P2P Distributed System

Kirti Dewan
M.E. Computer Science(2nd year)
Birla Institute of Technology and Science, Pilani, Hyderabad Campus
India

Varun A M
M.E. Computer Science(2nd year)
Birla Institute of Technology and Science, Pilani, Hyderabad Campus
India

Harsh Vijay Vani
M.E. Computer Science(2nd year)
Birla Institute of Technology and Science, Pilani, Hyderabad Campus
India

***Abstract*—To develop our own p2p network application from scratch using python. The application is to share files between clients using socket programming.**

Keywords— p2p , python , socket programming, file sharing.

INTRODUCTION

The motive of the project is to make a simple peer-to-peer network in any language. We choose to work with python. The application has a central server that basically holds the information about the clients that are present in the network. Any number of clients can join the network by first connecting to the server and later get files which are present in the network. The server holds the metadata about each client which includes their ports and the list of files they offer.

I. MOTIVATION

Storage system performance is a critical component in the development of data-intensive applications. When files reside on a central server, it is efficient to find the location of the file, but the server can be overwhelmed by a large number of requests for data. Thus, distributing the files across multiple servers may relieve that bottleneck, but it may also impose additional overheads in file discovery and retrieval.

Some of the overheads associated with parallel or distributed file systems occur because of the consistency constraints imposed on concurrent file access.

II. NETWORK MODELS

A. *Client-Server Model*

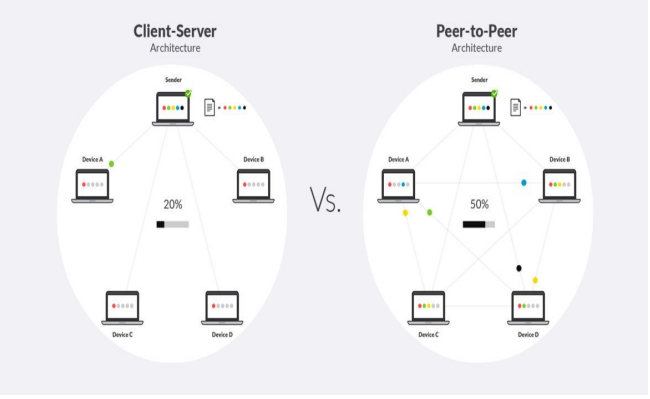
A client-server network involves multiple clients, or workstations, connecting to at least one central server. Most data and applications are installed on the server. When clients need access to these resources, they access them from the server. Servers often have private user directories as well as multiple public directories. Client-server networks tend to have faster access speeds because of the large number of clients they are designed to support. The clients are allowed to function as workstations without sharing any resources. It is easier to upgrade software applications and files because they are held on one single computer. System-wide services can be provided through the server software. Security is enhanced on a client server network because the security is handled by the server.

B. *Peer-to-peer*

Peer-to-peer networks involve two or more computers pooling individual resources such as disk drives, DVD players and printers. These shared resources are available to every computer in the network. Each computer acts as both the client and the server, communicating directly with the other computers. On a peer-to-peer network, for example, a printer on one computer can be used by any other computer on the network. These networks are inexpensive to set up. All you need is a way to connect them, like an Ethernet cable or a Wi-Fi router.

III. PEER TO PEER NETWORK ARCHITECTURE

In peer to peer networks the data is still transferred through the physical layer but it is at the application layer where peers are able to communicate with each other directly. A P2P Peer to Peer architecture is designed around several nodes taking part equally acting as both the client & the server. The data is exchanged over TCP IP just like it happens over the HTTP protocol in a client-server model. The P2P design has an overlay network over TCP IP which enables the users to connect directly & takes care of all the complexities and the heavy lifting. Nodes/Peers are discoverable & indexed in this overlay network. We have designed a centralised p2p. The proposed architecture is explained in the next section.

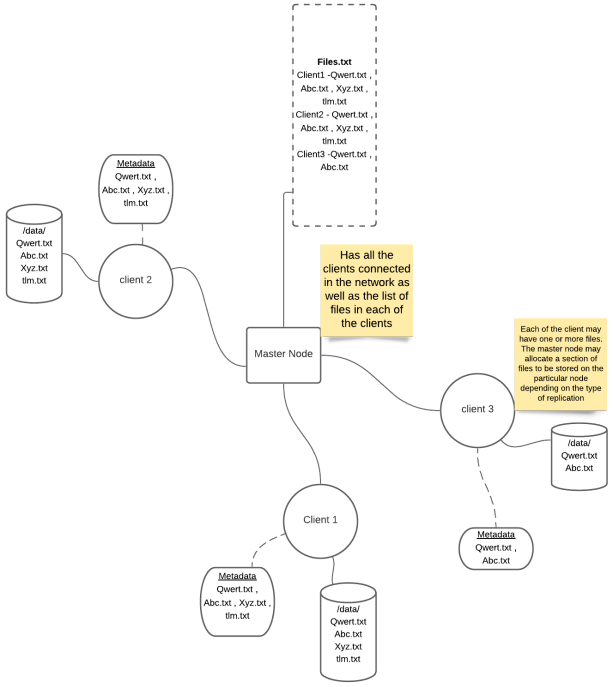


IV. Proposed System

The proposed system is a centralized P2P system. We call the central peer as master and all the others as nodes. A typical system may have any number of nodes however it is recommended there are at least 2 nodes to see the effect of the distributed database properties. Among the various features of distributed database systems, we have implemented replication and fault tolerance.

We have made the master and the node communicate using socket programming in python. The server is always listening for connection from clients.

For replication, our condition is that all files need to be present in all the nodes. Fault tolerance is achieved by making sure if a file is deleted in one of the nodes, then it is automatically brought back to the node.



Each of the nodes further run a http server that serves the files for the database users to access. It is important to note that unlike typical P2P models where everyone is a client/server, here only the database is a P2P implementation and the users are not a part of the database. That is, the users DO NOT act as servers. This also means that the implementation is a better choice for proprietary implementation rather than open source system unlike torrents.

V. Working

A central server will listen to connections from clients, once a client connects, it sends some connection information to the server. The server makes a note of this connection information for later use. Once this is done, the client at preset intervals, which is 5 seconds in our implementation, updates its meta data in the file called meta.txt. Each of the clients will have its own metadata file.

The server on the other hand, will use the client connection information that it has stored to visit the metadata of each of the clients, again at preset intervals of 5 seconds and update its own metadata file. Therefore the server will always have the global knowledge of the entire database.

Now, if a new client connects, the server will first take the metadata information from this client. It then updates the global metadata information, and for each client already existing in the system it checks what files are not present in the node and copies the file to that node.

The end effect of this process is that all the nodes will have all the files. We have made use of python sets to effectively make sure there are no duplicates either in the files across the database, or in the ports when clients connect.

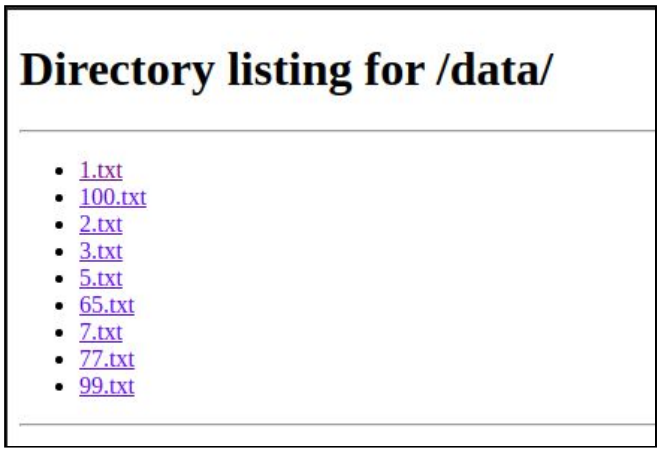
Most of the work is implemented using threads to get a feel of parallel execution.

VI. Results

```
pc@pc:~/Documents/Projects/DDS/own/master$ python3 server.py
socket binded to 5558
Socket listening
Use one of the mirrors to download files set()
File set is
set()
File set is
set()
```

```
Got connection from ('127.0.0.1', 10000)
File set is
set()
serv dict {10000: ['1.txt', '100.txt', '2.txt', '3.txt', '5.txt', '65.txt', '7.txt', '77.txt', '99.txt', '']}
File set is
['65.txt', '1.txt', '100.txt', '99.txt', '2.txt', '5.txt', '7.txt', '3.txt', '77.txt']
serv dict {10000: ['1.txt', '100.txt', '2.txt', '3.txt', '5.txt', '65.txt', '7.txt', '77.txt', '99.txt', '']}
```

```
166722696978364697531418663580800658900 5
166722696978364697531418663580800658900 5
166722696978364697531418663580800658900 5
166722696978364697531418663580800658900 5
166722696978364697531418663580800658900 5
166722696978364697531418663580800658900 5
166722696978364697531418663580800658900 5
166722696978364697531418663580800658900 5
166722696978364697531418663580800658900 5
166722696978364697531418663580800658900 5
177621137746098305764006469705575876856 2
205227264239312567259686095155122193554 0
```



References

[1] Tyson, J. (2018). How the Old Napster Worked. [online] HowStuffWorks. Available at: https://computer.howstuffworks.com/napster2.htm [Accessed 29 Jun. 2018]

[2] Neagu, C. (2017). Simple questions: What is P2P (peer-to-peer) and why is it useful? | Digital Citizen. [online] Digital Citizen. Available at: https://www.digitalcitizen.life/what-is-p2p-peer-to-peer [Accessed 21 Jul. 2018]

[3] I. Lee, Y. He and L. Guan, "Centralized P2P Streaming with MDC," 2005 IEEE 7th Workshop on Multimedia Signal Processing, Shanghai, 2005, pp. 1-4, doi: 10.1109/MMSP.2005.248598.

[4] Carmen Carmack "How BitTorrent Works" 26 March 2005., HowStuffWorks.com. Retrieved from https://computer.howstuffworks.com/bittorrent.htm