ISSN: 2393-8528

IJICSE
Open Access

Contents lists available at www.ijicse.in

International Journal of Innovative Computer Science & Engineering

Volume 2 Issue 1; 2015, Page No.16-20

A survey on peer to peer system applications

Ms. Neelkamal Chaudhary, Jayshree Surolia

#1 Govt. Engineering College, Ajmer

#2, Purnima University, Jaipur

ARTICLE INFO

Received 1 April. 2015
Accepted 15 April. 2015

Corresponding Author:

Ms. Neelkamal Chaudhary

1 Govt. Engineering College, Ajmer, Rajasthan, India.

E-mail:

neelkamal.chaudhary26@gmail.com

ABSTRACT

P2P (peer-to-peer) systems gain more interest from both the user and research communities, building a search system on top of P2P networks is becoming an attractive and promising alternative. The paper introduces the concept of P2P network architecture and structure of current P2P systems. The background, scope, objectives, and methodology adopted for carrying out the research work is also presented in this paper. This paper also evaluates the extensive research done in the field of p2p networks. A survey is very important part of every research. So here this paper shows the application based survey on peer to peer systems.

INTRODUCTION

Peer-to-peer systems are distributed systems consisting of interconnected nodes able to self-organize into network topologies with the purpose of sharing resources such as content, CPU cycles, storage and bandwidth, capable of adapting to failures and accommodating transient populations of nodes while maintaining acceptable connectivity and performance, without requiring the intermediation or support of a global centralized server or authority1].

- **High availability** - Centralized search systems are vulnerable to distributed denial-of-service attacks. However, P2P search tends to be more robust than centralized search as the demise of a single node or some nodes is unlikely to paralyze the entire search system. Furthermore, it is not easy for an attacker to bring down a significant fraction of geographically distributed P2P nodes.

- **Low cost and easy of deployment**- Centralized search engine requires a huge amount of investment in both hardware and software, as well as in maintenance. A P2P search system, however, is virtually free by pooling together slack resources in P2P nodes and can be deployed incrementally as new nodes join the system[2].

- **Data freshness** - In centralized search systems, it usually takes weeks for newly updated data to enter the data center that hosts the search database. Also due to the bandwidth constraints between the data center and the Internet there is no freshness guarantee on the index maintained in the centralized database (i.e., weeks delay). On the other hand, P2P nodes can publish their documents immediately once they appear, and the publishing traffic goes to geographically distributed nodes, thereby avoiding the bandwidth constraints imposed by centralized search systems.

- **Good scalability** - The exponentially growing data added each year would be beyond the capability of any search engine. However, the self-organizing and scalable nature of P2P systems raises a hope to build a search engine with very good scalability.

I. P2P NETWORK ARCHITECTURE

P2P network architecture 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. Three categories are identified as:

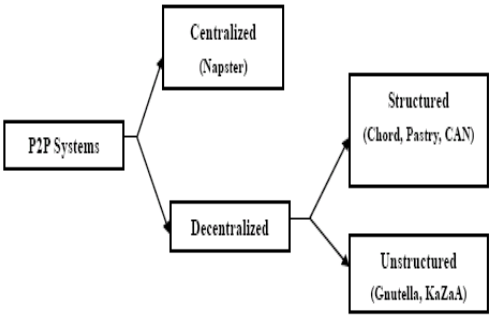

Fig: Classification of Peer-to-Peer Architecture

- **Purely decentralized systems** - All nodes in the network perform exactly the same tasks, acting both as servers and clients, and there is no central coordination of their activities. The nodes of such networks are often termed "SERVENTS" (SERVers+ clients). Example networks are original Gnutella architecture and Freenet.This kind of architecture requires all peers to act as both a client and a server. All nodes are therefore "assigned" the same tasks and there is no central point of failure [3].

The nature of these networks requires that the index cannot be stored on a central server. This means that it can be either distributed or local. Networks which adopt the local index logic require each node to hold an index for its own content. As an early P2P implementation for a distributed index network was Freenet. The early versions of Gnutella (v0.4) used a locally stored index which is however a terribly inefficient technique []. In order for a peer to find available content in the network the peer has to flood the whole network by broadcasting its request and wait for a response from the nodes that have the required content.This of course renders the network almost unable to scale to larger sizes. However, this also means that the network is very fault tolerant and if a node fails or just disconnects this would not affect the network. When a node wants to join the network the only requirement is for it to connect to any existing and active peer. Some techniques to improve the scalability issues will be discussed later in the thesis.

It is important to note that under this classification, besides the Gnutella-type networks, there are other networks also that are formed deterministically. Such networks form connections between their peers that are somehow organized, structured. However, "structured" in this case does not refer to the network topology but merely to the fact that peers do not join the network at random locations, as with Gnutella for example, but deterministically take a position in the decentralized network.

- **Partially centralized systems** -The basis issame as the one with purely decentralized systems. However, some of the nodes are assumed to play a more "important" role than the rest of the nodes, acting as local central indexes for files shared by local peers. These nodes are called "Supernodes", and the way in which they are selected for these special tasks vary from system to system shown in figure FF. It is important to note that these Supernodes do not constitute single points of failure for a p2p network, since they are dynamically assigned and in case they are subject to failure or malicious attack the network will take action to replace them with others. Example networks are Kazaa, Morpheus and more recent Gnutella.

Generally peers are automatically elected to become supernodes if they have sufficient bandwidth and processing power. Supernodes index the files shared by peers connected to them, and proxy search requests on behalf of these peers. All queries are therefore initially directed to supernodes. Two major advantages of partially centralized systems are that:

- Discovery time is reduced in comparison with purely decentralized systems, while there still is no unique point of failure. If one or more supernodes go down, the nodes connected to them can open new connections with other supernodes, and the network will continue to operate.
- The advantage of inherent heterogeneity of peer-to-peer networks is exploited. In a purely decentralized network, all of the nodes will be equally (and usually heavily) loaded, regardless of their CPU power, bandwidth, or storage capabilities. In partially centralized systems, however, the supernodes will undertake a large portion of the entire network load, while most of the other (so called "normal") nodes will be very lightly loaded, in comparison.

Kazaa is a typical, widely used instance of a partially centralized system implementation (as it is a proprietary system, there is no detailed documentation on its structure and operation). Edutella is another partially centralized architecture. The research [Yang and Garcia-Molina] addresses the design and searching techniques for partially centralized peer-to-peer networks. The concept of supernodes has also been proposed in a more recent version of the Gnutella protocol. A mechanism for dynamically selecting supernodes organizes the Gnutella network into an interconnection of *superpeers* and client nodes.

When a node with enough CPU power joins the network, it immediately becomes a *superpeer* and establishes connections with other superpeers, forming aflat unstructured network of superpeers. If it establishes a minimum required numberof connections to client nodes within aspecified time, it remains a *superpeer*. Otherwise,it turns into a regular client node.

- **Hybrid decentralized systems -** There is a central server facilitating the interaction between peers by maintaining directories of the shared files stored on the respective PCs of registered users to the network, in the form of meta-data. The end-to-end interaction is between two peer clients; however these central servers facilitate this interaction by performing the lookups and identifying the nodes of the network (i.e. the computers) where the files are located. The terms "peer-through-peer" or "broker mediated" are sometimes used for such systems. Obviously in these

architectures there is a single point of failure (the central server). This makes them vulnerable to censorship, technical failure or malicious attack, which in itself is enough to defeats the purpose of p2p as we view it.

Each client computer stores contents (files) shared with the rest of the network. All clients connect to a central directory server that maintains:

- A table of registered user connection information (IP address, connection bandwidth etc.)
- A table listing the files that each user holds and shares in the network, along with metadata descriptions of the files (e.g. filename, time of creation, etc.)

A computer that wishes to join the network contacts the central server and reports the files it maintains. Client computers send requests for files to the server. The server searches for matches in its index, returning a list of users that hold the matching file. The user then opens direct connections with one or more of the peers that hold the requested file, and downloads it (see figure FF). The advantage of hybrid decentralized systems is that they are simple to implement, and they locate files quickly and efficiently. Their main disadvantage is that they are vulnerable to censorship, legal, action, surveillance, maliciousattack, and technical failure.

The reason that the shared content or at least descriptions of it, and the ability to access contentis controlled by the single institution, company, or user maintaining the central server. Furthermore, these systems are considered inherently un-scalable, as there are bound to be limitations to the size of the server database and its capacity to respond to queries. Large Web search engines have, however, repeatedly provided counterexamples to this notion. Examples of hybrid decentralized content distribution systems include the notorious Napster and Publius systems that rely on a static, system-wide list of servers. Their architecture does not provide any smooth, decentralized support for adding a new server, or removing dead or malicious servers. It should be noted that systems that do not fall under the hybrid decentralized category may still use some central administration server to a limited extent, for example, for initial system bootstrapping or for allowing new users to join the network by providing them with access to a list of current users (e.g. gnutellahosts.com for the gnutella network).

II. APPLICATIONS:

A. Napster (Hybrid decentralized unstructured systems)

Hybrid decentralized systems are usually not considered to be real p2p systems as it constitutes a central server.

- **Background of Napster:**The Internet was originally built as a peer-to-peer system in the late 1960s to share computing resources within the US. The first hosts on ARPANET were connected together as equal peers rather than as client-server. The main users at the beginning were computing researchers who did not need protection against each other, and security break-ins were practically non-existent, making the Internet much less partitioned than it is today. Many peer-to-peer systems were widely used and still in existence today such as Usenet and DNS. In 1994 however, the structure of the Internet changed dramatically with millions of people flocking to the Net. Modem connection protocols like SLIP and PPP were widely used and applications were now targeting slow speed analog modems. Applications such as web browsers were based on a client-server protocol. The Structure of the Internet made a switch from peer-to-peer to the client-server model.

Before Napster there were online recordings on the Net. Using MP3 compression format music tracks could be transferred onto disk files and then published on a website and people would download them using FTP. The one major problem with this was that up-to-date MP3 files were difficult to find. Napster solved this problem by providing constant up-to-date MP3 files in a single location that everyone knew about. Users could register with the searchable Napster network name space and find files easily through Napster servers, which had information on registered hosts and MP3 data. The servers dealt with the transfer of files between clients but didn't actually store any of the music themselves. Napster's network protocol created direct peer-to-peer access between clients. It is the simplicity of use that peer-to-peer provides that helped with the success of Napster and other applications that use it.

- **Napster Architecture:** Peer-to-Peer (P2P) is a form of distributed computing that can be described as the sharing of computer resources such as files, MP3s etc. and computer services by direct transfer between two computer systems. Traditionally, the exchange of resources and services between computer systems is done using Client-Server techniques. With P2P systems, there is no such dominant server; control is decentralized. Each node or peer on the network may act as both a client and server. Clients in a P2P network can interact freely with other clients without the intervention of a server although sometimes there is the presence of a directory server (which stores IP addresses and other information about the computers in the network) for look up purposes. The figure FF depicts the process of file transfer in a P2P network.

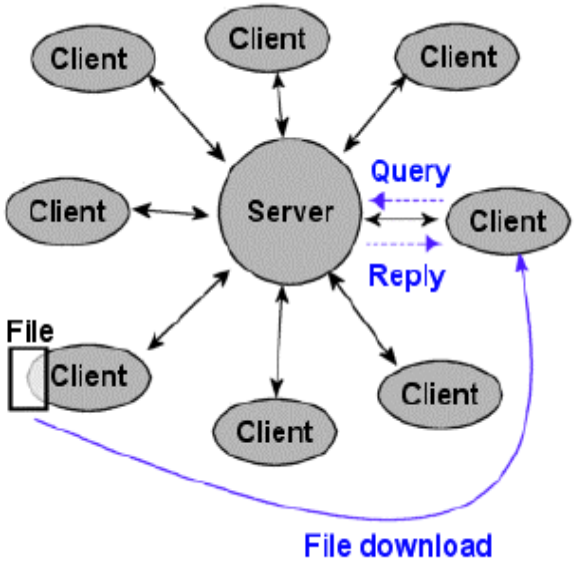

Figure: Illustration of the architecture of Napster.

The system comprises a network of registered users running some client software, and a central directory server maintaining:

- An index with metadata (file name, time of creation etc.) of all the files in the network.
- A table of registered user connection information (IP addresses, connection speeds etc.)
- A table listing the files that each user holds and shares in the network.

A. **Gnutella (Purely decentralized unstructured systems)**

Gnutella is a very simple file sharing protocol that uses the principles of peer-to-peer networking to allow users to share data. It became public domain through a process of reverse engineering of an experimental P2P client developed by Nullsoft. Although the company intended to release the specification of the protocol under the GPL at a later stage, it never came to that due to legal concerns. It is thanks to the open-source clients that appeared shortly after the protocol had been cracked, that Gnutella still appears on the P2P map today.

- **History:**

The first client was developed by Justin Frankel and Tom Pepper of Nullsoft in early 2000, soon after the company's acquisition by AOL. On March 14, the program was made available for download on Nullsoft's servers. The event was prematurely announced on Slashdot, and thousands downloaded the program that day. The source code was to be released later, under the GNU General Public License (GPL).The next day, AOL stopped the availability of the program over legal concerns and restrained Nullsoft from doing any further work on the project. This did not stop gnutella; after a few days, the protocol had been reverse engineered, and compatible free and open source clones began to appear. This parallel development of different clients by different groups remains the *modus operandi* of gnutella development today.

Queries are propagated in the same manner, with positive responses being routed back the same path. When a resource is found and selected for downloading, a direct point to point connection is made between the client and the host of the resource, and the file downloaded directly using HTTP. The server in this case will act as a web server capable of responding to HTTP GET requests.Gnutella packets are of the form:

Message ID (16bytes)	**Function ID** (1byte)	**TTL** (1byte)	**Hops** (1byte)	**Payload length** (4bytes)

Where,

Message ID in conjunction with a given TCP/IP connection is used to uniquely identify a transaction.
Function ID is one of theAdvertisement [response], Query[response] or Push-Request.
TTL is the time-to-live of the packet, i.e. how many more times the packet will be forwarded.
Hopscount the number of times a given packet is forwarded.
Payload length is the length in bytes of the body of the packet.

B. Kazaa, Morpheus (Partially centralized unstructured systems)

Kazaa and Morpheus are two similar partially centralized systems which use the concept of “SuperNodes”, i.e. nodes that are dynamically assigned the task of servicing a small subpart of the peer network by indexing and caching files contained in the part of the network they are assigned to. Both Kazaa and Morpheus are proprietary and there is no detailed documentation on how they operate. Kazaa Media Desktop was commonly used to exchange MP3 music files and other file types, such as videos, applications, and documents over the internet. The Kazaa Media Desktop client could be downloaded free of charge; however, it was bundled with adware and for a period there were "No spyware" warnings found on Kazaa's website. During the past few years, Sharman Networks and its business partners and associates were the target of copyright-related lawsuits, related to the copyright of content distributed via Kazaa Media Desktop on the FastTrack protocol.

C. Freenet (Purely Decentralized Loosely structured systems)

Freenet is a purely decentralized loosely structured system, operating as a self-organizing p2p network. It essentially pools unused disk space in peer computers to create a collaborative virtual file system providing both security and publisher anonymity.

In this respect, a main difference from other systems such as Gnutella is that Freenet provides file-storage service, rather than file-sharing service. Whereas in Gnutella files are only copied to other nodes when these nodes request them,

Files in Freenet are identified by binary keys. There are three types of keys: keywordsignedkeys, signed-subspace keys and content-hash keys. To search for a file, theuser sends a request message specifying the key and a timeout (hops-to-live) value.Messages in Freenet always include an ID (for loop detection), a hops-to-live value,source and destination, and are of the following types:

- Data request. Additional field: Key.
- Data reply. Additional field: Data.
- Data failed. Additional fields: Location and reason.
- Data insert. Additional fields: Key and data.

Each Freenet node maintains a common stack storing:

- ID: File identifier
- Next hop: Another node that stores this ID.
- File: The file identified by the id, stored on the local node.

D. Chord (Purely Decentralized Structured systems)

Chord is a decentralized P2P lookup service that stores key/value pairs for distributed data items. Given a key, thenode responsible for storing the key's value can be determined using a hash function that assigns an identifier toeach node and to each key (by hashing the node's IP address and the key). Each key k is stored on the first nodewhose identifier id is equal or follows k in the identifier space. Depending on the application using Chord, that node might be responsible for storing a value associated with the key. Chord uses a variant of consistent hashing to assign keys to Chord nodes. Consistent hashing tends to balance load, since each node receives roughly the same number of keys, and involves relatively little movement of keys when nodes join and leave the system.

The Chord protocol specifies how to find the locations of keys,how new nodes join the system, and how to recover from the failure(or planned departure) of existing nodes. This section describes asimplified version of the protocol that does not handle concurrentjoins or failures.

Chord improves the scalability of consistent hashing by avoidingthe requirement that every node know about every other node.A Chord node needs only a small amount of "routing" informationabout other nodes. Because this information is distributed, anode resolves the hash function by communicating with a few othernodes. In an N-node network, each node maintains informationonly about$O(log\ N)$ other nodes, and a lookup requires $O(log\ N)$ messages. Chord must update the routing information when a node joins orleaves the network; a join or leave requires $O(log^2 N)$messages.

CONCLUSION

Peer-to-peer (P2P) is an alternative network model to that provided by traditional client-server architecture. P2P networks use a decentralised model in which each machine, referred to as a peer, functions as a client with its own layer of server functionality. First section introduces the concept of Peer to Peer network and their several principles and classification based on architecture and structure of the network. Then, second section presents the literature review of the searching techniques in unstructured Peer to Peer systems. Third section presents the application based analysis of p2p systems.

REFERENCES:

1. Xiuqi Li and Jie Wu, Searching Techniques in Peer-to-Peer Networks, Department of Computer Science and Engineering, Florida Atlantic University.
2. V. Vishnumurthy and P. Francis. A comparison of structured and unstructured P2P approaches to heterogeneous random peer selection. In Proc. Usenix Annual Technical Conference, 2007.
3. Tsoumakos and N. Roussopoulos. Adaptive Probabilistic Search (APS) for Peer-to-Peer Networks. Technical Report CS-TR-4451, Un. of Maryland, 2003.
4. Imad Jawhar and Jie Wu, A Two-Level Random Walk Search Protocol for Peer-to- Peer Networks, Department of Computer Science and Engineering, Florida Atlantic University.
5. Beverly Yang Hector Garcia-Molina, Improving Search in Peer-to-Peer Networks, Computer Science Department, Stanford University.