Survey of Peer-to-Peer Architectures

Peer-to-peer (P2P) is an alternative network model to that provided by traditional client-server architecture. In the P2P networks, each machine, referred to as a peer, functions as a client with its own layer of server functionality. A peer plays the role of a client and a server at the same time. That is, the peer can initiate requests to other peers, and at the same time respond to incoming requests from other peers on the network. It differs from the traditional client-server model where a client can only send requests to a server and then wait for the server's response. [1]

The P2P networks can be classified into three main categories:

1. Centralized P2P systems
2. Decentralized P2P systems
3. Hybrid P2P systems

In this survey we attempt to explain the three different P2P architectures with an example for each type.

1. Centralized P2P Systems:

Centralized systems, like the client server architecture, have one or more central servers that are used to locate the desired resources or act like a task scheduler to co-ordinate different actions among the peers. But once the peer gets the information about the required resource, it can communicate directly with the other peers without going through the server. This architecture makes the network susceptible to single point failures and malicious attacks. Also, the centralized server becomes a bottleneck for large number of peers degrading the performance of the network. Napster is a good example of Centralized P2P architecture.

NAPSTER:

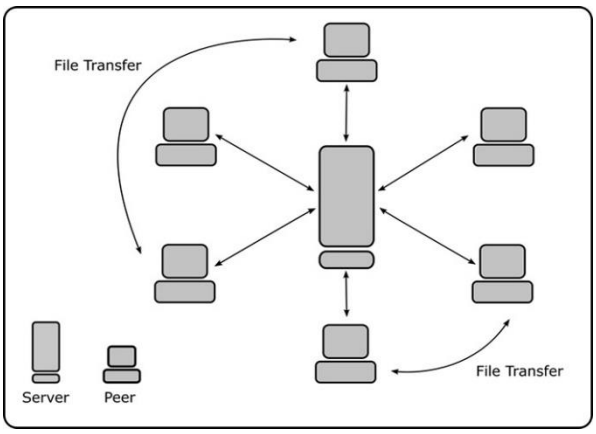
The architecture of Napster is based on the Centralized Model of P2P file-sharing. It has a Server-Client structure where there is a central server system which directs traffic between individual registered users. Napster provided three basic functions:

1. Search Engine: Finds the requested MP3 files. Acts as a dedicated server to realize the function of resource location.
2. File Sharing: Ability to trade MP3 files among the peers without using the storage space of the centralized server.
3. Internet Relay Chat: To find and chat with other peers who are online.

The central servers maintain directories of the shared files stored on the respective PCs of registered users of the network. These directories are updated every time a user logs on or off the Napster server network. The client registers with the server, providing identity and shared file information for the server's database. In turn, the client receives information about connected users and available files from the server. Each time a user of a centralized P2P file sharing system submits a request or search for a particular file, the central server creates a list of files matching the search

request, by cross-checking the request with the server's database of files belonging to users who are currently connected to the network.

The central server then displays that list to the requesting user. The requesting user can then select the desired file from the list and open a direct HTTP link with the individual computer which currently possesses that file. The download of the actual file takes place directly, from one network user to the other, without the intervention of the central server. The actual MP3 file is never stored on the central server or on any intermediate point on the network. [2]



The following are the advantages and limitations of Napster Architecture:

Security: Since this architecture includes a centralized server to store information about online users and shared files, it is vulnerable to single point failures and malicious attacks. A user can easily obtain the IP address of the peers by sending a query to the centralized server. This destroys the anonymity and privacy of the users.

Scalability: As Napster follows a centralized model, all the peers need to first connect to a server. The server needs to service the query of every peer before the file sharing can occur. If the number of queries and connections increase beyond a certain value, the server response time may be very large or may even result in denial-of-service. This imposes a limit on the amount of queries the server can service at any given time.

Efficiency and Cost of Ownership: In the client server architecture, the server stores all the files that need to be distributed. The cost for maintaining such a server is very high. The centralized server used in the Napster Architecture stores just the IP addresses and the file names of the available files and thus is very inexpensive to operate and maintain. Also, the centralized server accelerates the process of recourse location with cheaper cost and high efficiency. [3]

2.Decentralized P2P Systems:

The peers in a decentralized system have equal rights and responsibilities. The network does not have a centralized server and each peer has a partial view of the whole P2P network and offers services relevant to only a few queries or peers. Thus, quickly locating a service or peer in the network can be a critical and a challenging task. These networks are immune to single point failures and offer high level of performance, robustness and scalability.

GNUTELLA:

Gnutella is a large peer to peer network. It was the first decentralized peer-to-peer network of its kind. In this protocol there are a large number of users called as nodes, each of whom have a Gnutella client software. On the initial startup the client software must find at least one other node, various methods have been employed for this like, using the web caches of the known nodes, UDP host caches, or using pre-existing address list of possibly working nodes shipped

with the software. Once connected the client requests a list of working addresses. The client tries to connect to the nodes it was shipped with as well as the nodes that it receives from other clients until it reaches certain amount. It connects to only that many nodes, locally caching the addresses it has not yet tried and discarding the addresses it tried and were invalid.

After an address if found, the connecting node sends a request message GNUTELLA CONNECT to the already connected node. The requested node may either accept the request by sending a reply message GNUTELLA OK, or reject the request message by sending any other response back to the requesting node. A rejection can happen due to different reasons such as, an exhaustion of connection slots, having different versions of the protocol, etc. (Limewire). Once attached to the network, the node periodically pings its neighbors to discover other nodes. Typically, each node should connect to more than one node since the Gnutella network is dynamic, which means any node can go off-line or disconnect any time.

It is thus important to stay in contact with several nodes at the same time to prevent being disconnected from the network. Once a server receives the ping message, it sends back a pong message to the server that originated the ping message using the same path that the ping message came from. A pong message contains the details of the node like port, IP address, the number of files shared, and the number of kilobytes shared.

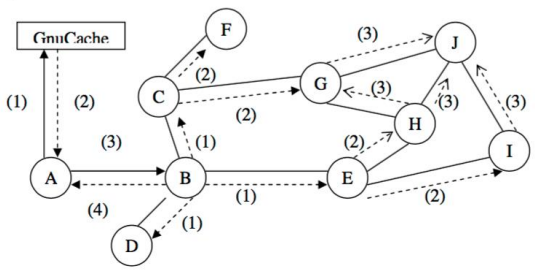
To search for a file, a query is sent to the neighbors. Once a query is received by a neighbor, the query criteria is checked against the local index directory and propagate this query message to their neighbors, and so forth.

If the check matches with the local data in any node, that node will send back the queryHit message to the query initiating node along the same path that carried the query message. However, when the node generating the queryHit message stays behind the firewall, the requesting node cannot create a connection to it. In this case, the push message will be sent by the requesting node to the node that generates the queryHit message (which stays behind the firewall) to initiate the connection instead. The file transfer is done by using HTTP protocol. [4]

To prevent flooding the network with the messages, the TTL (Time-To-Live) field is included in the header of every Gnutella message. The TTL field will be decremented by one every time the message is passed one node. The node decrementing the TTL and finding that its value equals to zero will drop that message. Each node also needs to maintain the list of recently seen messages by storing the Descriptor ID and the Payload Descriptor of each incoming message to prevent forwarding the same message repeatedly.

Let's see an example:

User A connects to the GnuCache to get the list of available nodes already connected to the network. GnuCache sends back the list to the user A. A



sends the request message GNUTELLA CONNECT to the User B. User B replies with the GNUTELLA OK message granting user A to join the network.

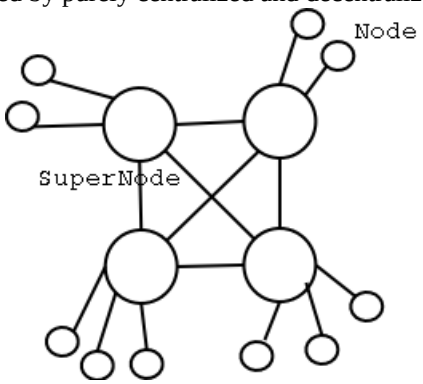
Once user A connected to the network, he wants to search for some files. Therefore, he sends a query message to his neighbor, user B. User B first checks that this is not the old message, then checks for matching with his local data. If matches, he sends the queryHit message back to user A. Else, User B decrements TTL by 1 and forwards the query message to user C, D, and E. User C, D, and E do the same things as user B and forward the query message further to user F, G, H, and I. User F, G, H, and I also do the same things, but suppose that user H is the first person who forwards the query message to user J. The subsequent query messages forwarded by user G and I to user J will be discarded by user J when he checks against his local table and finds that he already received this message. This is hold for user G as well when user H forwards the query message to him. Suppose now that user J finds matching against his local data, he sends the queryHit message back to user A by following the same path as the query message carried, that is from J, to H, to E, to B, and to A. Now, user A can initiate the file down load directly with user J by using HTTP protocol. [5]

3. Hybrid P2P Systems:

The main advantage of centralized P2P systems is that they are able to provide a quick and reliable resource locating. Their limitation, however, is that the scalability of the systems is affected by the use of servers. While decentralized P2P systems are better than centralized P2P systems in this aspect, they require a longer time in resource locating. As a result, hybrid P2P systems have been introduced to take advantages of both centralized and decentralized architectures. Basically, to maintain the scalability, similar to decentralized P2P systems, there are no servers in hybrid P2P systems. However, peer nodes that are more powerful than others can be selected to act as servers to serve others. These nodes are often called *super peers*. In this way, resource locating can be done by both decentralized search techniques and centralized search techniques (asking super peers), and hence the systems benefit from the search techniques of centralized P2P systems.

FastTrack:

FastTrack Network is a good example of Hybrid Architecture. It combines the best of Centralized and Decentralized architectures and it was designed with the aim to solve the problems faced by purely centralized and decentralized p2p systems[5]. This protocol was used by some famous p2p systems such as Kazaa and Grokster etc[6].



In this architecture, there are two tiers of control. The first tier is made up of clusters of ordinary nodes that log onto Super Nodes (ordinary machines with high speed connection). As discussed previously, this sort of connection mimics the centralized topology. The second tier consists of only Super Nodes that are connected to one another in a decentralized fashion [5].

The number of Super Nodes are not constant as they can join and leave the network as they please. In a sense, they are also peers in the network. But this requires that some nodes(peers) be always online. Such nodes are called bootstrapping nodes [5]. When a client joins a network, it contacts the bootstrapping node. The bootstrapping node then determines if the newly joined peer can be a Super Node. If it does qualify, it is provided with the IP address of all other Super Nodes. Otherwise, it is provided with the IP address of one of the Super Nodes.

When a peer wants a resource, it sends the query to the Super Node whose IP address it has. The Super Node then broadcasts this query to all the other Super Nodes that it is aware of. The broadcast stops once the Time to Live value of the query reaches zero. The TTL value is decremented at every broadcast. Every Super Node that receives the query searches through its database that contains the list of files that are connected to it. Once the required resource is found, the response is sent in the same path as the query. One can argue that the Super Node broadcast information between each other can clog or cause congestion in the network. However, we must remember that the Super Nodes by definition has a high speed connection. This ensures that the congestion, it at all it occurs, is minimal.

How is Fast Track hybrid and its advantages:

Fast Track Network implements the hybrid architecture by using the two tier network where some peers act as Super Nodes and some peers act as normal nodes. Any node can become a Super Node if and when it connection becomes good enough. Thus, the network dynamically improves its resource location mechanism by bringing in new Super Nodes. Instead of one resource directory like the case in Napster, it has many Super Nodes that help in resource location. Through this, it solves the central point of failure risk that is prevalent in Napster. Unlike Decentralized architecture, where the query has to pass through all intermediate nodes to reach the node that has the resource, here the query is broadcasted to Super Nodes. We must remember that in a decentralized architecture, the intermediate nodes have no idea if their neighboring nodes have the resource or not. The query is just passed further along. In Fast Track and similar hybrid network, the Super Nodes have already indexed what resource is available in its connected nodes. Hence, they can locate the resource carrying node easily and pass on the query.

Conclusion for Survey:

After surveying the different P2P architectures, we came to following conclusions. While the centralized architecture is good in resource locating, it is not as good as the decentralized model in enforcing scalability. However, decentralized model has resource locating limitations as no node has authority over other nodes. The hybrid model attempts to combine the best of these models while trying to solve the problems plaguing both the models.

References:

1. David Barkai (2001), An Introduction to Peer-to-Peer Computing http://www.intel.com/technology/magazine/systems/it02012.pdf
2. P2P Networks for Content Sharing, Choon Hoong Ding, Sarana Nutanong, and Rajkumar Buyya http://arxiv.org/ftp/cs/papers/0402/0402018.pdf
3. Peer to Peer Computing Principles and Applications (ISBN 978-3-642-03513-5), Chapter 2 - Architecture of peer to peer systems, pg17-18.
4. Query Routing for the Gnutella Network proposal Christopher Rohrs, 2001-12-18
5. P2P Networks for Content Sharing- Choon Hoong Ding, Sarana Nutanog, and Rajkumar Buyya. Dept. of Computer Science and Software Engineering, The University of Melbourne, Australia http://www.cloudbus.org/papers/P2PbasedContentSharing.pdf
6. https://en.wikipedia.org/wiki/FastTrack
7. Comparing Hybrid Peer-to-Peer Systems by Beverly Yang Hector Garcia-Molina. http://ilpubs.stanford.edu:8090/455/1/2000-35.pdf
8. Computer Networking- A Top Down Approach (Sixth Edition) by James F. Kurose and Keith W. Ross