Test Report: Verification of the Program

1. Overview
This document describes the tests performed on the hierarchical P2P system using sockets. The goal is to verify the correctness of the system and identify any edge cases where the system might not work as expected.

---

2. Test Setup
1. Configuration Files:
   - system_config.txt:

     PUSH_ENABLED=true
     PULL_ENABLED=true
     TTR=30000
   - network_config.txt:

     super-peer1: super-peer2,super-peer3
     super-peer2: super-peer1,super-peer4
     super-peer3-leaves: leaf1,leaf2
     super-peer4-leaves: leaf3

2. Files:
   - shared/leaf1: Contains file1.txt, file2.txt.
   - shared/leaf2: Contains file3.txt, file4.txt.
   - shared/leaf3: Contains file5.txt.

3. Testing Environment:
   - Java Version: OpenJDK 17.
   - OS: Ubuntu 22.04.
   - All super-peers and leaf nodes are executed on a single machine.

---

3. Tests Performed
Test 1: Push-Based Invalidation
- Objective: Verify that file invalidations are broadcasted and applied correctly.
- Steps:
  1. Start the system with PUSH_ENABLED=true.
  2. Simulate file modifications in leaf1.
  3. Monitor logs for invalidation broadcasts and updates on other nodes.

- Expected Output:

leaf1: Simulated modification for file1.txt
leaf1: Broadcast invalidation for file1.txt
super-peer1: Invalidated file1.txt for leaf2

- Result:
  Correct invalidation broadcast and file invalidation at other nodes.

Test 2: Pull-Based Polling
- Objective: Verify that leaf nodes poll for updates and refresh cached files correctly.
- Steps:
  1. Start the system with PULL_ENABLED=true and TTR=30000.
  2. Simulate queries and downloads from leaf2.
  3. Modify a master file in leaf1 and observe polling results at leaf2.

- Expected Output:

leaf2: Polling for file1.txt at super-peer1
leaf2: Response from server: VALID:file1.txt:30000
leaf2: File file1.txt remains valid. New TTR: 30000ms

- Result:
  Successful polling and updates based on origin server responses.

Test 3: Query Handling
- Objective: Ensure that queries and query responses work correctly.
- Steps:
  1. Start the system.
  2. Simulate a query from leaf2 for file1.txt.
  3. Observe the QUERY and QUERYHIT messages.

- Expected Output:

leaf2: Sent QUERY for file1.txt
super-peer1: Received QUERY for file1.txt
leaf2: Received QUERYHIT for file1.txt

- Result:
  Queries were propagated, and correct responses were received.

Test 4: Testing with Different TTR Values
- Objective: Evaluate system behavior with different TTR values (30s, 60s, 120s).
- Steps:
  1. Update system_config.txt to use TTR=60000 and TTR=120000.
  2. Repeat tests for query handling and polling.

- Result:
  The system adapts correctly to different TTR values.

---

4. Known Issues
1. File Not Found:
  - If a requested file is missing on the origin server, the system logs an error but does not propagate the error back to the requester.
  - Impact: Affects user clarity.
  - Resolution: Add error messages for missing files in QUERYHIT.

2. Port Conflicts:
  - The server fails to start if another application uses the same port range (e.g., 8000+).
  - Impact: Deployment on shared machines.
  - Resolution: Modify port ranges in the configuration file.

3. Message Duplication:
  - If duplicate QUERY messages are sent by a node, they are not de-duplicated if TTL is reset.
  - Impact: Increases network traffic.
  - Resolution: Implement stricter message de-duplication using timestamps.

---

5. Summary
- Push-Based Mechanism:
  - Strong consistency guarantees.
  - Increased bandwidth usage due to broadcasts.
  - Verified correctness in invalidating outdated files.

- Pull-Based Mechanism:
  - Low bandwidth usage but risks temporary staleness.
  - Effective in maintaining consistency with appropriate TTR values.

- Overall:
  - The system works correctly under normal conditions.

- Handles file queries, downloads, invalidations, and polling as expected.
- Improvements can be made to error handling and duplicate message filtering.