CMSC 626 PROJECT PROPOSAL

Peer-To-Peer Distributed File System with Encryption

Department of CSEE, UMBC, FALL 2022

Group 7 (Team Name :- C - SUIT TO BE) Members:

- 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

Application:

Clients may make requests to a single processing unit (server) to access data, but they may not receive a timely response due to high traffic volume or a number of other factors. The distributed file system, which unifies several nodes (computing and storage units), is used to address this issue. Other nodes (which are a part of the distributed file system) can handle data even if a single node in a distributed system is unable to accommodate a client's request to access it.

The peer-to-peer approach is a more resilient approach which provides a number of benefits over the more widely used server-client approach, including peer supply of resources like bandwidth, storage, and computing power. File distribution is one area where resource efficiency and resilience are crucial, especially for large files. The project is an effort to implement the peer-to-peer sharing mechanism. A central node holds connection information and distributes file searches across linked peers. Peers are told if the requested file is present in any of the other peers, and they then have the option to download the file straight from the peer. New peers can be added to the network, and existing peers can unsubscribe from the network.

Objective:

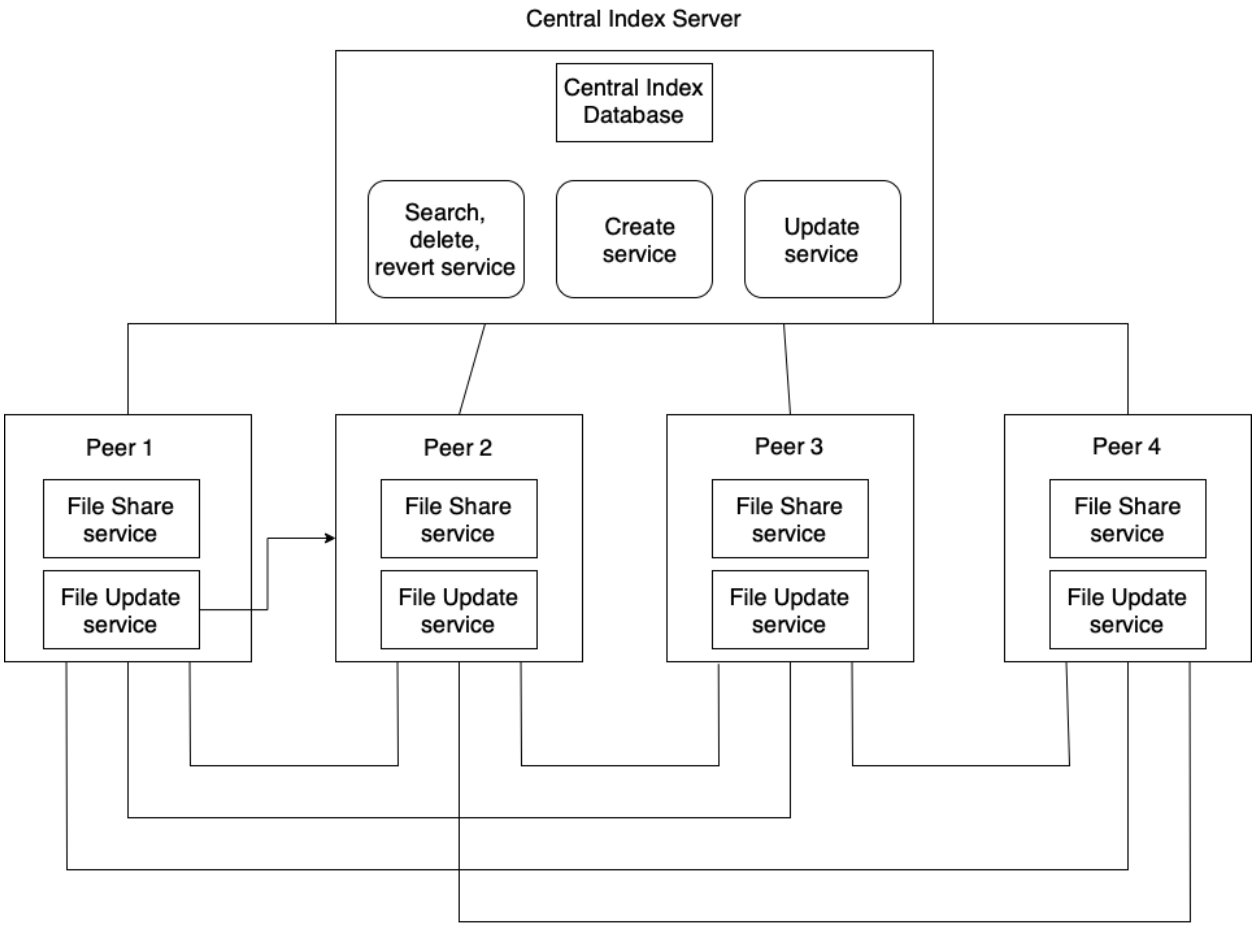
There are also some key points that the Peer-to-Peer networks should follow :

1. **Mutual Exclusion:** When two Peers in a connected network are performing any operation, then any other peer in the network should not interfere or disturb the connection or the operation. This comes under the security measurement. Other peers should not know about the type of operation or about the resources which the two peers are using.
2. **Load Balancing:** There might be the situation in the network when one type of file is asked many times or it is more in demand. Then in that case a similar file is to be transferred by the owner of the file to all the peers who are demanding. So to balance the load on a single peer, it would be a great idea to replicate this file to more than one peer, so that if any of the peers asked for the file then except the owner, other peers could also transfer the file when asked.
3. **FIFO Communication Channel:** The communication channel between the two peers can be FIFO or non-FIFO. Both the methods have their own advantages and disadvantages. But in FIFO channels, it is guaranteed that the message passed first will be delivered before the other message. It will be helpful in making the decision to take place and peers could perform the task rather than thinking about the order.
4. **Multi-Threading in Sending resources:** If more than one peer has the file which was demanded by any of the peers rather than sending the whole file by one of the peers, it could be sent by other peers who have the file in chunks means the whole file could be divided into chunks and then one part could be sent by one peer and other chunk could be sent by some other peers. Through this approach, load on the single peer is also reduced as well as operation could also be done fast.
5. **Avoiding Single Point of Failure:** By replicating the files to multiple peers, the concept of Single point Failure could also be avoided as when the file was asked, if the owner was down than another peer who has the replicated version of that file could become active and can perform the operation.

Approach:

- Initializations
    - Server code runs on a single system that is linked to a database, whereas peer code can run on multiple systems.
    - The authorized peer IPs are configured in the database on the server. The client property file also contains the server's IP address and peer ID.
    - Peer ID and IP address are used to uniquely identify the peers.
    - The secret key for encryption is configured on both the server and peer sides.
    - After starting the system, the server and client run various services listening on different ports, and the user menu is displayed on the peer end.

Architecture

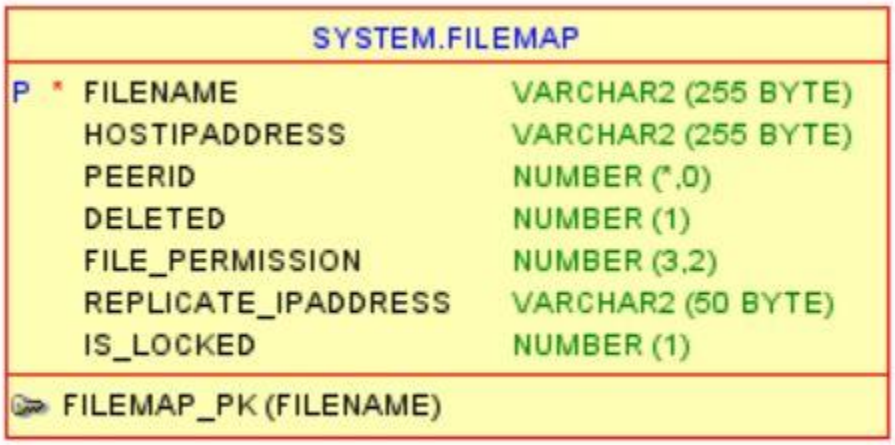


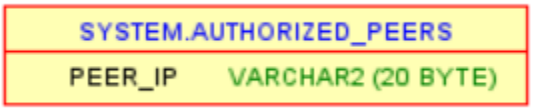
The implementation of our project resembles like a Napster P2P file implementation in which there is Central Index Server which maintains indexes of the files and the details about the peers are stores in a database with encrypted file names.All the peers connect to the the Central Index Server to get to know about details of the files and their respective storage in other peers.The peers can share and update the file from each other.The peers get their respective ids of other peers to which thry want to connect from the Central Index Server.

We are using Oracle Database to store the file names and the ip addresses of the peers the following image describes the schema of our database:

Database Schema

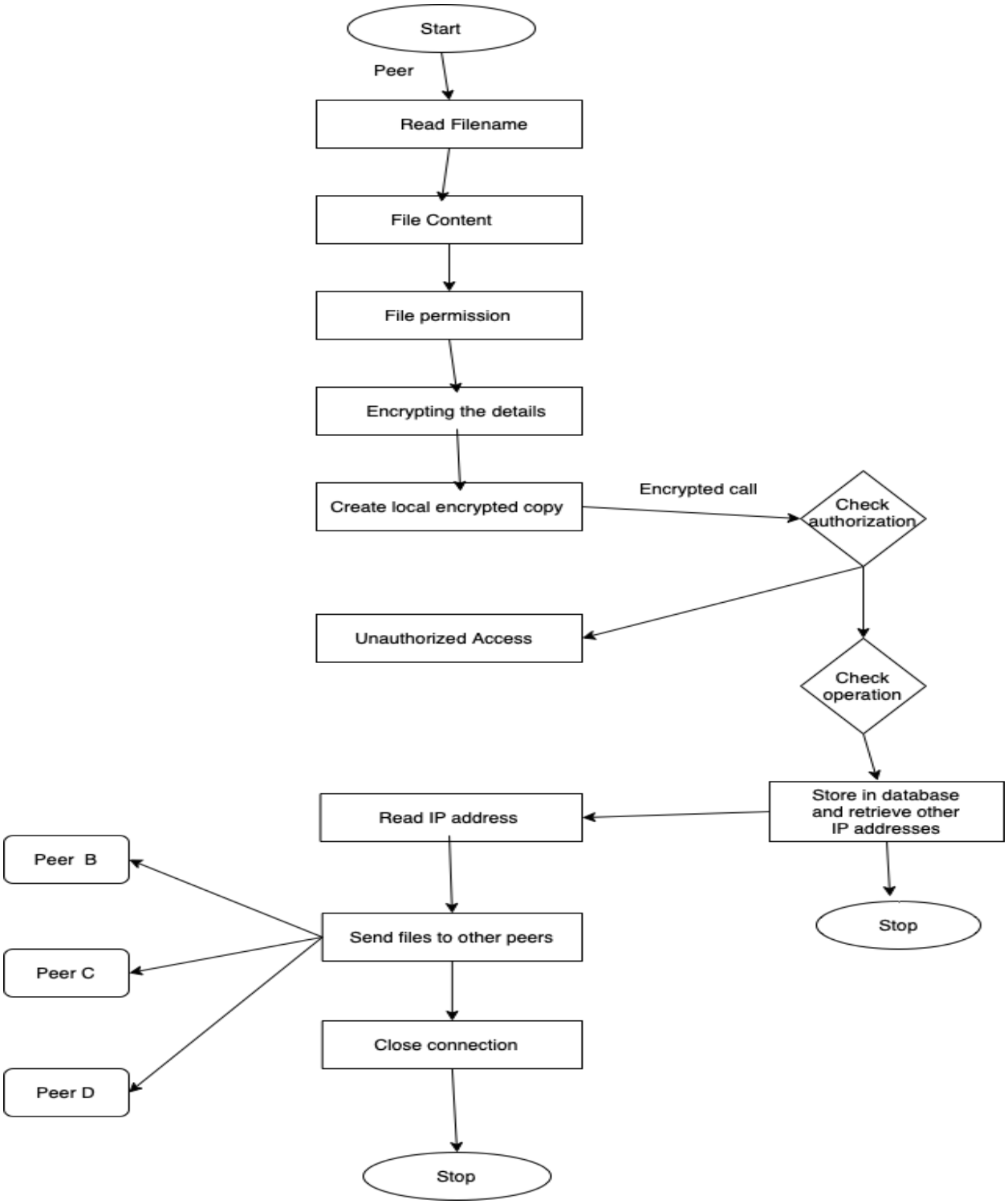
A database is used to store peer information because it is more secure, and we can retrieve existing information from the database if a server goes down or fails.





Our system implements the following functionalities at peer side:

- **File Create**



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

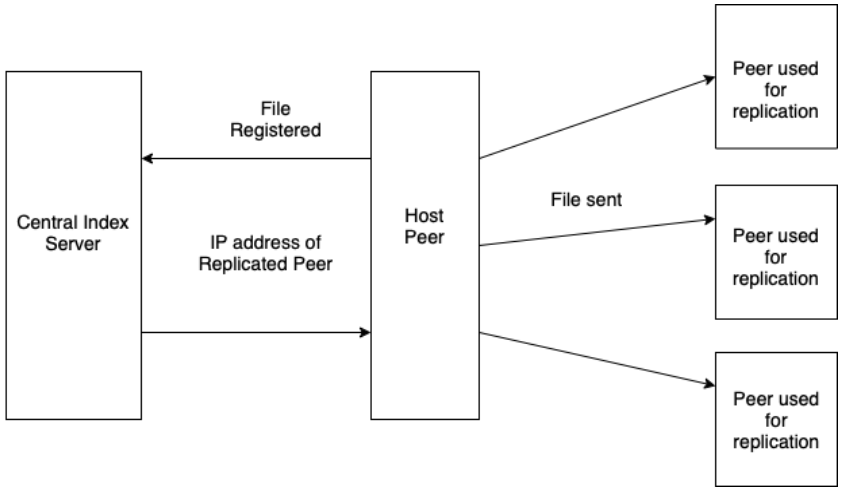
  Here the file created by the peer can only be read by the other peers.
- Read and write only

  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,replicated IP address. Peer IP and the file name acts as a primary key.



- **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. And whenever a file is being updated, it gets locked, and when another peer tries to update it can't get access as it's locked. This is how concurrency is achieved. 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.

- **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. File is Decrypted before displaying to user

- **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.

- **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.

- **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.

Implementation and functions in ClientServer/Peer:

We have 2 classes: CentralIndexServer which handles all the Server operations like querying in databases etc, The ClientServer class handles all the client operations. In the ClientServer class we have the following functions and each has the following responsibility,

1.RegisterWithIServer();

This function is used to connect the peer to the server by providing the unique peer Id. A peer at first must need to connect with the Central Index server to perform all CRUD functionalities.

2. SearchWithIServer():

This function is used in the peer to search for a file name at the Central Index server.It takes the file name as input and returns the peer the details of all the peers who have the mentioned filename.

3.createFileInSystem()

This function is used to create a file in the file system. Users can create a file by selecting the option Create New File. It prompts the server to enter the file name and asks the server to set the permissions to the file. It creates a file in the file system and maintains the record of file name in the Central Index Server.

4.updateReplicates()

In this function we create the replicates of the file.Once the Peer creates a file, we replicate the same file in atleast 3 peers.

5.updateRequest()

Using this function we can update the content i,e. write to the file in the file system. It prompts the user to enter the file name which needs to be updated and checks the permission of the file. If the peer has access to read and write to the file then we allow the peer to write to the file.

Security

- In the database, a table called "authorized_peers" is created, which contains all of the IP addresses of peers. When a new request arrives from a peer, Server Validate looks it up in the database. This prevents an unauthenticated malicious peer from connecting.
- Files and filenames are encrypted using AES initially when input is read from the user, and the encrypted format is stored in the local file system, while communicating with the server and peers, and being stored in the database. Even search operations in databases are done by encrypting filenames. Content gets decrypted only when the file is in read mode. This way, the entire system will be secure, and if others spoof into it, they will be unable to understand the content.
- File permissions assist archives in controlling file access.

Libraries Used:

- This system is developed in the Java programming language.
- Communication protocols are implemented using socket programming.
- The "AES" algorithm is used in the system for encryption.
- Oracle Database is used to securely store data.

References:

1. Awesome Peer to Peer Awesome P2P
2. P2P (Peer to Peer) File Sharing Geeks For Geeks P2P GFG
3. [1] R. Schollmeier, "A definition of peer-to-peer networking for the classification of peer-to-peer architectures and applications," Proceedings First International Conference on Peer-to-Peer Computing,Linkoping,Sweden,2001,pp.

4. [2] J. Sen, "Peer-to-peer networks," 2012 3rd National Conference on Emerging Trends and Applications in Computer Science, Shillong, 2012, pp.

5. [3] A. Hac, “A distributed algorithm for performance improvement through file replication, file migration, and process migration” , “ IEEE Transactions on Software Engineering” , 2009