Verification Document for P2P File Sharing System

1. Introduction

This document outlines tests conducted to verify the functionality, performance, and stability of the P2P file-sharing system. Tests aim to ensure core functionalities such as file registration, search, download, deregistration, fault recovery, and concurrency handling. Known limitations are also described.

2. Test Environment

- **Operating System:** Windows 10
- **Programming Language:** Go (Golang 1.19+)
- **System Resources:** 8 GB RAM, 50 MB disk space for testing
- **Testing Peers:** Super-Peers (SP1, SP2) and Leaf-Nodes (LN1, LN2, LN3)
- **Port Configurations:**
  - Super-Peers: Ports 8000-8009
  - Leaf-Nodes: Ports 9000-9009

3. Test Cases

Test Case 1: File Registration

- **Objective:** Verify that leaf nodes can register files with the super-peer.
- **Steps:**
  1. Start super-peer (SP1) and leaf nodes (LN1, LN2).
  2. Register files in each leaf node's shared directory.
- **Expected Output:** Each file should show a successful registration message on SP1.
- **Status:** ✅ Passed

Test Case 2: File Search

- **Objective:** Confirm search functionality locates the correct peers holding the specified file.
- **Steps:**
    1. Start SP1 and LN1.
    2. Perform a search from LN1 for "sample_file.txt."
- **Expected Output:** Super-peer should return a list of nodes that contain "sample_file.txt," including LN2 if it holds the file.
- **Status:** ✅ Passed

Test Case 3: File Download

- **Objective:** Verify that files can be downloaded from one peer to another.
- **Steps:**
    1. Search for "document.pdf" from LN3.
    2. Initiate a download of the file from LN2 to LN3.
- **Expected Output:** File "document.pdf" should be downloaded into LN3's directory.
- **Status:** ✅ Passed

Test Case 4: File Deregistration

- **Objective:** Confirm that files can be deregistered when removed or modified.
- **Steps:**
    1. Remove "obsolete_file.txt" from LN1's shared directory.
    2. Notify SP1 for deregistration.
- **Expected Output:** SP1 removes "obsolete_file.txt" from its index.
- **Status:** ✅ Passed

Test Case 5: Fault Recovery

- **Objective:** Verify system's ability to handle unexpected disconnections and failover.
- **Steps:**
    1. Disconnect SP1 during an active query process.
    2. Reroute queries to an alternate super-peer (SP2).
- **Expected Output:** System reroutes requests and maintains operations.
- **Status:** ✅ Passed

Test Case 6: Concurrency Testing

- **Objective:** Assess system performance with concurrent file downloads and queries.
- **Steps:**
    1. Simulate 50 concurrent queries for "shared_resource.pdf" from LN1, LN2, and LN3.
    2. Repeat with 100 concurrent downloads from multiple nodes.
- **Expected Output:** System handles concurrency, with logged response times for analysis.
- **Status:** ✅ Passed

Test Case 7: Performance Testing (Response Times)

- **Objective:** Measure average response time under sequential and concurrent requests.
- **Steps:**
    1. Run 200 sequential file queries.
    2. Run 50 concurrent queries from various leaf nodes.
- **Expected Output:** Average response times are within acceptable limits under both conditions.
- **Status:** ✅ Passed

Test Case 8: Security and Unauthorized Access

- **Objective:** Ensure only authorized nodes can register, query, and download files.
- **Steps:**
    1. Attempt file registration from an unauthorized node.
    2. Perform an unauthorized search attempt.
- **Expected Output:** Unauthorized actions are denied with an appropriate error message.
- **Status:** ✅ Passed

4. Known Issues

1. **Case Sensitivity in Search:** File searches are case-sensitive, which may lead to missed results if filename cases differ between nodes.
2. **Concurrency Limitations:** Performance degrades with more than 50 concurrent downloads or queries, suggesting potential optimization in network handling.
3. **Directory Watcher Delay:** Detection of file changes in directories has a delay due to polling. Reducing the polling interval may improve responsiveness.
4. **Log File Locking Issues:** Log files may experience access conflicts during high-concurrency logging events. Implementing asynchronous logging could resolve this.