Encrypted Peer To Peer File Sharing System

CMSC 626 Principle of Computer Security

A Project Report

Submitted by

Aayush Jannumahanti (aayushj1@umbc.edu) - FK32171
SreeVikas Edukulla (sedukul1@umbc.edu) - PY32577
Varshitha Palakurthi (vpalaku1@umbc.edu) - FA17137
Hanumantha Rao Somaraju (da16976@umbc.edu) - DA16976

Under the Guidance of
Professor Gerald Tompkins

Department of Computer Science and Electrical Engineering

December 2022

The University of Maryland Baltimore County

ABSTRACT

In comparison to the more popular server-client strategy, the encrypted peer-to-peer technique is more robust and offers a variety of advantages, including peer provision of resources like bandwidth, storage, and computing power. Resource effectiveness and resilience are essential in the file distribution process, especially for big files. The initiative aims to put the peer-to-peer sharing mechanism into practice.

Due to heavy traffic or a variety of other causes, clients may submit requests to a single processing unit (server) to access data, but they might not get a prompt answer. To solve this problem, the distributed file system, which unifies several nodes (computing and storage units), is utilized. Even if a single node in a distributed system is unable to accommodate a client's request to access data, other nodes (which are a member of the distributed file system) can manage it.

We have built a menu for a peer-to-peer sharing method, from which peers may choose the options to execute file creation, file searching, file restoration, download, delete, and read and write operations on file content. By granting the files encryption and authorisation, we have created a secure environment for this implementation. We have implemented a file locking system so that the file will lock for one request to perform and unlock as soon as the request is done in order to prevent overloading from peers' requests on a single file.

ACKNOWLEDGEMENT

We would like to express our deepest gratitude to our professor, Greald Tompkins for his guidance, everyday encouragement, intricate caring, timely help and providing us with a humble and helpful atmosphere throughout the project. We are also thankful to our teaching assistant Ran Liu and our grader Anirudh Balaiah Mahesh. In the process of completion of the project, we have had the opportunity to learn many new technologies from them which has helped us a lot in understanding the depth of the project.

Aayush Jannumahanti - FK32171
SreeVikas Edukulla - PY32577
Varshitha Palakurthi - FA17137
Hanumantha Somaraju - DA16976

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 GENERAL INTRODUCTION

Peer-to-peer (P2P) technologies are often utilized for content sharing. P2P file-sharing applications like Napster, WhatsApp, Gnutella, Freenet, and others are some examples that are currently accessible. The concept of files being distributed across nodes is the foundation of this system's architecture. The P2P system differs from older Client/Server Models, when data was housed on a single Central Server and all file transfers occurred entirely between the Central Server and the Clients. Between certain Nodes or Peers, files can be transmitted via a P2P file sharing program.
Users may access multimedia content like movies, music, e-books, games, etc. thanks to the P2P file-sharing mechanism utilized in computer networking. The individual users that make up this network are known as peers. In order to request files from other peers, the peers establish TCP or UDP connections.

1.2 DESCRIPTION

The Project is set up using an encrypted P2P paradigm, which includes a Central Index Server to gather meta-data such as Peer ID and the Peer on which the material is stored, as well as the file titles and location. In this paradigm, Peers interact with the Central Index Server to exchange files, search for files, and find out which files are on other Peers and are accessible for download. This paradigm states that all peer-to-peer file transfers must always be done through a direct data connection established by a Socket between the peers sending and receiving the file.

CHAPTER 2

SYSTEM ANALYSIS

2.1 PROBLEM STATEMENT

To design, develop and produce an Encrypted Peer-To-Peer system for robust communication, transfer and sharing of files. To create an encrypted peer-to-peer file sharing system and discover its internal workings. Furthermore, it is advisable to become familiar with the ideas of Sockets, Processes, Threads, and Makefiles.

2.2 REQUIREMENTS

HARDWARE REQUIREMENTS

A. Single system for simulating the Central Index Server
B. Multiple systems to simulate the Peers
C. In case of a single system, Multiple Virtual Machines to simulate the Central Index Server and the Peers

SOFTWARE REQUIREMENTS

A. Any Generic Machines like Windows,Linux,Mac can be used to run both the Central Index Server as well as the Peers
B. Java Development Kit (JDK), Java Runtime Environment,SQL(oracle)

PROGRAMMING LANGUAGE USED

Java and SQL

CHAPTER 3

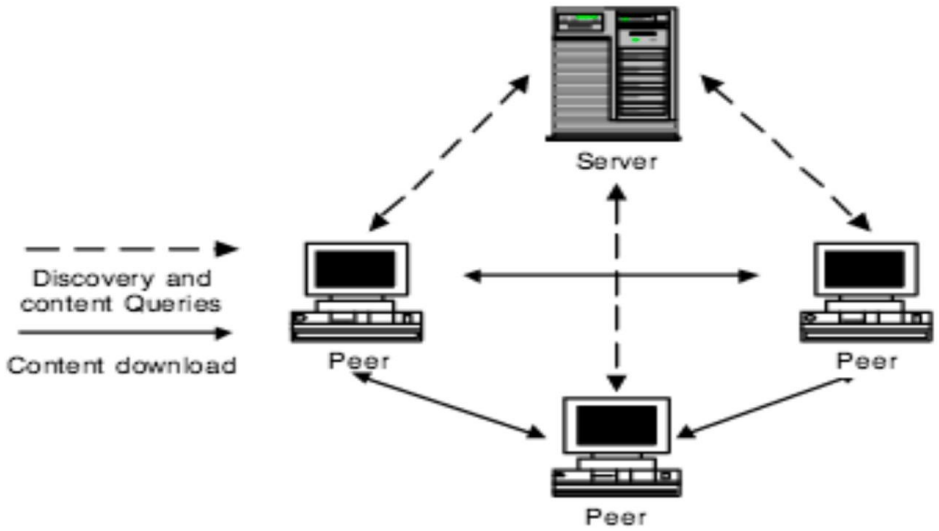
PROPOSED METHODOLOGY

3.1 SYSTEM ARCHITECTURE

The System can be used as a server-mediated P2P system or a hybrid P2P system. This P2P architecture functions exactly like the pure P2P design, with the exception that peer discovery and content lookup are handled by a Central Index Server.

In this proposed methodology, When the P2P file sharing program registers or tries to search the content on the Central Index Server, it typically notifies the Central Index Server of its existence. Instead of sending queries to each peer, the application (peer application) uses this server to search for certain contents like files and makes use of the Central Index Server. The peer application can then get in touch with those peers directly to download/retrieve the required content after the Central Index Server replies with a list of peers that have it. By requiring only a message to be sent to the Central Index Server instead of all peers for peer discovery and content search, this paradigm makes it simpler to scale this solution more effectively than the pure P2P model.

The Figure below shows the P2P architecture -



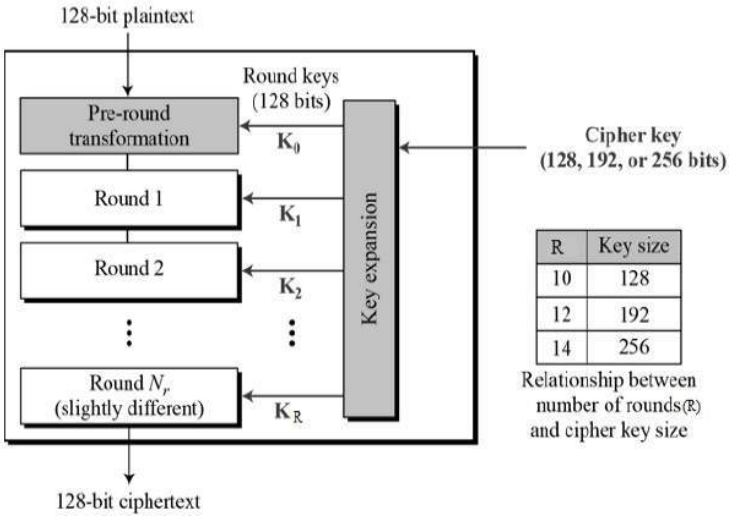
We are encrypting file and file names in our systems and we are using the AES algorithm in our project. Initially we have used DES and later we found a few shortcomings in this algorithm as it was vulnerable to attacks. A replacement for DES was needed as its key size was too small. With increasing computing power, it was considered vulnerable against exhaustive key search attacks. Triple DES was designed to overcome this drawback but it was found slow.

The features of AES are as follows −

- Symmetric key symmetric block cipher
- 128-bit data, 128/192/256-bit keys
- Stronger and faster than Triple-DES
- Provide full specification and design details
- Software implementable in C and Java

Operation of AES

The following graphic provides the AES structural schematic



R	Key size
10	128
12	192
14	256

Relationship between number of rounds(R) and cipher key size

3.2 PROTOCOL

The Napster protocol provides the basis for this paradigm. Peer-to-peer and peer-to-server communication are separated into their own sections in the protocol. Every connection, every query, and every reply travels via the socket.

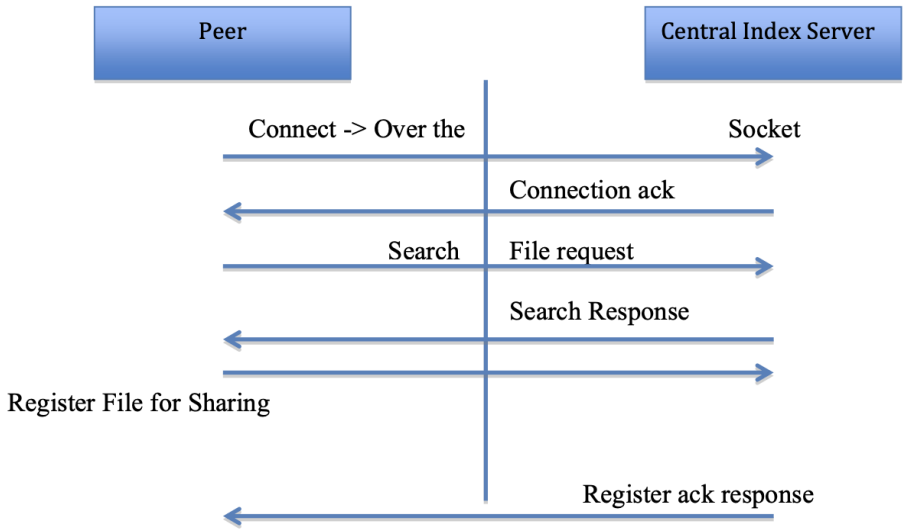


Figure: Displays the logical relationship between the central index server and the peer

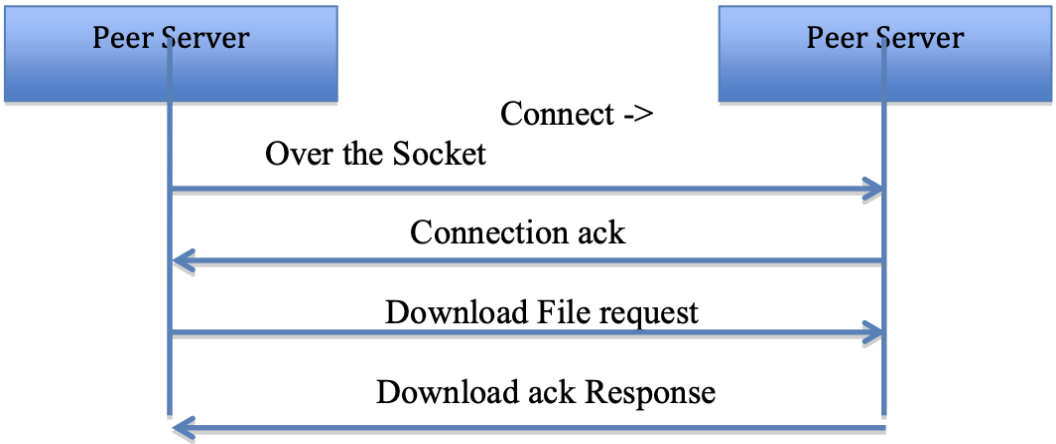


Figure: Shows the Logical flow between the two Peer Servers

3.3 DESIGN

The P2P file sharing system was created with the P2P architecture and its supporting protocols in mind.
Java is utilized to implement the entire design, and Sockets and Threads are two of the abstractions that are employed.

The P2P file sharing system has two components

1. **Central Index Server**: All of the peers that register with this server get their content indexed. It also offers the peers a search function.
   The Central Index Server offers the Peers the following Interfaces:
   - A peer may use the **Registry (peer id, filename)** to register its files with the Central Index server. After then, the CIS creates an Index for the peers.
   - **Search (filename)** – this procedure searches the index and gives the requestor a list of all matched peers.
   - **Create** - used to create new file and register these File IPs and attributes in Database

2. **Peers:** The peer performs both client and server roles. Using "lookup," the user communicates the filename as a client with the indexing server. A list of every other peer that has the file is then returned by the indexing server. The client then establishes a connection with the user's chosen peer and downloads the file. The peer acts as a server, waiting for requests from other peers and sending the requested file in response.
   The Peer Server provides the following interface to the Peer client:
   a) **Obtain/Download (Peer ID, filename)** - invoked by a peer to download a file from another peer.

CHAPTER 4

IMPLEMENTATION

4.1 Operations and Results

1. **Create New File**

   The peer has access to the IP address of the main index server while using this functionality. By connecting to the central index server, which is handled by the multi-threading idea, several peers can generate a file at the same time. The peer connects to the server, and the server authenticates the peer before allowing the peer to create a new file. To provide this behavior, we are utilizing the server end of a particular port, 2001. By inputting the name, the peer can create a directory or file, and the server will determine whether the name is already present in the database entries. If a directory or file does not already exist, a new directory or file with the specified name and the specified content is created and explicitly prompts the user to choose the permissions.The permissions include:

   - Read only

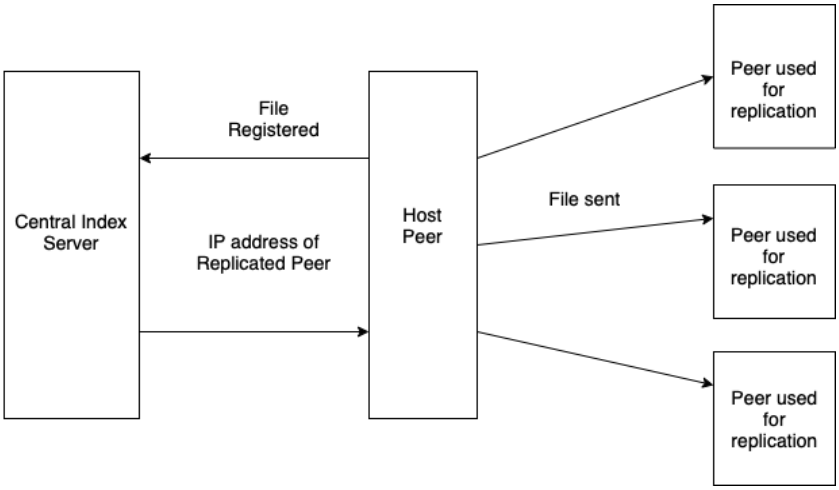
     Here the file created by the peer can only be read by the other peers.
   - Read and write only

     Here the other peers have the access to both read and write to the file created by the peer.
   - Private

     For this only the peer who created the file has access to read or write to the file.

Both the file name and the content created are encrypted and stored in peer. This encryption is done by using the AES algorithm.

After a file is created, we are storing the file information in the database i.e. filename, file permissions, peer ip. Peer ip and the file name together acts as a primary key.



2. **File Update/Write**

   When a peer requests to update a file by providing the file name, this functionality first verifies that the file actually exists. If it does, it then checks the permissions that have been assigned to the file, and if the permissions are satisfied (read and write), the peer is then able to access the file and update the content in it. All copies of the file that are currently in the system have the updated content. This is accomplished by retrieving all of the database's peers with which the file is associated and then updating the file with the new information. Here, the concurrency of files is also addressed.
   We use a specific port 2003 at the central index server and port 9003 at peer.

3. **File Read**

   In order to use this functionality, the peer must send a request with a filename to the central index server. If a file is entered, the central index server determines whether or not it is already existent in the database. If the file is in the database, the answer is given by listing every peer who is in possession of the file. To access the file and read it, we may choose the peer with whom we wish to connect.

4. **File Delete**

   We may remove files according to this function. We have a distinct column in our database called deleted that is toggled to 1, meaning the file is no longer available to that peer, when the peer selects the file name to be removed.

5. **File Search**

   A request is sent to the central index server with the file name when a peer gives the file name to see whether it already exists. The central index server uses a particular port to

listen for search functionality, which verifies the database's file availability. If the file is present, the list of peers who have it is returned to the requesting peer in response.

6. **Restore File**
   The peer enters the deleted file name if it wants to restore the file. The central index server receives this request and retrieves all of the peers that own this file. The connection request to restore the file can then be sent by the requesting peer to any of the peers to restore the file.

RESULTS

Main menu:

```
Enter The Option :
==================
1. Registering the File
2. Search File in System
3. Downloading File From Peer
4. Delete File From System
5. Restore File
6. Create a New File
7. Update File
8. Read File
9. Exit
```

File Create:

```
Enter File Name
PcsDemo.txt

Searching in server

FILE Does Not Exist !!

Enter enter content to insert into File
This is for Pcs Demo
Assign File Permission for File Type
1 - read only
2 - read and write
3 - private
2

Connected to Register on CentralIndxServer on port 2001

File Registered in System

----------------------------------------------------------------------
```

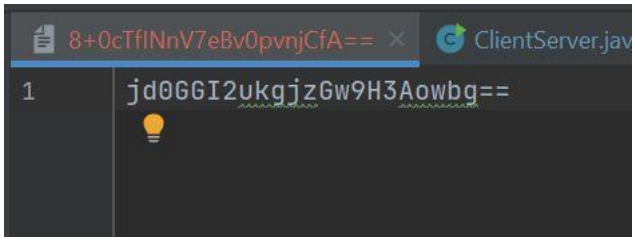
File Download:

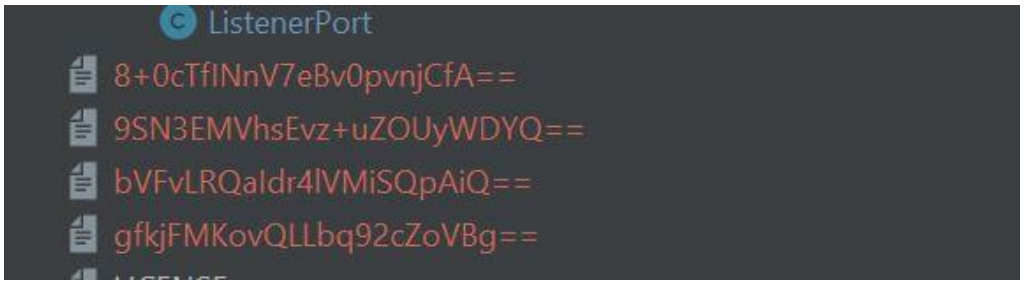
```
8
Reading File.
Give Ip address of to file
127.0.0.1
Enter the File Name to be Downloaded:
PcsDemo.txt

Connected to peerid :

Connection Received From 127.0.0.1 For Download

This is for Pcs Demo
```

File Update:

```
7
Enter File Name to Update
PcsDemo.txt

Update Request Sent To Sever

You can add your content
Enter new Content to insert
updating the file

Update Replicated files
```

Encrypted FileName:





	FILENAME	HOSTIPADDRESS	PEERID	DEL...	FILE_PERMISS...	REPLICATE_IPADDRESS	IS_LOCKED
1	kej5mq5pOE9O9QbzTcZxKw==	10.200.72.143	1002	0	2	10.200.4.151;10.200.4.151;127.0.0.1	0
2	8+0cTfINnV7eBv0pvnjCfA==	10.200.72.143	1002	0	2	10.200.4.151;10.200.70.91	0
3	RX0SkDOtDuQImlTIilbUSg==	10.200.72.143	1002	0	2	10.200.4.151;10.200.70.91;10.200.4.151	0
4	bVFvLRQaIdr4lVMiSQpAiQ==	10.200.72.143	1002	1	2	10.200.4.151;10.200.70.91;127.0.0.1	0
5	NNGYEdbhPB97kA4uBxRCJg==	10.200.72.143	1002	0	2	10.200.4.151;127.0.0.1;10.200.70.91	1
6	PZgzvvYMabh3TIp8PSurhA==	127.0.0.1	1001	0	2	10.200.70.91;10.200.4.151;10.200.4.151	0
7	tATsunKX9+bnoVdH8E6PzA==	10.200.72.143	1002	0	3	10.200.70.91;10.200.4.151;10.200.4.151	1
8	Y8vzIG3tTYQPsywnTgmWCQ==	127.0.0.1	1001	0	2	10.200.70.91;10.200.72.143;10.200.7...	0
9	gfkjFMKovQLLbq92cZoVBg==	10.200.4.151	1003	0	1	10.200.72.143;10.200.70.91	0
10	9SN3EMVhsEvz+uZOUyWDYQ==	10.200.4.151	1003	0	1	127.0.0.1;10.200.70.91	0

5 FUTURE ENHANCEMENTS

- This project can be further improved to support other types of files apart from txt files.
- We can create a proper GUI for CRUD operations.
- Automatic registration of all the files in the peer model can be implemented.

6 REFERENCES

https://en.wikipedia.org/wiki/Peer-to-peer

https://www.mirrorfly.com/blog/aes-encryption/

https://www.tutorialspoint.com/advanced-encryption-standard-aes

https://tutorialspoint.dev/computer-science/computer-network-tutorials/difference-between-aes-and-des-ciphers

https://www.simplilearn.com/tutorials/cryptography-tutorial/aes-encryption