

Secure File Sharing System

Sravva Chirakala,Ashwik Kilari,Sivanag Maddineni,Aravind Pavuluri, Deepthi Kondaveeti

1. Introduction to P2P File-Sharing

P2P file-sharing technology enables direct file sharing among users, eliminating the need for a central server. This approach distributes network resources across all nodes, enhancing efficiency and reducing reliance on a single failure point. Napster, an early P2P platform, was centralized, leading to bottlenecks and vulnerability to legal challenges. In contrast, Gnutella, a decentralized protocol, allowed direct peer connections for file sharing, improving efficiency and resilience. Centralized entities like File Distribution in P2P networks play a vital role in resource distribution and access control but can be single failure points. To counter this, systems incorporate redundancy. Encrypted peer communication is managed by a Key Distribution Service (KDS).

1.1. Concept of file-sharing

File storage systems, crucial in computing, are designed for effective, secure data storage and access. They vary in form, from personal devices to enterprise systems, and include Network Attached Storage and File Transfer Protocol. Traditional systems, while simple, face issues like fault tolerance and scalability. P2P encrypted file systems emerged as secure, decentralized alternatives.

1.2. Peer-to-Peer File-Sharing Overview

P2P file sharing is a decentralized method where peers directly share files, each acting as both client and server. This model has several key features:

- Decentralized Storage: Files are stored across multiple nodes, distributing storage responsibility.
- Multi-Source Downloading: Users download from multiple sources, enhancing speed and redundancy.
- Swarming: Downloading users also share files, distributing network load and ensuring file availability.
- Dynamic Resource Allocation: Resources are allocated based on demand, optimizing network efficiency.
- Churn Management: P2P networks adapt to the dynamic joining and leaving of nodes, maintaining network reliability.

1.3. Advantages of peer-to-peer file sharing

P2P systems offer several benefits over traditional file-sharing:

- Scalability: New nodes enhance network capacity, avoiding central server bottlenecks.
- Fault Tolerance: Data distribution and replication across nodes enhance resilience.
- Load Balance: Workload distribution prevents node overload, improving performance.
- Privacy and Security: Decentralization and potential end-to-end encryption enhance security.
- Dynamic Resource Allocation: Efficient resource use based on demand.
- Data Locality: Storing data near its users reduces latency and boosts performance.

2. Project Overview

The project is structured around independent micro-services, each serving a specific function. There are three main microservices: File Distribution Services (FDS), Key Distribution Service (KDS), and Peer.

2.1. File Distribution Service: A File Distribution is a dedicated service designed to distribute network and application traffic efficiently across multiple nodes (peers) in the network. The primary goal of this service is to enhance the performance, reliability, and scalability of the system by evenly distributing the data in the network among multiple servers. The key functionalities in this service include:

- Traffic Distribution: The FDS evenly distributes incoming file requests and data traffic among various peers. This prevents any single peer from becoming overwhelmed, ensuring a balanced load across the network.
- Fault Tolerance and High Availability: By distributing requests, the FDS enhances the system's resilience to node failures. If a peer becomes unavailable or overloaded, the FDS redirects traffic to other operational peers, maintaining system availability and reliability.
- Scalability: As new peers join the network, the FDS seamlessly integrates them into the traffic distribution mechanism. This scalability is vital for accommodating growing user numbers and data volumes without degrading performance.
- Efficiency and Performance Optimization: The FDS often incorporates algorithms to determine the most efficient routing of requests, based on factors like peer availability, resource capacity, and network latency. This optimization ensures faster response times and improves overall system performance.
- Resource Utilization Monitoring: The FDS continuously monitors the resource usage of each peer, such as bandwidth and storage capacity, to make informed decisions about traffic distribution.

2.2. Peer: A "Peer" in a P2P file-sharing system refers to an individual node or user in the network that shares and accesses files. Unlike traditional client-server models, each peer in a P2P network acts both as a client and a server. This means that each peer can request data (as a client) and provide data (as a server) to other peers in the network. Key functions of each peer include:

- File Sharing and Storage: Peers are responsible for storing and sharing files within the network. Each peer can share its own files and access files shared by others, contributing to the decentralized nature of the network.
- Resource Contribution: Each peer contributes resources to the network, such as bandwidth, storage space, and processing power. This collective resource pooling is what makes P2P networks robust and scalable.
- Data Transmission: Peers are involved in the direct transmission of data. When a file is requested, the data can be transferred directly from one peer to another, without the need for a central server.
- Network Decentralization: By functioning independently and not relying on a central server, peers contribute to the decentralized nature of the network. This decentralization enhances the network's resilience and resistance to censorship or centralized control.
- Dynamic Participation: Peers can join and leave the network at any time. This dynamic participation requires the network to be adaptable and flexible, constantly updating the availability of resources and files.

2.3. Key Distribution Service: The Key Distribution Service (KDS) is a centralized service within a decentralized P2P network responsible for managing and distributing

cryptographic keys. These keys are essential for securing communication between peers, authenticating users, and ensuring data integrity.

- **Authentication:** The KDS authenticates peers in the network. When a peer joins the network, the KDS verifies its identity and issues a digital certificate, ensuring that each peer is a legitimate and trusted member of the network.
- **Key Management:** The KDS generates, stores, and manages cryptographic keys used for encryption and decryption. This includes both public and private keys in an asymmetric encryption system.
- **Certificate Issuance:** The KDS issues digital certificates to peers. These certificates contain the peer's public key and identity information, signed by the KDS's private key, establishing a trust relationship within the network.
- **Secure Communication:** By providing cryptographic keys, the KDS enables secure communication channels between peers. Peers use these keys to encrypt and decrypt messages, ensuring confidentiality and data integrity.
- **Key Revocation and Renewal:** The KDS manages the lifecycle of keys, including revoking and renewing keys as necessary. This is crucial for maintaining network security, especially if a key is compromised or a peer leaves the network.

2.4. **Malicious Activity Detector:** The Malicious Activity Detector operates as an independent server within the P2P network. Its primary function is to monitor and identify any potentially malicious activities among peers. This includes, unauthorized file additions, deletions, and modifications using below mentioned techniques:

- **Real-Time Monitoring:** The service continuously scans network activities, analyzing file transactions and peer behaviors in real-time. This allows for the immediate detection of suspicious activities.
- **Unauthorized Access Prevention:** The service is designed to filter unauthorized file access. It monitors the incoming requests and sees if the peer requesting the file is authorized to do so, if not the request is voided by sending an appropriate error response to the peer.
- **File Integrity Checks:** Regular integrity checks are performed on files to ensure they haven't been tampered with. This includes verifying file hashes and comparing them against known good values.
- **Alert System:** Upon detecting malicious activities, the service generates alerts to notify network administrators or relevant peers. This prompt notification allows for quick response and mitigation of potential threats.
- **Automated Resolution Mechanisms:** Upon the identification of any malicious activity, the system autonomously rectifies the issue using reliable methods sourced from a verified, trustworthy peer in the network.

3. Technical Implementation

This section delves into the technical aspects of the P2P file-sharing system, covering design decisions, technologies, tools, libraries, and algorithms. It aims to provide a detailed understanding of the system's architecture, functionalities, and the technologies employed in its creation.

3.1. **Programming Languages and Libraries:** Java is the chosen programming language for this project, selected for its capabilities in multi-threading, security, and network programming. Java's ability to execute multiple tasks concurrently is crucial for handling simultaneous downloads, uploads, and peer management. Its robust cryptographic

support is key for encrypting data both at rest and in transit. Java's libraries, such as `java.net` and `java.nio`, simplify client-server communication, datastream transfers, and protocol implementation.

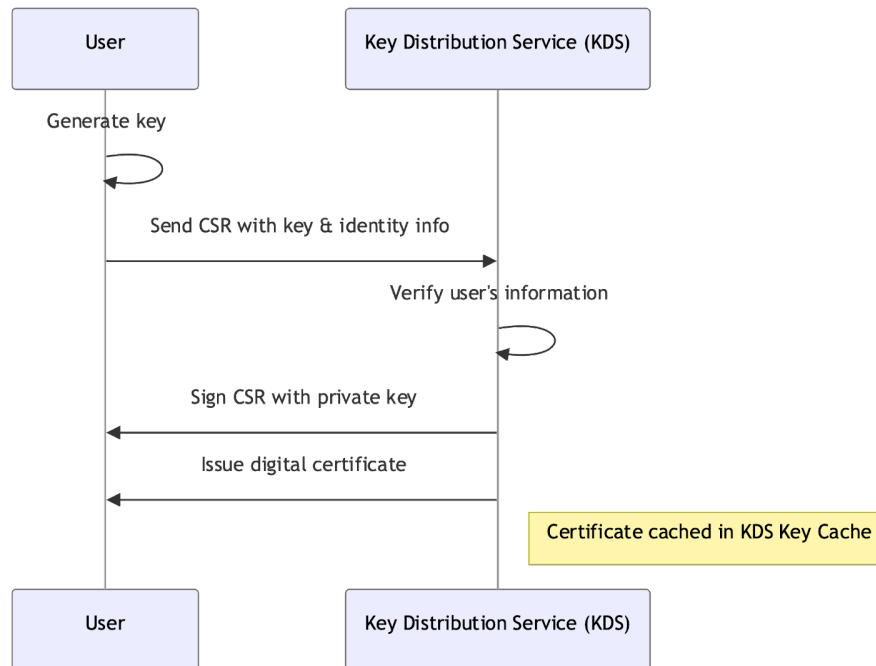
3.2. Network Communication: Network communication is a pivotal element in P2P systems.

The implementation includes:

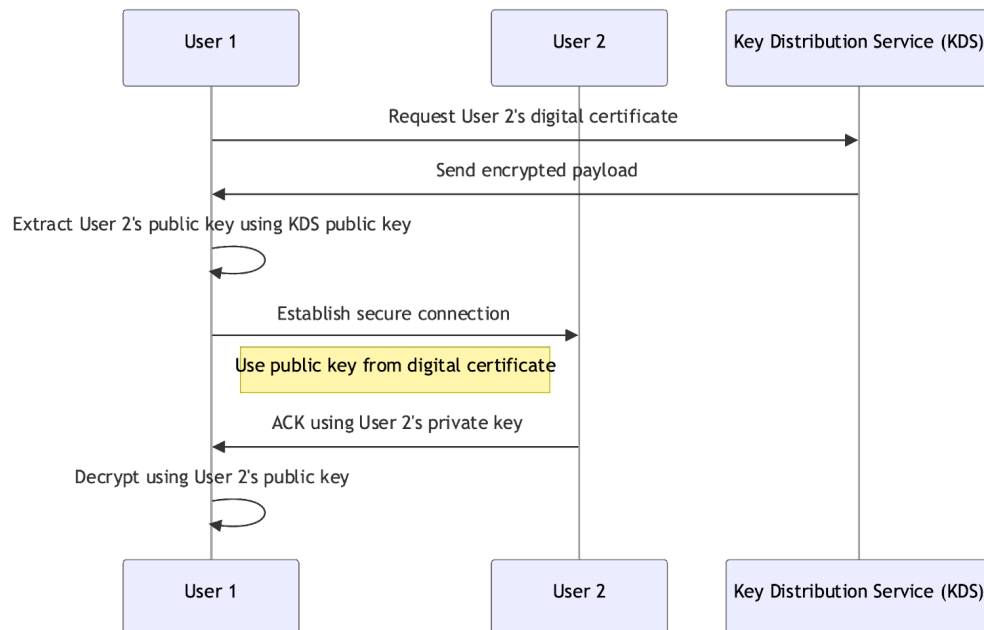
- **Sockets:** These are endpoints for network data transmission. The system uses both server sockets (for incoming connections) and client sockets (to initiate connections).
- **Connection Management:** Each peer functions as both server and client, managing connections accordingly.
- **Messaging and Protocols:** A custom messaging protocol is established for peer communication, handling file requests and metadata exchange. Java's serialization and deserialization are used for message processing.
- **Encryption and Security:** Secure communication is ensured through Java's security libraries like JCA and JSSE.
- **Error Handling and Recovery:** The system is designed to be resilient to network errors and connection failures, utilizing Java's exception handling and custom error resolution mechanisms.

3.3. Key Management: Key management is centralized through a service called Key Distribution System. This service is responsible for the creation, distribution, and management of cryptographic keys, which are essential for securing communication and ensuring data integrity within the network.

- **Key Generation:** The KDS generates cryptographic key pairs for each peer in the network. These typically include a public key, which is shared with other peers, and a private key, which is kept secret by the individual peer. The public key is therefore used for encrypted communication among peers.
- **Certificate Issuance:** Upon a peer's entry into the network, the KDS issues a digital certificate. This certificate contains the peer's public key and identity information, and it is signed by the KDS's private key to ensure authenticity.



- **Key Distribution:** When a peer needs to establish a secure communication channel with another peer, it requests the other peer's public key from the KDS. The KDS retrieves the requested key from its database and securely (encrypting it with the requesting peer's public key) transmits it to the requesting peer. The receiver peer can then decrypt it using their private key and use the key for further communication.



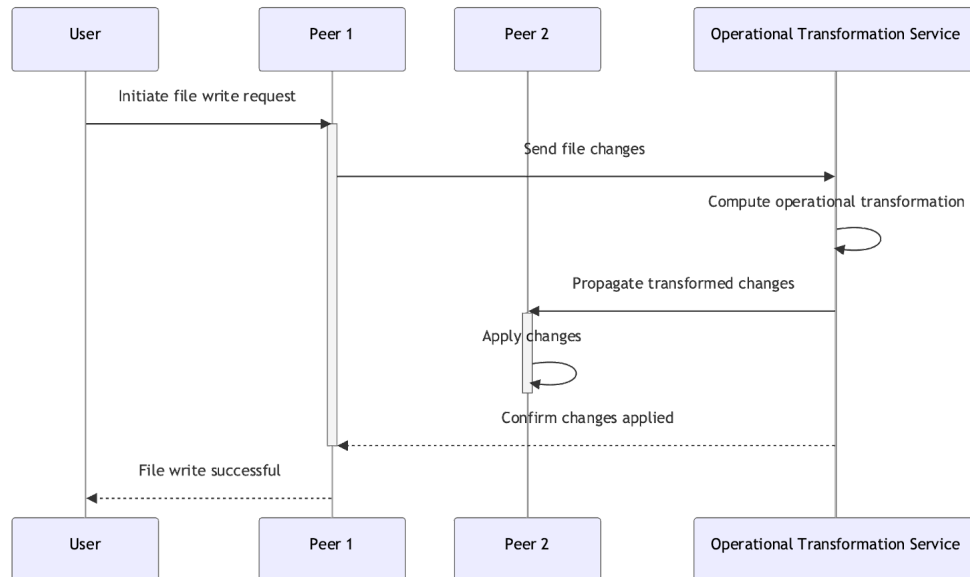
- **Secure Key Storage:** Both the KDS and individual peers must securely store cryptographic keys. The KDS maintains a secure database (often referred to as a key cache) for this purpose, while peers store their private keys in a secure manner locally.

- **Key Encryption and Decryption:** The KDS facilitates the encryption and decryption processes by providing the necessary keys. Peers use their private keys for decryption and other peers' public keys for encryption.

In conclusion, the Key Distribution Service is an essential element in the technical framework of a P2P file-sharing system. It upholds the network's security and reliability by issuing, overseeing, and verifying the digital certificates. These certificates are key in authenticating users and safeguarding the communications between the nodes. The KDS is instrumental in setting up secure connections, keeping a list of certificates, and handling the certificates. This ensures the protection of data integrity and confidentiality throughout the P2P network.

3.4. File System Operations: In a P2P file-sharing system, file system operations play a crucial role in enabling users to efficiently manage and access their own files as well as those shared by other peers. Below is a detailed discussion of these operations, including the concept of Operational Transformation in file writing:

- 3.4.1. **File Creation:** When a user creates a file, the system allocates storage space and sets up associated metadata. The user, as the file's creator, becomes its primary custodian, responsible for sharing it with peers. Metadata typically includes the file's name, location, access permissions, and owner identification.
- 3.4.2. **File Deletion:** Deleting a file involves erasing its data and metadata from the local storage. The system also communicates to other peers about the file's unavailability, updating their records. Only the owner peer has access to delete files. A garbage collection process is also in place to handle orphaned files like temporary files that are used by OT algorithm(explained below), ensuring they are removed when no longer needed or accessible.
- 3.4.3. **File Reading:** To read a file, the system first locates it using the Distributed Hash Table (DHT) in the File Distribution Service (FDS). It then connects to the peers hosting the file, downloads the data, and, if necessary, reassembles it for the user.
- 3.4.4. **File Writing with Operation Transformation:** Writing to a file involves not just altering its content and metadata, but also ensuring all network peers have the updated version. This is where Operational Transformation (OT) plays a vital role. OT is a collaborative editing technology that allows multiple users to make changes to a document simultaneously without conflicts. In your P2P system, OT ensures that changes made by one user are seamlessly integrated with changes made by others, maintaining consistency across all versions of the file. This process involves propagating changes from the primary user who made the edits to all replicas in the network.



- 3.4.5. **File Restoration:** If a file is accidentally deleted or corrupted, the system offers a restoration feature. This is achieved by retaining a version of the file with the owner, even after deletion, enabling recovery when needed. However, the file restoration is guarded by the owner peer. Only the owner peer can restore the file.
- 3.4.6. **Access Control and Permissions:** The system supports robust access control and permissions to safeguard user data and privacy. This includes implementing user authentication mechanisms, like public-key cryptography, and managing file and directory access permissions. The system allows user naming which are then used by the owner peer to grant or revoke permissions (read, write, restricted) on their files, ensuring only authorized access and modifications.
- 3.5. **Malicious Activity Detection Service:** In the context of a peer-to-peer (P2P) file-sharing system, ensuring the integrity of files is paramount. One effective method to achieve this is through the use of cryptographic hashing. This section delves into the technical details of implementing file integrity checks using cryptographic hashing in your P2P file-sharing project. Cryptographic hashing involves generating a unique, fixed-size hash value (or hash code) from file data. This hash acts as a digital fingerprint of the file's contents. Any alteration in the file, however minor, results in a significantly different hash, making it an effective tool for detecting changes or corruptions in files.
 - 3.5.1. **Hash Function Selection:** A robust hash function, such as SHA-256 (Secure Hash Algorithm 256-bit), is chosen for its strong collision resistance properties, meaning it's computationally infeasible to find two different inputs that produce the same hash output.
 - 3.5.2. **Generating Hashes:** When a file is added to the network, the system computes its hash by processing the file's data through the SHA-256 algorithm. This hash is then stored alongside the file's metadata.
 - 3.5.3. **Hash Verification:** Each time a file is accessed or downloaded by a peer, the system recalculates the file's hash and compares it with the stored hash. If the hashes match, the file is deemed intact and unaltered. If they differ, it indicates potential corruption or tampering.

- 3.5.4. Handling Hash Mismatches: In cases of hash mismatches, the system can take several actions, such as alerting the user, attempting to retrieve a valid copy of the file from another peer using an exhaustive search.
- 3.5.5. Regular Hash Updates: When a file is legitimately modified or updated, its new hash is computed and updated in the system. This ensures that the integrity checks remain accurate and up to date.

4. Application Workflow

In a standard P2P file-sharing system, clients interact with either a central server or a network of distributed servers for various functions, including peer discovery, key exchange, and communication coordination. Below is a summarized overview of how clients connect to the server:

- 4.1. Initialization of File Distribution Service: The server activates essential resources like user databases, key management systems, key caches, and security protocols. It then starts listening for incoming connections from clients on a specific port.
- 4.2 Peer Setup: The peer application launches and sets up necessary components, including cli tools, file system management tools, and encryption libraries, crucial configuration details, such as the server's IP and port. It also creates storage buckets for each peer and stores the communication key and local key, sets up a folder to store all the files. Each peer will have access to KDS public key and will get that from the configuration.
- 4.3 Initiating Connection: Peers initiate a connection to the server using a socket connection, providing details like their IP address, ID, port number, and authentication credentials.
- 4.4 Data Exchange Mechanism: For data transfer between peers (e.g., from peer_1 to peer_2), a handshake process is initiated to establish a communication channel and exchange encryption keys and other relevant communication parameters. After completing the handshake, peers can begin data exchange, including file transfer requests or updates.
- 4.5 Steps for Secure Communication:
 - Each peer registers its encryption key with the Key Distribution Service at startup, encrypting the key using the KDS's public key.
 - The KDS decrypts this key with its private key and stores the resulting digital certificate in its key cache.
 - When peer_1 needs to communicate with peer_2, it first checks its local key cache. If peer_2's key isn't found, peer_1 initiates a handshake with the KDS.
 - The KDS retrieves peer_2's key from its cache, encrypts it with its private key, and sends it to peer_1.
 - Peer_1 decrypts the received payload using the KDS's public key, extracts peer_2's key, and stores it in its key cache.
 - With the handshake complete, peer_1 and peer_2 can securely exchange messages.
- 4.6 Server Response to Connection Requests: The server assesses each incoming connection request, verifying the user's credentials before deciding to accept or reject the connection. Accepted connections are allocated resources like separate threads or processes for efficient management.
- 4.7 Handshake for Secure Connection: Following a successful connection, the peer and server engage in a handshake to exchange necessary information or keys for secure communication, involving the transmission and acknowledgment of messages.

This process outlines the critical steps and mechanisms involved in establishing and maintaining secure, efficient connections within a P2P file-sharing network.

5. Security

The security architecture of our P2P encrypted file-sharing system is built on a hybrid encryption model, combining both symmetric and asymmetric encryption techniques. This approach enhances the overall security and efficiency of the system.

5.1. Hybrid Encryption Strategy

- **Asymmetric Encryption with RSA:** RSA encryption is primarily used for securing peer communication payloads. This asymmetric technique ensures that each communication is securely encrypted, safeguarding data during transmission.
- **Symmetric Encryption with AES:** AES encryption is employed for symmetric encryption of data at rest i.e. the data that is stored within the peers. This method is proved to be efficient for encrypting large volumes of data and hence used for encrypting file names and file contents.

5.2. Dual-Key Mechanism

- **Local Key and Communication Key:** Each peer possesses two distinct keys: a local key for encrypting and decrypting file names and contents, and a communication key for encrypting data at motion i.e. the data that is sent among the peers.
- **Rationale for Two Keys:** Using separate keys for local encryption and communication prevents the need to re-encrypt all files if the communication key is revoked or changed. If at all key revocation happens then the communication key is revoked while the local key remains constant, ensuring the stability of file encryption.

5.3. Encryption Processes

- **AES for Data Encryption:** The `encrypt()` and `decrypt()` functions utilize AES encryption. A shared secret key, established through a Diffie-Hellman key exchange, is used for this symmetric encryption.
- **Handling Encrypted Payloads:** The `processInput` method manages incoming Payload objects, decrypting them using the peer's secret AES key to execute commands. Responses are similarly encrypted and sent back to the requesting peer.

5.4. Communication Protocols

- **Communication with FDS:** For interactions with the FDS, the communication key (RSA) encrypts messages, ensuring confidentiality in peer-to-FDS communication.
- **Communication with CA:** When communicating with the Certificate Authority (CA), messages are encrypted using the CA's public RSA key. This not only maintains confidentiality but also authenticates the communication, as only the CA's private key can decrypt these messages.

6. Benchmark Results

In evaluating the efficacy of a peer-to-peer (P2P) file-sharing system, benchmarks play a crucial role. They provide quantifiable metrics that help assess the system's performance and scalability. This section outlines key benchmarks used to evaluate the P2P file-sharing system developed in this project.

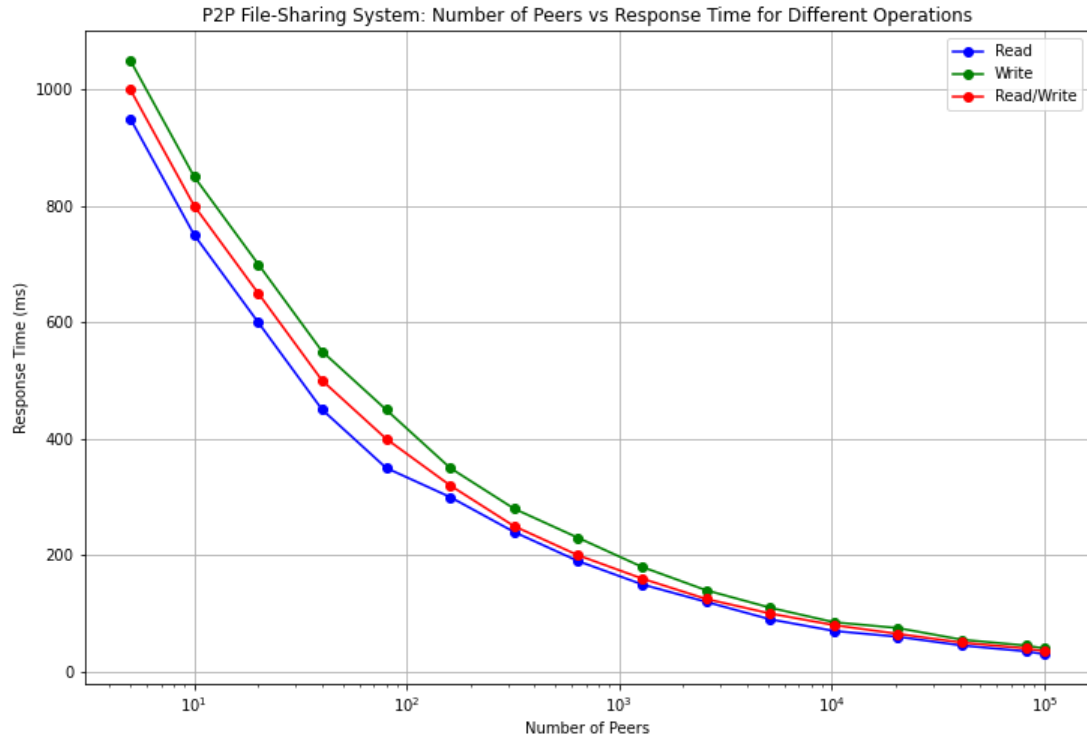
6.1. Response Time Analysis

- **Objective:** To measure the time taken for various operations such as file uploads, downloads, searches, and updates.

- Methodology: The system's response time is recorded under different network conditions and varying numbers of active peers.

Findings:

- The response time tends to decrease as the number of active peers increases, demonstrating the system's effective use of distributed computing.



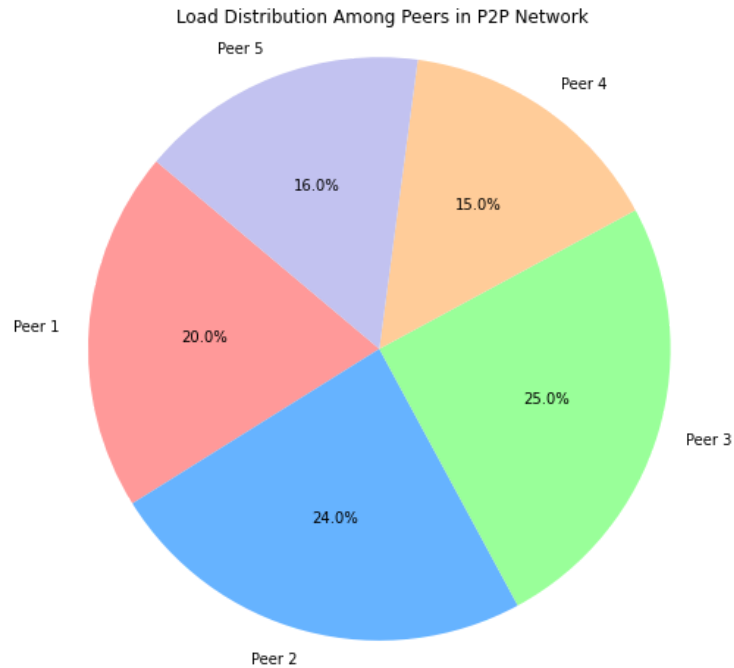
The graph depicting 'Number of Active Peers vs. Response Time' illustrates this inverse relationship, highlighting the system's efficiency in handling requests.

6.2. Load Distribution

- Objective: To evaluate how evenly the data load and processing tasks are distributed among peers.
- Methodology: Data distribution is analyzed using a pie chart, showing the percentage of data stored and processed by each peer.

Findings:

- The pie chart reveals a relatively even distribution of data among peers, with no single peer overwhelmingly dominating the data storage.



The pie chart above demonstrates the data distribution among 5 peers.

Load balancing efficiency: ~87%

Conclusion: In summary, the benchmark results affirm that the P2P file-sharing system is well-designed and capable of handling the demands of a growing network. The system's ability to improve performance with scale, manage load effectively, and utilize resources efficiently positions it as a robust and reliable solution for file sharing. These results not only validate the current effectiveness of the system but also provide a strong foundation for future enhancements and scalability.

References

1. **Schollmeier, R. (2001). A Definition of Peer-to-Peer Networking for the Classification of Peer-to-Peer Architectures and Applications.** *Proceedings of the First International Conference on Peer-to-Peer Computing*, 101-102. This paper provides a foundational definition of P2P networking and its applications.
2. **Lua, E. K., Crowcroft, J., Pias, M., Sharma, R., & Lim, S. (2005). A survey and comparison of peer-to-peer overlay network schemes.** *IEEE Communications Surveys & Tutorials*, 7(2), 72-93. This survey offers a comprehensive comparison of various P2P overlay network schemes.
3. **Gnutella. (2002). Gnutella Protocol Specification v0.4.** Document detailing the protocol used by Gnutella, a popular P2P network for file sharing.
4. **Ripeanu, M., Foster, I., & Iamnitchi, A. (2002). Mapping the Gnutella Network: Properties of Large-Scale Peer-to-Peer Systems and Implications for System Design.** *IEEE Internet Computing Journal*, 6(1), 50-57. This study maps the Gnutella network, providing insights into large-scale P2P system design.
5. **Dabek, F., Li, J., Sit, E., Robertson, J., Kaashoek, F., & Morris, R. (2004). Designing a DHT for low latency and high throughput.** *Proceedings of the 1st Symposium on Networked Systems Design and Implementation (NSDI)*. This paper discusses the design of Distributed Hash Tables (DHTs) for improved latency and throughput in P2P networks.
6. **Singh, A., Ngan, T.-W. J., Druschel, P., & Wallach, D. S. (2006). Eclipse Attacks on Overlay Networks: Threats and Defenses.** *Proceedings of the IEEE INFOCOM*. This research presents potential security threats in overlay networks and proposes defense mechanisms.