Decentralized Peer To Peer Network Architecture
Assignment 1

By Bhavesh Khatri (M23cse011)
Github link [Link]

Q1. System Design (20 points):
a. Clearly define the architecture of your P2P system.
b. Describe how peers will connect, communicate, and share files.
c. Discuss how you will handle challenges like peer discovery and security.

Architecture of the P2P System :

Peer 1 and Peer 2 are two distinct roles in the Peer-to-Peer (P2P) network file sharing system. Within a local network, these roles interact in a client-server model.

Peer 1 : In a P2P network, this component serves as the server. It is in charge of connecting to a specific host and port, waiting for incoming connections, and facilitating file sharing, messaging, etc. In our case we have used Python's socket library to communicate over network.

Peer 2 : Peer 2 serves as the network's client. It connects to Peer 1 by using the host name and port specified. Peer 2 sends a file request, receives the file, and saves it locally once connected. In our case we have used Python's socket library for network communication.

Connect, Communicate, Share Files & Security.

Connection

Peer 1 : It starts by asking the user for their host name. Peer 1 provides a step-by-step guide to help users find their host address, including instructions on how to access the IPv4 address via the cmd. After that the user is asked to choose a port number that falls within the valid range of 3000 to 65535 (in our case). Peer 1 binds to the specified host and port after successful input and is ready to accept incoming connections.

Peer 2 (Client): Using socket functions, Peer 2 obtains its host name automatically. It communicates with Peer 1 via a predefined port number (8888). This connection creates a connection between the two peers.

Communication

Peer 1 : Sends a confirmation message after connecting to Peer 2 {in our case}. It then waits for a message from Peer 2 confirming that the file sharing process can begin.

Peer 2 : Sends a request message to Peer 1 in order to indicate its desire to receive a file. It then waits for Peer 1's acknowledgement.

File Sharing

Peer 1 : Peer 1 asks the user for the file path to start the file sharing process. After that, it sends the file to Peer 2.

Peer 2 : Peer 2 is waiting for the file size details after receiving the file request. It then receives the file in pieces and reconstructs it locally.

Peer Security

1. Our system is managed by a local network setting. However, encryption and authentication techniques would be essential for real-world adoption to guarantee secure file sharing. It might be investigated to implement secure communication protocols like SSL/TLS.

2. To prevent unauthorized access, additional security measures like access controls and user authentication should be implemented.

3. It is advised to conduct regular security audits and updates to address any potential vulnerabilities.

Q2. Design Implementation (30 points):
a. Implement a working P2P system.
b. Peers should be able to join the network, share files to other peers.
c. Ensure appropriate error handling and user-friendly interactions.

Implement a working P2P System :

Kindly refer to the GitHub repository for the implementation of the peer-to-peer network architecture. [Link]

For steps to run please refer to the github readme.md file. [Link]

Q3. Efficiency and Scalability (30 points):
a. Consideration of system performance as the number of peers increases.
b. Discussion of strategies to ensure efficient file discovery and download..

Efficiency and Scalability :

The current implementation works well with a small number of peers. It is critical to recognise that as the network grows, the following areas, in particular, may require attention:

1. Bandwidth Management: As the number of peers grows, network bandwidth may get congested. To improve overall network throughput, it's a good idea to look at data transmission optimisation techniques like chunking or compression.

2. Load Distribution: Distributing the load evenly among peers is necessary for scalability. Peers may be overwhelmed on some due to ineffective load distribution while being underutilized on others. To ensure equitable resource utilization across the network, load balancing algorithms and methods should be put into place.

3. Fault Tolerance: The likelihood of peer failures or disconnections rises as the network gets larger. The system ought to be built to elegantly handle these errors without degrading performance as a whole.

Strategies for Efficient File Discovery & Download:

1. Frequently requested files can be cached or copied among several peers to increase download performance. This lessens the necessity of downloading the same material from the same source again, improving overall performance.

2. Implementing parallel download capabilities enables a peer to simultaneously download various portions of a file from various sources. As a result, the download process is sped up and the workload on individual peers is lessened.

3. Implement techniques to ensure equitable bandwidth sharing amongst peers. By preventing any one peer from controlling all network resources, this encourages fair file sharing.

Q4. User Interface (10 points):
a. Create a user interface that allows users to interact with the P2P system.
b. The interface should facilitate file sharing and downloading.

User Interface : Github link [Link]

A pleasant user experience is guaranteed by the user interface's simplicity and intuitiveness. In order to help users navigate the file sharing process, it offers clear prompts and instructions. Users are required to enter important data, like the host name and port number.

Validation of Input

The programme uses regular expression pattern matching to validate the user-inputted host name to ensure accuracy. This verification process reduces mistakes and makes sure the P2P system connects to the right server.

File Sharing and Downloading :

The interface is equipped with functionalities that enable users to seamlessly share and download files between peers.

File Sharing
1. Prompt for Hostname and Port Number: Users are prompted to enter the host name, ensuring they have the correct details for a successful connection.

2. Port Number Verification: The program verifies that the chosen port number falls within the recommended range (3000 to 65535) to prevent any potential connectivity issues.

3. Binding and Connection: The P2P system establishes a connection with the specified host and port, initiating the file sharing process.

4. File Selection: Users are prompted to enter the file path or name. If the file is in the same directory, they can simply input the file name.

5. File Transfer: The selected file is sent to the connected peer, facilitating seamless sharing.

File Downloading
1. Connection to Peer 1: Peer 2 connects to Peer 1 using the provided host name and port.

2. File Request: Peer 2 sends a request message to Peer 1, indicating the desire to download a specific file.

3. File Receiving: The file is received and saved with the specified file name (coming_File1.png in this case).

Q5. Follow up Question (10 points):
a. Discuss the advantages and challenges of a decentralized P2P architecture compared to a hybrid architecture that involves some centralization.

Advantages of Decentralized P2P Architecture & Hybrid Architectures with Centralization :

Advantages of Decentralized P2P Architecture :	**Advantages of Hybrid Architectures with Centralization:**
1. Decentralized systems are easily scalable to support an increasing number of peers. The network can accommodate new peers without much reconfiguration.	1. Increased Efficiency: Centralised components can improve load balancing, resource allocation, and overall system efficiency.

2. User privacy is frequently prioritized. Data is frequently scattered, lowering the possibility of data breaches or centralized surveillance.	2. Centralized components facilitate management and monitoring of network activity and security.
3. Decentralized networks are naturally resistant to censorship because there is no central authority over them. There are no limits on the free flow of information.	3. In hybrid systems, synchronization and data consistency are easier to implement.

Challenges in Decentralized P2P Architecture & Hybrid Architectures with Centralization :

Challenges in Decentralized P2P Architecture :	**Challenges of Hybrid Architectures with Centralization :**
1. Network Overhead: As the number of peers increases, the complexity and overhead of maintaining network connections and routing information may rise.	1. Single points of failure are introduced by centralized components. The entire network can suffer if the main system fails.
2. In this, it might be difficult to ensure the security and reliability of transactions and data exchanges. Frequently, cryptographic solutions are needed.	2. When it comes to assaults or disruptions, hybrid architectures could be less resilient than totally decentralized networks.
3. Resource allocation and utilization in decentralized networks may be less effective than in centralized systems, where resources can be handled more skillfully.	3. If not adequately controlled and guarded, centralized components can present privacy hazards.