University of Science and Technology of Hanoi
Information and Communication Technology Department

DISTRIBUTED SYSTEM
MIDTERM REPORT

Peer-to-peer (P2P)
File transferring system

By Group 7

Tran Manh Duy - 22BI13126
Vu Thanh Long - BA12-110
Dinh Vu Anh - 22BI13012
Pham Ngoc Lam - 22BI13240
Le Viet An - 22BI13009

Contents

1 Introduction

1.1 Overview about Peer-to-peer networks

Peer-to-peer (P2P) architecture is a decentralized computing model where network participants share resources directly with each other without the need for a centralized server. In a P2P network, each node acts as both a client and a server, enabling distributed sharing of files, data, and computing resources. This article provides a comprehensive overview of the P2P architecture, including its characteristics, benefits, types, key components, bootstrapping process, data management, routing algorithms, challenges, security techniques, and applications.

1.2 Characteristics and types of a P2P network

1.2.1 Characteristics

- Decentralization: P2P networks operate without a central authority, allowing nodes to communicate and share resources directly.
- Scalability: Scalability: P2P networks can be easily scaled to accommodate a large number of nodes without relying on a centralized infrastructure.
- Fault tolerance and autonomy: P2P networks are resilient to node failure because the absence of a central server means that the network can continue to function even if some nodes become unavailable. In addition, each node in a P2P network has autonomy over its own resources and decisions, which contributes to the overall resilience and flexibility of the network.
- Resource Sharing: P2P network participants can share files, data, and computing resources directly with each other.

1.2.2 Types of P2P networks

1. Pure P2P Networks: Also known as decentralized or true P2P networks, pure P2P networks operate without any central authority or dedicated infrastructure.

    - Peers in these networks have equal privileges and responsibilities, and they directly communicate and share resources with each other.

    Examples: BitTorrent and Gnutella.

2. Hybrid P2P Networks: Hybrid P2P networks combine elements of both decentralized and centralized architectures.

    - They typically include some central servers or super peers that coordinate network activities, manage resources, or provide additional services. Hybrid P2P networks aim to achieve a balance between decentralization and efficiency.

Examples: Skype and eDonkey.

3. Overlay P2P Networks: Overlay P2P networks create a virtual network on top of an existing infrastructure, such as the internet.

    - Peers in these networks establish direct connections with each other, forming an overlay structure that facilitates resource sharing and communication. Overlay P2P networks often employ distributed hash tables (DHTs) or other routing mechanisms to locate and retrieve resources efficiently.

    Examples: Chord and Kademlia.

4. Structured P2P Networks: Structured P2P networks organize peers into a specific topology or structure, such as a ring, tree, or mesh.

    - Peers maintain routing tables or other data structures to facilitate efficient resource lookup and data retrieval. Structured P2P networks offer predictable performance and scalability but may require additional overhead for maintenance.

    Examples: CAN (Content Addressable Network) and Pastry.

5. Unstructured P2P Networks: In contrast to structured P2P networks, unstructured P2P networks do not impose any specific topology or organization on peers.

    - Peers in these networks typically rely on flooding or random search algorithms to locate resources, resulting in lower efficiency but greater flexibility and simplicity.

    Examples: early versions of Gnutella and Freenet.

1.3 References

- geeksforgeeks.org

2 Key Components of P2P Architecture

- **Peers (Nodes):** Each participant in the network functions as both a client and a server, contributing to resource sharing and data exchange. Nodes may have varying capabilities, ranging from personal devices to powerful servers. Dynamic peer behavior, including node churn, poses challenges in maintaining stability and consistency.

- **Overlay Network:** The logical structure connecting peers determines routing efficiency and network performance. Overlay types include ring, mesh, and hierarchical topologies. The design of the overlay network significantly impacts the network's scalability and fault tolerance. Advanced overlays integrate fault-tolerant mechanisms and self-healing properties to maintain robustness.

- **Routing and Lookup:** Mechanisms like DHTs and flooding ensure effective resource discovery and data transfer between peers. Efficient routing minimizes latency and enhances scalability. Lookup protocols must handle dynamic changes in the network, such as node churn, to maintain reliability. Techniques like proximity-aware routing further optimize network performance.

2.1 Peer-to-Peer Routing

2.1.1 Flooding (Unstructured)

Flooding is a straightforward routing method where nodes broadcast queries to all their neighbors. This approach ensures that resources are located, but it generates excessive network traffic and scales poorly.

To improve efficiency, techniques such as Time-To-Live (TTL) and query limits are often employed. These strategies reduce redundant transmissions while maintaining high availability. Flooding remains a practical approach for small networks or where query latency is not a critical concern. Recent studies suggest hybrid flooding mechanisms that combine controlled broadcast with selective query forwarding to enhance scalability.

2.1.2 DHT-Based Routing (Structured)

DHT-based routing maps keys to specific nodes using hash functions, enabling deterministic and efficient resource discovery. Systems like Chord employ circular DHTs, while Kademlia uses XOR-based distance metrics for routing.

These methods enhance scalability by ensuring logarithmic time complexity for resource lookups. However, maintaining DHT consistency requires sophisticated algorithms to handle dynamic node join/leave events. The use of replication and caching further improves the performance and fault tolerance of DHT-based systems. Innovations in decentralized load balancing and proximity-aware replication add further robustness.

2.2 Scalability and Maintenance

2.2.1 Node Joining and Leaving

Structured P2P systems dynamically adapt to changes by redistributing resources and updating routing tables. This minimizes disruption and maintains overall network efficiency. Protocols like Chord and Pastry implement mechanisms to reassign node responsibilities seamlessly during such transitions.

Unstructured systems, while less organized, handle churn rates effectively due to their redundant connections, albeit at the cost of higher overhead. Techniques like neighbor caching and backup paths enhance resilience against frequent node departures. Research into adaptive algorithms for churn management aims to further optimize these systems.

2.2.2 Replication

Replication ensures data availability and fault tolerance by duplicating resources across multiple nodes. This strategy is crucial for mitigating the impact of node failures or network disruptions.

Replication policies, such as full and partial replication, balance storage costs and performance based on application requirements. Additionally, adaptive replication schemes respond to network conditions, ensuring optimal resource distribution and availability. Real-time replication techniques enhance the reliability of mission-critical applications.

3 File Transfer Process

3.1 File Transfer Process in P2P Networks

File transfer in P2P networks involves the direct exchange of files between peers without intermediary servers. This process is often faster and more efficient than traditional file-sharing methods. Below is a detailed explanation of the file transfer process in a P2P network.

3.1.1 Step-by-Step File Transfer Process

1. **Peer Discovery:** A peer seeking a file must first discover other peers in the network. This can be achieved through methods such as:

   - Bootstrapping servers: Temporary servers that provide a list of active peers.
   - Broadcasting: Sending a discovery request to all peers in the network.
   - Distributed Hash Tables (DHTs): A decentralized mechanism to locate peers with the required file.

2. **File Search:** The requesting peer sends a query to the network to locate the desired file. Depending on the network type, the query can be flooded to all peers or targeted using DHTs.

3. **Connection Establishment:** Once the file's location is identified, the requesting peer establishes a direct connection with the peer(s) hosting the file.

4. **File Download:** The file is downloaded using a transfer protocol such as:

   - TCP (Transmission Control Protocol): Ensures reliable and complete file transfer.
   - UDP (User Datagram Protocol): Faster but less reliable, suitable for real-time applications.

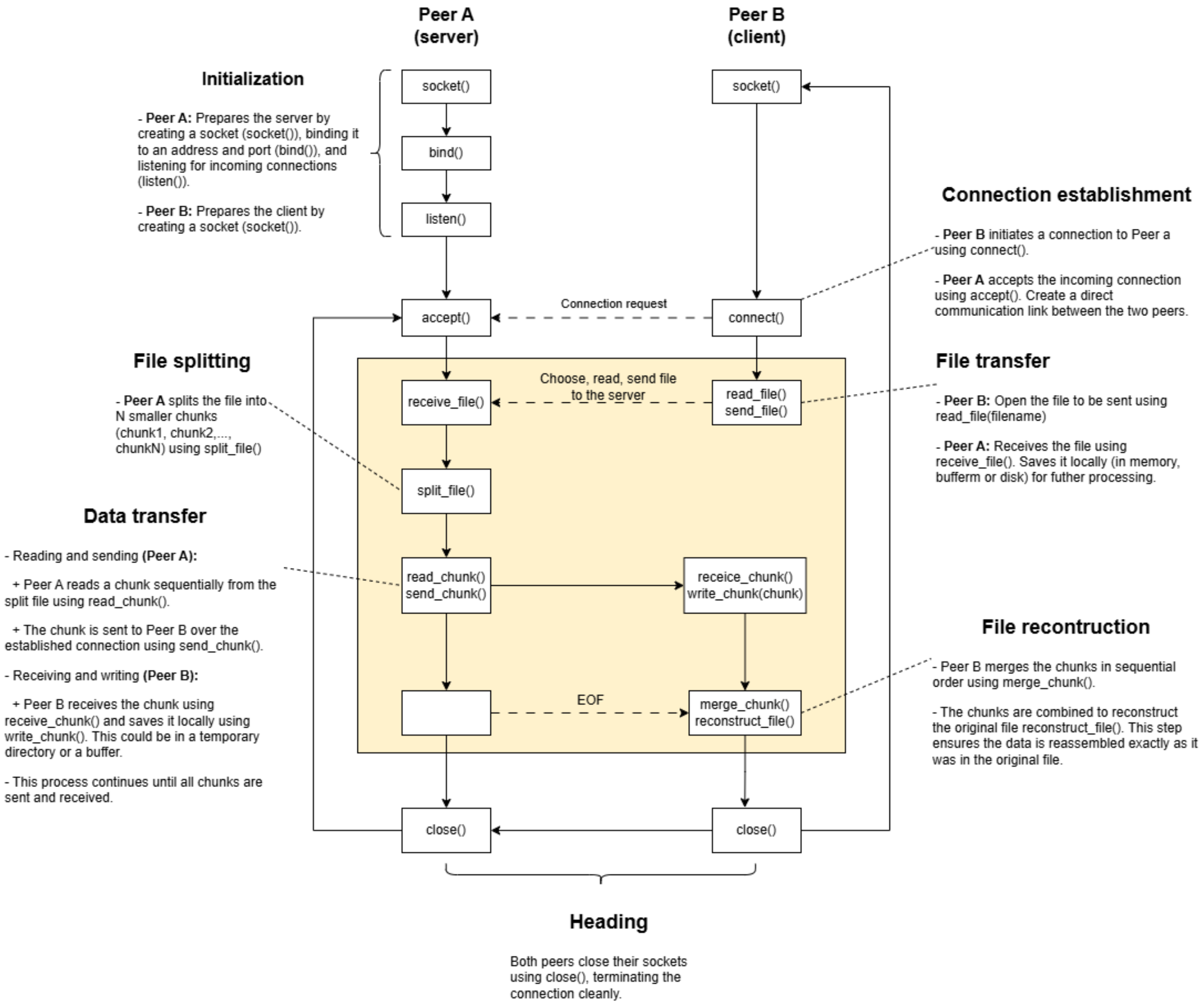
   Files may be:

   - Downloaded entirely from a single peer.
   - Split into smaller chunks and downloaded from multiple peers simultaneously (e.g., BitTorrent protocol).

5. **Data Verification:** Each downloaded file or chunk is verified using hash functions to ensure data integrity. If a chunk is corrupted, it is re-requested from the network.

6. **Sharing (Seeding):** After downloading, the peer becomes a seeder, sharing the file or chunks with other peers requesting it.

7. **CODE:**



Initialization

- **Peer A:** Prepares the server by creating a socket (socket()), binding it to an address and port (bind()), and listening for incoming connections (listen()).

- **Peer B:** Prepares the client by creating a socket (socket()).

Connection establishment

- **Peer B** initiates a connection to Peer a using connect().

- **Peer A** accepts the incoming connection using accept(). Create a direct communication link between the two peers.

File splitting

- **Peer A** splits the file into N smaller chunks (chunk1, chunk2,..., chunkN) using split_file()

File transfer

- **Peer B:** Open the file to be sent using read_file(filename)

- **Peer A:** Receives the file using receive_file(). Saves it locally (in memory, bufferm or disk) for futher processing.

Data transfer

- Reading and sending **(Peer A):**

  + Peer A reads a chunk sequentially from the split file using read_chunk().

  + The chunk is sent to Peer B over the established connection using send_chunk().

- Receiving and writing **(Peer B):**

  + Peer B receives the chunk using receive_chunk() and saves it locally using write_chunk(). This could be in a temporary directory or a buffer.

- This process continues until all chunks are sent and received.

File recontruction

- Peer B merges the chunks in sequential order using merge_chunk().

- The chunks are combined to reconstruct the original file reconstruct_file(). This step ensures the data is reassembled exactly as it was in the original file.

Heading

Both peers close their sockets using close(), terminating the connection cleanly.

Figure 1: Illustration of Peer-to-Peer Network File Transfer Process

4 Security Challenges

4.1 Overview

Peer-to-peer (P2P) file sharing has revolutionized the way data is distributed by enabling direct file transfers between users without reliance on central servers. However, the decentralized nature of P2P networks also presents significant security risks and operational challenges.

Security concerns include data corruption, malware distribution, and privacy risks due to unencrypted exchanges. Operationally, P2P networks face bandwidth strain, network instability, and difficulties in scaling effectively. They are also often associated with legal and ethical issues like copyright infringement.

4.2 Security Issues in Peer-To-Peer File Sharing

- **Malware, Spyware and Viruses**: Since P2P networks allow direct sharing between users, files shared within the network may contain malware or viruses. These types of malicious software can cause all sorts of havoc, including bringing down your entire network.
- **Data Integrity and Corruption**: Without a central authority to validate files, there is a risk of data corruption or unauthorized alterations. Users may unknowingly download corrupted files or altered versions of original files.
- **Privacy Risks**: Many P2P systems lack encryption for the files being exchanged. This exposes sensitive data to potential eavesdropping or interception. Users may unknowingly share personal or confidential information if the network is not adequately protected.
- **Insecurity File Transfer**: In some cases, P2P users may not be fully anonymous. IP addresses and activity logs can be traced back to users, leading to privacy breaches and potential legal consequences.

4.3 Operational Challenges

- **Network Instability**: P2P networks often rely on user nodes that may join or leave at any time, causing fluctuations in the network's availability and reliability. This can affect the quality and speed of file sharing.
- **Scalability**: As the network grows in size, managing the increasing number of peers and ensuring efficient file distribution becomes more difficult. P2P systems may face issues in scaling efficiently, leading to slower file sharing and connectivity problems.
- **Bandwidth Strain**: P2P networks often require users to share their internet bandwidth, which can lead to congestion and slow speeds, especially for users with limited bandwidth. This could affect the overall user experience.
- **Legal and Ethical Issues**: P2P networks are often used for sharing copyrighted content without authorization, leading to legal challenges. Copyright infringement lawsuits and ethical debates about file sharing in the context of intellectual property can complicate the use of P2P systems.

4.4 Solutions and Mitigations

- **Encryption and Secure Protocols**: Implementing encryption at both the file and transport levels can mitigate the risks of data interception and malware. Protocols like TLS or SSL can be used to ensure the privacy and integrity of file transfers.

- **Authentication and Trust Models**: Introducing methods of verifying the identity of peers, such as digital signatures or reputation systems, can help prevent malicious actors from exploiting the network. Users can be rated or verified, increasing trust in the file-sharing process.

- **Legal Compliance and Content Filtering**: To avoid legal issues, some P2P networks implement content filtering and compliance with copyright laws, preventing the sharing of pirated content. This approach can help protect users from lawsuits and reduce the legal risk associated with file sharing.

- **Improved Network Protocols**: Enhancing network protocols to handle dynamic nodes more effectively and ensure more reliable peer discovery and file retrieval is crucial for overcoming scalability and stability challenges.

5 Application of P2P

5.1 Advantages and disadvantages of P2P

Advantages

- P2P networks do not require the use of servers.
- Each computer device is a separate user manager.
- P2P operations do not require any complex specialized knowledge.
- Home and small business environments are suitable for using P2P networks.
- Does not require too much traffic when accessing the network.

Disadvantages

- Information on the device cannot be backed up centrally.
- If multiple computer devices access at the same time, it will reduce performance.
- Data sets are not scientifically arranged but are stored on personal computers. This significantly affects the process of determining their location.
- Only providing some basic rights, poor security.

5.2 Real-World Applications of P2P

1. File sharing: P2P is widely used to share files such as music files, videos, etc. The most typical application is Spotify's application so that users can search and save their favorite music. In more detail, Spotify has applied P2P network structure.

2. Cryptocurrency: The basis of prominent cryptocurrencies such as Bitcoin is blockchain technology. Transactions all use the P2P network to authenticate transactions and maintain some decentralization. P2P technology ensures equality and transparency in financial transactions.

3. Internet of Things (IoT): P2P is a technology that communicates directly between devices without relying on a central server. Therefore, P2P allows for faster data transmission and reduced latency in IoT applications.

5.3 Future Applications of P2P

1. Data sharing and privacy: P2P will improve and develop more in terms of security, while strengthening data control. This helps individuals manage their assets more effectively.

2. Edge computing: P2P has the potential in edge computing, allowing devices at the edge of the network to communicate and collaborate effectively. P2P technology can improve the performance and scalability of edge computing applications by distributing computing resources and data processing.

3. Collaborative Workspace: P2P technology can create a private network within the workspace by allowing secure and efficient resource sharing among distributed teams. Create a secure and highly productive collaborative workspace.

6 References

- geeksforgeeks.org
- Napster: The First Peer-to-Peer File Sharing Service.
- Gnutella Protocol Specification.
- chord Stoica, I., Morris, R., Karger, D., Kaashoek, M. F., & Balakrishnan, H. (2001). Chord: A scalable peer-to-peer lookup protocol for internet applications. Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications.
- IPFS: The InterPlanetary File System.
- Nakamoto, S. (2008). Bitcoin: A Peer-to-Peer Electronic Cash System.
- WebRTC: Enabling Real-Time Communications in Browsers.
- P2P Networks and Blockchain: Overview and Innovations.
- Brosix: Peer-to-Peer File Transfer
- GeeksforGeeks: P2P File Sharing

- Raysync: Point-to-Point Transfer
- Andrew S. Tanenbaum, ”Computer Networks,” 5th Edition.
- BitTorrent Protocol Specification.
- Peer-to-Peer Network Design and Applications.
- Caplinked: How save is Peer-To-Peer File Sharing
- CUHK-Information Technology Services Centre: Risk in Peer-To-Peer File Sharing
- SecureDocs: Peer-To-Peer (P2P) File Sharing Risks