Distributed Systems - Course Project Proposal
Napster with Replication (Project 31)

Vysishtya Karanam - 2022102044
Renu Sree Vyshnavi - 2022101035
Vivek Hruday Kavuri - 2022114012

1. Problem Statement

Traditional Napster's centralized index introduces critical challenges in distributed environments. The single-point-of-failure problem makes the system vulnerable to server crashes, while the absence of replication leads to poor fault tolerance when peers leave or disconnect. Furthermore, as the system scales, uneven file availability can result in longer download times and failed lookups.

To address these issues, our project introduces replication at the data level. File chunks stored on peers are replicated across several nodes according to configurable replication factors and availability criteria. This ensures that file access remains possible even when some peers fail or disconnect, improving the system's overall reliability and resilience.

2. Project Scope

This project aims to design and implement a distributed file sharing system based on the classical Napster architecture, extended with a data replication mechanism for improved availability, fault tolerance, and load distribution. Napster represents one of the earliest peer-to-peer (P2P) systems, relying on a centralized directory server to maintain metadata about files shared by participating peers. Our system seeks to replicate shared data across multiple peers to mitigate the effects of node failures, handle churn gracefully, and ensure data availability even when original sources go offline.

3. Features We Will Implement

1. **Central Directory Server (Broker):** A metadata server that maintains mappings between files and peers. It handles peer registration, search queries, and lookup operations. For this project, a single central broker will be used.
2. **Peer Nodes:** Peers connect to the broker, advertise available files, and directly exchange data with other peers without broker involvement during transfer.
3. **File Chunking and Replication:** Each file is divided into fixed-size chunks and replicated across multiple peers based on a fixed replication factor to improve availability.
4. **Replication Strategy:** Static replication policy ensuring that each file chunk has a minimum number of replicas across distinct peers.
5. **Consistency Handling:** Basic consistency through version control or timestamp-based updates between replicas.
6. **Communication Protocols:** TCP-based client–server and peer–peer communication using Go's `net` and `encoding/json` packages for message exchange and control flow.

How the Scope Will Be Realized

Central Directory Server (Broker):

- Maintains peer registration, active peer lists, and file-to-peer mappings.
- Handles lookup requests and responds with lists of peers that hold requested files.
- Does not participate in or monitor actual file transfers.

Peers:

- Advertise locally available files to the broker upon connection.
- Handle upload and download requests directly with other peers.

- Maintain replicas of assigned file chunks and synchronize versions periodically.
- Send heartbeat messages to the broker to indicate liveness.

Replication Layer:

- Ensures data availability despite peer churn through fixed replication factors.
- Detects peer failures via timeouts and triggers re-replication when necessary.
- Manages consistency through version-based synchronization.

Advanced features like dynamic replication or distributed metadata servers will be considered optional extensions depending on time and feasibility.

4. Technology Stack and Testing:

- **Language:** Go (Golang) for concurrency and networking.
- **Storage:** Local file system for chunks; lightweight metadata persisted as JSON or SQLite.
- **Networking:** Built-in `net`, `net/http`, and `encoding/json` packages for implementing the server-client communication and peer-to-peer file transfer protocols.
- **Visualization & Metrics:** Python (Matplotlib) and Go-based CSV export utilities for analyzing lookup latency, fault tolerance, and availability metrics.

5. Evaluation and Metrics

We will evaluate our system on the following metrics:

- **Lookup Latency:** Average time to locate and begin downloading a file.
- **Availability:** Fraction of successful file retrievals during simulated peer failures.
- **Replication Overhead:** Additional storage and bandwidth consumption caused by replication.
- **Scalability:** System behavior as the number of peers and files increases.

6. Timeline

- **Phase 1 - Planning and Research:** Study Napster architecture, analyze replication strategies, and design system architecture.
- **Phase 2 - Core System Development:** Implement base Napster model with central directory and peer-to-peer file transfers.
- **Phase 3 - Replication Integration:** Extend peers to support data replication, replica selection, and consistency updates.
- **Phase 4 - Testing and Evaluation:** Conduct fault-tolerance and scalability experiments under various churn scenarios.
- **Phase 5 - Finalization and Reporting:** Optimize performance, generate result plots, and finalize project documentation.

We plan to complete up to Phase 2 (full Napster system implementation without replication) by the mid evaluation.

7. Expected Outcome

The enhanced Napster system is expected to demonstrate improved fault tolerance, high data availability, and robustness against peer churn, while maintaining manageable replication overhead. The project will offer insights into practical replication mechanisms within hybrid P2P systems.

8. References

1. P2P Networks for Content Sharing, 2004. Available at: `https://arxiv.org/pdf/cs.DC/0402018`
2. Napster Architecture Overview, *Lecture Slides*.