Architecture of Peer-to-Peer Systems

Tran Manh Duy

December 23, 2024

Contents

Abstract

Peer-to-Peer (P2P) architectures enable decentralized resource sharing and file transfer directly between nodes without the need for central authority. This report analyzes the P2P system architecture, highlighting types, components, routing mechanisms, scalability, and applications.

1 Introduction

Peer-to-Peer (P2P) systems are a fundamental aspect of distributed computing. These systems facilitate data sharing and task distribution among interconnected nodes, eliminating reliance on a centralized server. This report explores their architecture, advantages, and practical uses.

2 Types of P2P Architectures

2.1 Centralized P2P

Centralized P2P systems rely on a central server to manage an index of files and peers. Users query the server to locate resources and connect directly for data transfer. Examples include Napster. While efficient for file discovery, centralized systems are prone to single points of failure and legal risks.

2.2 Decentralized Unstructured P2P

Unstructured P2P lacks a defined topology. Nodes locate files using query flooding, broadcasting requests to neighbors. Examples include Gnutella and Kazaa. This model is easy to implement and fault-tolerant but generates significant network traffic and lacks scalability.

2.3 Decentralized Structured P2P

Structured P2P employs systematic organization, typically using Distributed Hash Tables (DHTs). Nodes and data are indexed to enable efficient retrieval. Examples include Chord and Kademlia. These systems offer scalability and efficient file location but are complex to implement.

3 Key Components of P2P Architecture

- **Peers (Nodes)**: Participants in the network acting as both clients and servers.
- **Overlay Network**: Logical interconnections among peers.
- **Routing and Lookup**: Mechanisms for resource discovery, such as DHTs.

4 Peer-to-Peer Routing

4.1 Flooding (Unstructured)

Flooding involves broadcasting queries to all neighboring nodes. While suitable for smaller networks, it becomes inefficient at larger scales.

4.2 DHT-Based Routing (Structured)

Structured routing maps keys to specific nodes using hash functions. Examples include:

- **Chord**: Employs a circular DHT structure for efficient message routing.
- **Kademlia**: Utilizes XOR metrics for effective search.

5 Scalability and Maintenance

5.1 Node Joining and Leaving

New nodes integrate by updating routing information, while departing nodes redistribute their responsibilities. Structured systems manage these changes effectively.

5.2 Replication

Data replication enhances availability and fault tolerance across nodes.

6 Hybrid Architectures

Hybrid systems combine client-server and P2P features. Central servers may handle metadata while file transfers occur between peers. An example is BitTorrent, which employs tracker servers but also supports DHT-based operations.

7 Comparison of Architectures

Aspect	Centralized	Unstructured	Structured
Search Speed	Fast	Slow	Fast
Scalability	Limited	High	Very High
Fault Tolerance	Low	High	High
Implementation	Simple	Simple	Complex
Overhead	Low	High	Moderate

8 Real-World Applications of P2P Architectures

8.1 Centralized

Early platforms like Napster for music sharing.

8.2 Unstructured

Networks like Kazaa and LimeWire for file sharing.

8.3 Structured

- **Distributed Storage Systems**: Platforms like IPFS utilize structured P2P systems to offer decentralized and efficient storage solutions.
- **Blockchain Networks**: Blockchain platforms such as Ethereum rely on structured P2P networks to maintain ledger consistency and enable smart contracts.
- **Content Delivery Networks (CDNs)**: Peer-assisted CDNs, such as those used in streaming platforms, utilize P2P technologies to reduce latency and server loads.
- **Online Gaming**: Multiplayer online games employ P2P systems for real-time communication and resource sharing among players.
- **File Sharing Protocols**: Applications like BitTorrent revolutionized file sharing by combining structured overlays with unstructured features for efficiency.

9 Conclusion

P2P architectures are crucial for enabling decentralized, scalable, and fault-tolerant systems. By addressing inefficiency and scalability issues, structured and hybrid models pave the way for versatile applications in modern computing.

References

[1] "Napster and the history of file-sharing," Wired, https://www.wired.com/.

[2] "How Gnutella Works," HowStuffWorks, https://computer.howstuffworks.com/gnutella.htm.

[3] Chord: "A Scalable Peer-to-Peer Lookup Service for Internet Applications," https://pdos.csail.mit.edu/papers/chord:sigcomm01/chord_sigcomm.pdf.

[4] Kademlia: "A Peer-to-Peer Information System Based on the XOR Metric," https://link.springer.com/.

[5] "Understanding BitTorrent," Ars Technica, https://arstechnica.com/.

[6] "InterPlanetary File System (IPFS)," https://ipfs.io/.

[7] "Ethereum Whitepaper," Ethereum, https://ethereum.org/en/whitepaper/.

[8] "Peer-assisted CDNs and Streaming," Cloudflare, https://www.cloudflare.com/learning/cdn/what-is-a-cdn/.