information on the peers and responding to requests for that information, a hybrid peer-to-peer network plays the role of a client-server network. The centralised peer knew which resources were free and which were shareable. They accept full responsibility for bringing all available resources to host management.

Peer-to-peer and client-server models are combined in hybrid models. A common hybrid model involves a central server that assists peers in locating one another. Spotify was an example of a hybrid model [until 2014]. There are a number of hybrid models available, all of which make trade-offs between the centralised functionality provided by a structured server/client network and the node equality provided by an unstructured peer-to-peer network.

4. Study of some present systems

4.1 Freenet

Freenet is a distributed storage system where identical nodes participate in storing and fetching of data without a centralized server. Since no centralized server is employed, index data of the individual files are stored across the nodes and routing protocols are employed to find a node containing a file with a given key. Freenet ensures that a node which requests a file cannot find the uploader of the file implicitly. It employs various techniques and routing protocols to store and retrieve data efficiently. Data stored on freenode does not have a fixed expiry date and may not live indefinitely. As per a node data storage capacities, older files may be deleted to make room for newly inserted files. The deletion is usually based on LRU principle i.e., the file will live as long as there are nodes which make requests for it. It provides a great resistance towards deniability of data in the system.

4.1.1 Architecture [7][8]

Keys and Searching of Files - Files on freenet are identified with a key which is 160 bit sha-1 hash where this hash can be created in three main ways.

- Keyword-signed-key (KSK) : At the time of insertion of a new file, the publisher of the file needs to assign a short description to the file. Say, if we wanted to upload a textbook on distributed systems, we give a description such as "computer-science/distributed-system-textbook.pdf". A public-private key pair is deterministically generated from this text and public key is hashed to get the key to the file. So, any node which needs to retrieve the file by knowing just the descriptive-text of the file. But this method will incur a collision of keys as soon as another file with the same description is uploaded to the network. But it provides a minimal identification of file using the descriptive text.

- Signed-subspace-key (SSK) : In case of KSK, two different publishers of a file can choose the same descriptive text. To avoid this, instead of a flat global namespace, each publisher can generate keys from individual namespaces using asymmetric cryptography. Here, the publisher will generate a random public private key-pair before uploading any files. To upload a file, the KSK of the file is XORed with the public key of the publisher and signed with the private key. So, the publisher needs to post both the descriptive text along with his/her public key for others to retrieve the file. So, any requester of the file can verify the authenticity of the file using the public key associated with the subspace key.

- Content-hash-key (CHK) : Here, the hash is generated by hashing the contents of the file. This is useful in the case of storing large files using splitting and updating of the file. A single subspace key can list down content hash keys of all the parts of

the file. The requester can download all the individual parts to get the complete file. During updating the file, the same subspace key can list down the content hash key of the modified file.

File Retrieval - A node can request a file in the system using the key associated with the file. If the node itself can not fulfill the request, it will forward the request to the next node by referring to the local routing table. Each request contains a unique id and hops-to-live parameter so that the request may not circulate in the network indefinitely. The request will keep on circulating until the hops-to-live parameter expires or a node is found which can send the file data.

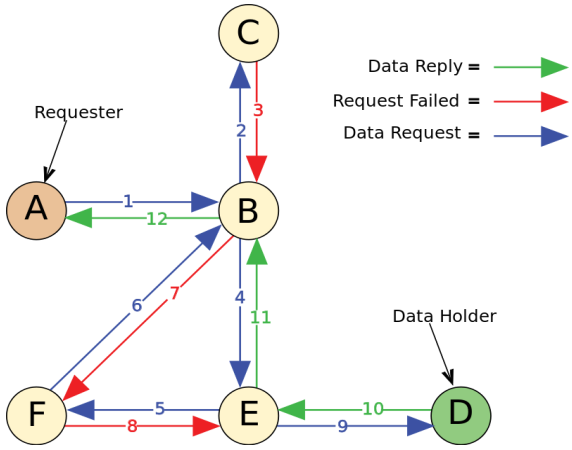

Figure 4.1 : A typical request sequence [8]

In the above figure 4.1 requests are forwarded from one node to another in a backtracking manner. If a node fails to satisfy the request, it will send an error message to it's preceding node. Here, node C replies to node B indicating failed requests. Thus, node B can forward the request to the next available node E. The request from F to B will cause a loop and B will report F indicating a loop, thus F can report back to E and E will choose the next available node which is D. Since D contains the file data, it will send the data to E, from there to B and finally arriving at A which is the originator of the request.

It is not guaranteed that two similar files reside at the same node since they may contain different hashes. This mechanism also guards the system such that if a node containing a file of a topic (such as computer science) fails, all the other files on the same topic will not completely disappear from the system.

Storing Files - To store a file in the system, the publisher needs to generate a key for the file using any of the three different methods and needs to set a hops-to-live parameters. This hops-to-live parameter will determine the replication of the file. During insertion, the insert request is propagated to all the nodes which are in the range of hops-to-live dfs length of the origin node. If at any point, a key collision occurs, the error will be

propagated back to the origin node, and the publisher needs to use a different key for the file.
Once there is no key collision, the data will be propagated to all the nodes in the hops-to-live range. The next node at each propagating node will be determined locally using a routing table. The routing table tells the node to send the file to a node which contains files with similar keys. This results in clustering of files with similar keys at the same node.

The system needs to deal with the finite amount of storage available and store the newly added files as well. Nodes have limited amounts of storage and cannot accommodate data exceeding its capacity. So, Freenet provides a way to forget old files using the LRU method. So, a file exists in the system as long as nodes are requesting it. So, old files are deleted gradually to accommodate space for new files. In the same way, the routing table is modified to reflect deletion and addition of files in the system.

Joining of new nodes - When a new node needs to join the system, it can do so using ip addresses of existing nodes. Although, once a node joins the system, it can learn more about the system by uploading and downloading files. But, the new node somehow needs to announce its presence to other nodes. The complicated issue is that which key should be assigned to the new node such that files with similar keys are stored at that node and similarly the node key can be inserted in the routing tables of other nodes. If not done properly, it could result in inefficient storage and deniability of data and may be susceptible to security attacks. To prevent that, a node chooses a random seed which is xored with seeds of other nodes which generate the key. This generated key is not influenced by one malicious party and all other nodes can identify this node using this key under their routing table.

Routing - The routing table at each node is constructed based on the lexicographical similarity of file keys. An entry in the routing table may look like “ABGHTK : tcp://100.12.10.10:3345” . So if the node fails to satisfy a request with file key “ABGHT”, it will forward it to the next node with ip 100.12.10.10 as per the routing table. This routing table is updated as the files are transferred between nodes. Freenode tries to store files with similar prefixes at the same node. Since the keys are hashes, there is no content similarity between files at one node. Even at the phase of uploading a new file, the file is duplicated at nodes with similar key prefix.

4.1.2 Protocol Features

Freenet provides a truly peer to peer file sharing system using identical nodes with no central servers thus avoiding single point of failure. It allows uploading different versions of the same file and provides authenticity of the publisher of the file. The routing protocol employed efficiently finds the node which contains the file without the use of a centralized server.

4.1.3 Challenges

In freenet, Even though it is not possible to determine the data source of a file, it is not completely fool-proof. If the nodes in between the data source and requester of the file are compromised, it is possible to determine the identity of the data source as well. At the cost of anonymity, Freenet consumes too much bandwidth to serve a file from a data source and requester. A direct connection can greatly speed up this process but undermining anonymity. Even though the routing protocols are efficient in finding the data source of a request, it is not as fast as using a central server.

4.2 Bit Torrent

BitTorrent [9] is one of the most popular peer-to-peer file-sharing protocols known for sharing large files over the Internet like tv shows, movies, games, software, etc. BitTorrent gained a lot of popularity amongst several service providers as an efficient and scalable way of delivering data while reducing the load on central servers along with the download time for the peers.

According to "Application Usage & Threat Report" [10] in 2013, Bittorent accounted for 3.5% of the whole world wide bandwidth which was more than half of the bandwidth allocated for file sharing. This meteoric rise in the popularity of BitTorrent was due to several innovative mechanisms introduced like tit-for-tat (TFT) and rarest first (RF). These mechanisms helped in solving the problems faced by the traditional protocols. BitTorrent has attracted many pieces of research over the years ranging from a survey of its performance [11] to modifications in the core algorithm to further increase its performance [12].

4.2.1 Architecture

BitTorrent works on an overlay network called "Torrent" where connections are established between the peers for the corresponding file being distributed. Along with the peers, tracker and web server are also required for file distribution in BitTorrent.
The tracker is a special node used by peers to find each other downloading the same file by keeping a log of peers that are currently downloading a file. The tracker is not directly involved in the transfer of data as it does not have a copy of the file. The information between tracker and peers is exchanged using a simple protocol on top of HTTP. The peer gives information to the tracker about which file it's downloading, ports it's listening on, etc. The tracker in return sends a response consisting of a list of other peers who are downloading the same file.

A peer is classified into two types: leecher and seed. Leecher is the one downloading the file. A seed is the one having a full file staying online to serve others. To publish a file using BitTorrent for download, the first step is to create a meta info file called a "torrent".

The torrent file contains the information regarding the file like filename, size, hashing information of the pieces, and URL of the tracker. This torrent file is required by the peer who wants to download the file corresponding to that torrent file.

To download a file corresponding to a .torrent file, the peer contacts the tracker to request a list of peers downloading the same file. Tracker on receiving the request selects a number of potential neighbor peers randomly (usually 50) from the list of all active peers in the torrent. On receiving the list, the peer contacts them to create a set of neighbors. If the number of neighbors falls below 20, the peer contacts the tracker again for the list.

When creating the torrent file from the original file, the original file is cut into smaller pieces, usually 512 KB or 256Kb in size. Earlier SHA-1 hash codes of the pieces were included in the torrent file. Newer protocol shifted to SHA-256 as the earlier one was declared not secure. The downloaded data is verified by computing the hash for the downloaded data with the one present in the torrent file. This ensures that the data is downloaded free of errors and guarantees that the real files were downloaded. As the piece is downloaded and verified, the peer downloading the file reports this to other peers in the swarm about the new piece.

The process followed in file-sharing can be seen as 5 steps as shown in figure 4.2:

- For Peer A to download the file, Peer A first downloads the corresponding .torrent file from a web server.
- Peer A then contacts the tracker for a list of active peers participating in the torrent.
- The tracker returns a list of peers involved in the torrent.
- Peer A adds all the connected peers from the list as its neighbors and sends the file piece request to each other.
- Once the request is accepted, Peer A can exchange file pieces with the neighbors.

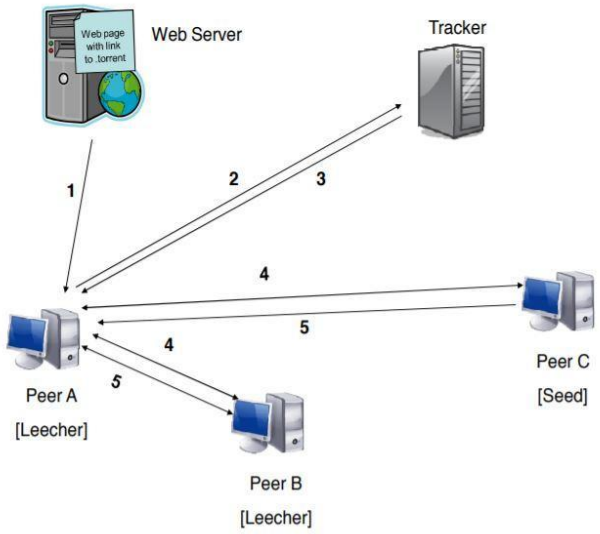

Figure 4.2 : BitTorrent file sharing process [11]

The key architecture points of BitTorrent are peer selection strategy and piece selection strategy. The peer selection strategy focuses on maximizing the service capacity amongst the peers and the piece selection strategy tries to keep the pieces available which the peers and its neighbors might be interested in.

To understand the importance of peer selection strategy we can look at an example. Let a peer A be downloading a file. A will be interested in another peer, say B, only if B possesses pieces which A doesn't possess. The more the peers A is interested in, the better will be the downloading ability for A. So it becomes crucial for A to have neighbors with maximum peers like B. But due to the tit-for-tat mechanism (TFT) adopted by BitTorrent, B would like to serve A only if B is also interested in A. This highlights the importance of a good peer selection algorithm.

Hence the peer selection is outlined by four major mechanisms: : *tit-for-tat (TFT), optimistic unchoking (OU), anti-snubbing, and upload only.* The combination of these mechanisms try to reward the peers which contribute by uploading and punish the freeloaders which just download without uploading the pieces back. This enhances the download experience for the contributing peers. The mechanisms are as follows:

- Tit-for-tat: As the name suggests this strategy rewards the peers who can provide the best value back. As the peers are connected to only a fixed number of neighbor peers (typically 4), it makes sense to choose the ones with best downloading rates and choke others . After every 10 seconds, the downloading peer reevaluates by reassessing the downloading rates of the neighboring peers. If any choked peer promises better downloading rate than any of the unchoked ones, then the roles are swapped. This strategy encourages more contribution and punishes free-riders.

- Optimistic Unchoking: With just the TFT strategy there is no opportunity for discovering other peers which might be better than the ones being used. Hence it becomes important to discover better neighboring peers. This is done every 30 seconds by unchoking a randomly selected neighboring peer regardless of its uploading rate. If the new peer connection is better than one of the existing unchoked ones then this replaces one of them. While this helps in discovering peers with higher upload rate, it is also useful for newly joined peers to get started.
- Anti-snubbing: A peer might be getting poor download rates as it may be choked by the peers it was formerly downloading from. So if a peer doesn't get any piece from a neighboring peer in some time, the leecher assumes it is 'snubbed' by that peer. As optimistic unchoking is done every 30 seconds, the peer now does not upload to this neighboring peer any further through regular unchoke to recover download speed faster.
- Upload Only: As a leecher finished downloading the entire file, it becomes a seed. As for the TFT strategy the seed has nothing to gain by selecting peers based on downloading
  rates, it prefers to upload to peers with better uploading rates. This helps in replicating the pieces faster and then uploading them to other peers faster.

After the peer selection, choosing pieces to download unintelligently can also lead to situations where all peers have pieces currently available and none of the missing ones. This highlights the importance of a good piece selection algorithm. The goal is to replicate different pieces on different peers faster. This ensures that all the pieces are at least present somewhere in the network. The pieces are further broken down into smaller sub-pieces which can be downloaded from different peers. There are four major mechanisms as follows:

- Strict Priority: Here peers try to download a whole piece before requesting another piece. This means that if sub-pieces for a piece are requested then the remaining sub-pieces of the same piece will be requested before the sub-pieces of any other piece. This is done to download the complete piece as soon as possible as only the complete pieces can be traded with others.
- Rarest First: This means that the peers prefer to download pieces which are the rarest. Each peer keeps a list of the pieces that each of its neighbors possess. This list is updated as a new piece becomes available from its neighbors. The peer then downloads the pieces rarest among its neighbors. This helps in balancing the load by spreading the pieces which are rare. As the current peer gets the rare piece which others might want, it will make other peers interested in trading with this piece further increasing the download speed. Along with removing the fear of missing pieces in the network as the seed leaves creating a bottleneck, it also helps in increasing the download speed across peers as the rare pieces are replicated over time providing multiple copies.
- Random First Piece: For a peer joining a torrent, it has nothing to upload so it becomes important to get the first piece as fast as possible so as to reciprocate for TFT strategy. Here the rarest first strategy will fail as in the rarest first strategy the

rare pieces are present with a lesser number of peers, it slows down the download rate. So the peer here will download the first piece randomly to get a whole piece as fast as possible.

- Endgame Mode: This mode is activated as the peer is nearing the complete download of the file. If the final request piece is requested from a peer with a slow transfer rate, the finishing of download will be delayed. To solve this the peer requests all of its neighbors for remaining sub-pieces by broadcasting the request. This helps in completing the download of that last piece faster. As the piece is downloaded a cancels message is sent to indicate that the download is completed.

4.2.2 Protocol Features

BitTorrent while being complex, configures the connections to the peers without the users being involved making it self-configuring. Dynamically decisions are made inside the client of every peer, based on the list of possible peers and the algorithms presented making it easy to use for the users.

The major issue before BitTorrent was that most users tend to have different speeds for download and upload. Many traditional peer-to-peer file sharing protocols suffered by not handling this. This used to cause a user to have a very lower download speed for a file due to uploaders' downlink and uplink. This implied that one-to-one file-sharing was not an optimal solution. BitTorrent tackled this by splitting files into smaller chunks and downloading chunks from different users at the same time which resulted in better utilization of downlink bandwidth. This significantly reduced the downloading time making the downloads faster.

In traditional models, the increase in the number of downloaders increases the load on servers hosting the file. The architecture of BitTorrent turned this con around by utilizing the upload capacity of the peers that are downloading a file and making the protocol work better as the number of downloaders increased. This enables a much faster download speed than before. Along with the better download speeds, it relieves the pressure off the servers having large files.

Further the "tit-for-tat" strategy tries to prohibit "free riders" from destroying the dynamics of the peer-to-peer network. If peers contribute more it keeps their download rates higher while if a peer blocks other peers from uploading, it will lead to itself being choked by them and thus affecting its download rate. BitTorrent encourages uploading in return for a better chance for faster download.

The cryptographic hash stored in the torrent file provides a way to promote security and authenticity. By identifying any modification or changes done to the file with the hash provided, any malicious attempts against the peers can be stopped.

4.2.3 Challenges

While BitTorrent proposed a unique way to improve performance with more downloaders, it created trouble for the old or unpopular files. As the unpopular files will have a lower number of users downloading them, it will be difficult to find them and fewer users to download from. This results in making the protocol effective when dealing with highly demanded files and less popular files tend not to be available.
The random neighbor selection strategy also could be modified to better use the network resources and further improve performance.

4.3 Gnutella

Gnutella, established in 2000, was the first decentralized Peer-to-Peer (P2P) file sharing network and is still active today. Using a Gnutella client, users can search, download, and upload files across the internet.

Gnutella[13] is an open source distributed file sharing protocol that defines the way distributed nodes communicate over a peer-to-peer (P2P) network.It is a decentralized network in which users can see the files of a small number of other Gnutella network members, and they in turn can see the files of other network members, in a kind of daisy-chain effect. After installing and launching Gnutella, the user's computer becomes both a client and a server in the network, which is called GnutellaNet. Gnutella allows network members to share any file type.

4.3.1 Architecture[15]

Gnutella's distributed search protocol allows a set of peers, servants or clients,to perform filename searches over other clients without the need of an intermediate Index Server.The Gnutella network topology is a pure Ad-Hoc topology where Clients may join or leave the network at any time without affecting the rest of the topology in any sense. The Protocol hence, is designed in a highly fault-tolerant fashion with a quite big overhead of synchronization messages travelling through its network. All searches are performed over the Gnutella network while all file downloads are done offline. In this way every servant that needs to serve a file launches a mini HTTP 1.1 web server and communicates with the interested servants with HTTP commands.The core of the protocol consists of a set of descriptors which are used for communication between servants and also sets rules for the interservantexchange.

These descriptors are:

- Ping : Ping messages are sent by a servant that needs to discover hosts that are currently active on the network.
- Pong : Pong messages on the other hand are responses to Ping requests every time a host wants to come in communication with the peer that requests to join the network.

- Query : The primary mechanism for searching the distributed network. A servant receiving a Query descriptor will respond with a Query Hit if a match is found against its local data set.
- Query Hit : A positive reply to a search, indicating that the sender has got the file.
- Push : Requests a file be sent to the requester.

Although the protocol doesn't set any bound on the amount of incoming or outcoming connections from a particular host, most Gnutella clients come with a default value for these parameters, usually 10, but which are adjustable by the user. A client joining the Gnutella network for the first time may of course not have any clue regarding the current topology or its neighbouring peers making the connection to the network impossible. For this reason most client vendors have set up Host-Caches Servers which serve clients withIP addresses of servants currently connected to the network. Another Mechanism which is widely deployed, but is not part of the protocol, is the use of local caches of IP addresses from previous connections.

In this way aservent doesn't need to connect to an IP acquisition server (i.e. a host-cache)but can rather try to establish a connection with one of its past peers.Right after a peer has obtained a valid IP address and socket port of another servant it may perform several queries by sending Query descriptors and receive asynchronously results in Query Hit descriptors. Downloading a file is done offline with the HTTP protocol. If the servant (say node p2) is firewalled, then the requesting node (say p1) may request from p2, with a Push descriptor to"push" the file.

Files are searched and located based on the lookup protocol . In Gnutella, this is done by flooding the network. Since a peer knows at least one other peer that runs the software (i.e., Gnutella), the initial search query message (QUERY) is sent to all directly known peers. A peer is identified by having its IP address. Therefore, Gnutella is considered non-anonymous P2P networks. Such networks are characterized by the possibility to associate a file with an IP address (i.e., the identifier of a user). When a peer receives a QUERY message, it checks its local hard disk to evaluate if the file is available. If the file exists, a QUERY HIT message is returned. The response message contains the IP address as well as the file name located. In addition to checking for the file on the user's own machine, the QUERY is forwarded to all peers known to this machine. By following this procedure, the original QUERY is forwarded and multiplied by a huge factor. For the sake of the network load, each QUERY contains a Time-to-live (TTL). The TTL limits the time a QUERY is forwarded on the network. Without it, messages would be routed forever, heavily decreasing the network's performance.

At the end of the above described process, the searching peer receives a list of file names and the corresponding peer (IP address) possessing it. The peer then directly connects to one other peer holding the desired file by sending a GET message. Upon receiving a GET message, the peer starts sending the file to the requesting peer by sending PUSH messages. This process repeats until the file is completely exchanged. Based on this architecture,

Gnutella is only designed to be a public file-sharing system. It is not suited to enable private file sharing due to its flooding lookup. However, since it is an open protocol, everyone can use it to build their own private sharing network.

4.3.2 Protocol Features[14]

Connect - A Gnutella servant connects itself to the network by establishing a connection with another servant currently on the network. The acquisition of another servant's address can be done by caching the data from the last connection. Once the ad-dress of another servant on the network is obtained, a TCP/IP connection to the servant is created, and the following Gnutella connection request string(ASCII encoded) may be sent:

> GNUTELLA CONNECT/protocol version string, where protocol version string is defined to be the ASCII string ”0.4” (or,equivalently, ”302e34”) in this version of the specification.A servant wishing to accept the connection request must respond with : GNUTELLA OK . Any other response indicates the servant's unwillingness to accept the connection. A servant may reject an incoming connection request for a variety of reasons - a servant's pool of incoming connection slots may be exhausted,or it may not support the same version of the protocol as the requesting servant.

Search - Gnutella protocol uses the standard flooding mechanism. The Steps of search are as follows:

- Node initiates search for the file.
- Send a message to all neighbors.
- Neighbors forward message.
- Nodes that have the file initiate a reply message.
- Query reply message is back-propagated.
- File download occurs. In case one client is behind a firewall, another servant can request that the servant push the file to itself.

Download - Files are downloaded out-of-network i.e. a direct connection between the source and target servant is established in order to perform the data transfer. File data is never transferred over the Gnutella network. The file download protocol is HTTP. The servant initiating the download sends a request string of the following form to the target server :

> GET /get/File Index/File Name/ HTTP/1.0 Connection: Keep-AliveRange: bytes=0 User-Agent: Gnutella

where File Index and File Name are one of the File Index,File Name Pairs from a Query Hit descriptor’s Result Set. The file data then follows and should be read up to, and including, the number of bytes specified in theContent-length provided in the server’s HTTP response. The Gnutella protocol provides support for the HTTP Range parameter, so that interrupted downloads may be resumed at the point at which they terminated.

4.3.3 Challenges

The first problem lies in bandwidth: Gnutella creates a huge amount of network traffic, roughly 50% of which are just pings. Some of this could be mitigated with aggressive caching and consolidation of ping messages, but early versions of the protocol were very bandwidth-intensive[13].

The second problem is that 70% of Gnutella users are freeloaders who only ever downloaded files and never uploaded any of their own. This makes for an unhealthy balance between downloaders and uploaders[13].

In Gnutella, more than half of the network receives every single query. This is overkill for file sharing, where we don't need that large a portion of the network to receive each message—most searches are for common files, which many nearby nodes can serve you.

These problems are compounded by Gnutella's flat architecture, which treats every node as equal in the network topology. But when it comes to bandwidth, nodes are not all equal—some nodes stay online more consistently[14].

References

1. Stephanos Androutsellis-Theotokis and Diomidis Spinellis. 2004. A survey of peer-to-peer content distribution technologies. ACM Comput. Surv. 36, 4 (December 2004), 335–371. DOI:https://doi.org/10.1145/1041680.1041681

2. R. Hasan, Z. Anwar, W. Yurcik, L. Brumbaugh and R. Campbell, "A survey of peer-to-peer storage techniques for distributed file systems," International Conference on Information Technology: Coding and Computing (ITCC'05) - Volume II, Las Vegas, NV, USA, 2005, pp. 205-213 Vol. 2, doi: 10.1109/ITCC.2005.42.

3. Konstantin (2018, June 28). What's the difference between peer to peer and client server?. Last accessed 22nd March 2021: https://www.resilio.com/blog/whats-the-difference-between-peer-to-peer-and-client-server

4. Peer-to-peer - WikiPedia . Last accessed on 22nd March'21 : https://en.wikipedia.org/wiki/Peer-to-peer

5. Daswani N., Garcia-Molina H., Yang B. (2003) Open Problems in Data-Sharing Peer-to-Peer Systems. In: Calvanese D., Lenzerini M., Motwani R. (eds) Database Theory — ICDT 2003. ICDT 2003. Lecture Notes in Computer Science, vol 2572. Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-36285-1_1

6. D. Hughes, J. Walkerdine and K. Lee, "Monitoring Challenges and Approaches for P2P File-Sharing Systems," International Conference on Internet Surveillance and Protection (ICISP'06), Cote d'Azur, France, 2006, pp. 18-18, doi: 10.1109/ICISP.2006.22.

7. Clarke I., Sandberg O., Wiley B., Hong T.W. (2001) Freenet: A Distributed Anonymous Information Storage and Retrieval System. In: Federrath H. (eds) Designing Privacy Enhancing Technologies. Lecture Notes in Computer Science, vol 2009. Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-44702-4_4

8. Freenet - WikiPedia . Last accessed on 20th March'21 : https://en.wikipedia.org/wiki/Freenet

9. BitTorrent. Last accessed on 20th March'21 : https://www.bittorrent.com/

10. Palo Alto Networks Application Usage & Threat Report. Last accessed on 20th March'21 : https://blog.paloaltonetworks.com/app-usage-risk-report-visualization/

11. R. L. Xia and J. K. Muppala, "A Survey of BitTorrent Performance," in IEEE Communications Surveys & Tutorials, vol. 12, no. 2, pp. 140-158, Second Quarter 2010, doi: 10.1109/SURV.2010.021110.00036.

12. Fair File Swarming with FOX (2006): http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.60.8125

13. Haseeb Qureshi (2019, Dec 29). Gnutella: an Intro to Gossip. Last accessed 22nd March 2021: https://nakamoto.com/gnutella/

14. How Gnutella Works. Last accessed 22nd March 2021: https://computer.howstuffworks.com/file-sharing.htm

15. Ammann, M. and Jona. "Design and Implementation of a Peer-to-Peer based System to enable the Share Functionality in a Platform-independent Cloud Storage Overlay." (2013)