CSC340 Project 1 Report

Adelina Chocho, Megan Mohr, Reeya Patel

Client Server Implementation

This project implements a system where clients periodically send heartbeat messages containing file listings to a central server. The Client retrieves its file list from a specified home directory, serializes it into a Protocol object, and transmits it through DatagramSocket. It listens for updates from the server and prints received information every 30 seconds. The Server maintains a record of active clients using ConcurrentHashMaps, marking clients as inactive if no heartbeat is received within a 30 second time span. It broadcasts updates about client availability and file listings to all active clients periodically. It uses multi-threading for operations occurring at the same time and protocol serialization for messaging.

Peer to Peer Implementation

This implementation allows nodes to communicate directly without a central server. Each node sends a UDP packet to share file updates and send heartbeat signals with other nodes for failure detection. Nodes maintain a list of active peers and remove unresponsive ones after a timeout. Multi-threading ensures that sending, receiving, and processing messages happen at the same time.

Protocol Design

Version	Message Type	Packet length
Length		
Node ID		
Mode	True = P2P	False = Client-Server
Time Stamp		
Reserve		
Payload		

Testing Logs

- Killing one machine
  - Client Server- After 30 seconds of inactivity from the node the Server would mark the node as “inactive” in its updates to the other clients. If the client reconnects to wifi, Server’s updates would include the file again as “active” once it receives a new heartbeat from it.
  - Peer to Peer- After 30 seconds, the other nodes will realize it is missing, and will update their list to not include that ode or the files it contains anymore. They will re-update their list if the node comes back on to include its data after it sends a heartbeat when back online.
- Deleting a file
  - Client Server- Server sends an updated list excluding the file that was deleted to all child nodes.
  - Peer to Peer- Each node would update to remove the file that was deleted from their overall list of nodes and their data.

- Adding a file
    - Client Server- Server sends updated list included the file that was added to all child nodes.
    - Peer to Peer- Each node would update to include the file that was added from their overall list of nodes and their data.

Installation/Running Code Process

Run the program using the jar files:

ClientServer:
```
java -jar Server.java
java -jar Client.java
```

PeerToPeer
```
java -jar Peer.java
```