Jesselle Grata - 17978132
Sam Delaney - 17982065

Assignment 2 - Peer to Peer File Sharing

We have decided to split the grade equally 😊

Features:

Peer

A peer is a ‘user’ that will open up the ‘file sharing’ gui. This gui will allow them to connect to a port and then connect to another system.

RMI

For the RMI we had it set up so that when a peer starts it creates itself a RMI. Then created a stub object for others who connect to its registry to call its methods. Each RMI is started on separate ports and other peers can access their methods by locating the peers registry and creating a remote object to call their methods.

File Sharing

The file sharing system works by making the peer that contains the file a temp server so that when the peer requesting the file connects. It will proceed to send the file to that peer and then stop being a server. While the other peer then writes what it received to the appropriate file location.

Timestamp

For the timestamp feature, we used the Chandy-Lamport algorithm. At the start of each peer, they will have a timestamp that begins at 0. For each event (messages between Peers, grabbing/uploading files, etc) that happens for each peer, it will increment by 1. Each peer has a HashMap of the timestamps it currently has for each known peer. When peers message each other, they will also send their own timestamp as well as the other peer’s timestamp. The receiving peer will compare the sender’s timestamp with the timestamp that they have saved locally for the sender. If the new timestamp is larger than what they have saved, then the new timestamp will be saved over the old value. This

Leader Election

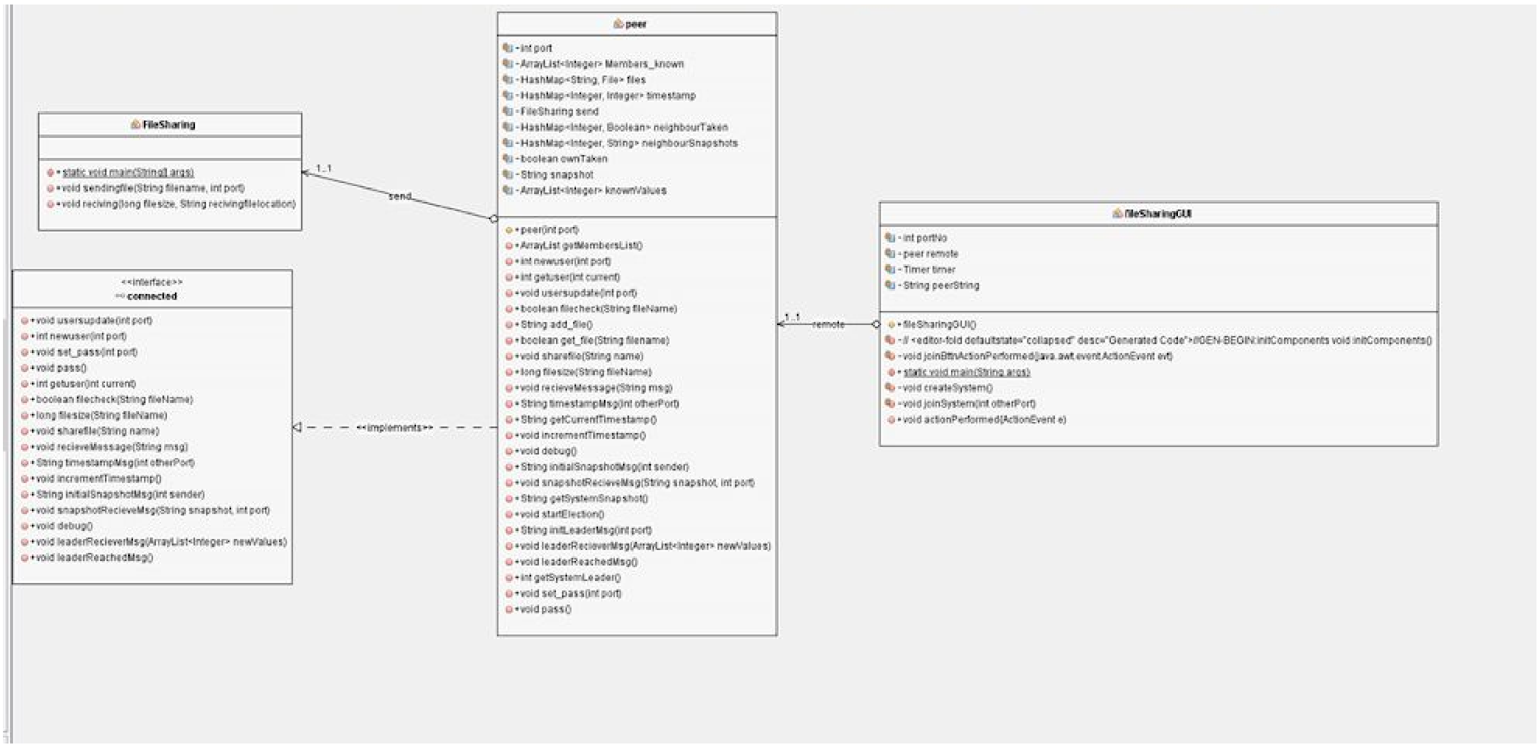
Leader Election for our application begins when a user clicks on the ‘Election’ it begins the ‘Leader Election’ process. We used the Crash-Consensus algorithm. The initial election message will be sent to every single peer which will start the leader-election algorithm. Each peer will broadcast their own port as a suggested leader and during each broadcast, they will save each sent value. At the end of the election process, a pre-agreed strategy (in which the peer with the highest value will become leader) will be used to get the leader.

Snapshot

For our system, a snapshot is called through the ‘debug’ button. Only the leader can debug (which starts the snapshot algorithm). We used the Chandy-Lamport Snapshot algorithm. When a peer receives the snapshot message, it takes a snapshot of their system (currently just saves the files it has saved on their system) and broadcasts it to other peers. Once a peer has received a snapshot from each of their known peers, they will send it back to the initiator.

Token Ring

When the system starts(meaning there is more than 1 peer) they will start to pass around one another in a ring order. Constantly passing it until one of the peers needs to get a file when that happens. The token will stop moving when it gets to the peers that need to get a file. Then that peer will pass the token back on once the file sharing has been completed. Additionally other peers must wait until they have the token to be able to get a file


peer
- int port
- ArrayList<Integer> Members_known
- HashMap<String, File> files
- HashMap<Integer, Integer> timestamp
- FileSharing send
- HashMap<Integer, Boolean> neighbourTaken
- HashMap<Integer, String> neighbourSnapshots
- boolean ownTaken
- String snapshot
- ArrayList<Integer> knownValues
+ peer(int port)
+ ArrayList getMembersList()
+ int newuser(int port)
+ int getuser(int current)
+ void usersupdate(int port)
+ boolean filecheck(String fileName)
+ String add_file()
+ boolean get_file(String filename)
+ void sharefile(String name)
+ long filesize(String fileName)
+ void recieveMessage(String msg)
+ String timestampMsg(int otherPort)
+ String getCurrentTimestamp()
+ void incrementTimestamp()
+ void debug()
+ String initialSnapshotMsg(int sender)
+ void snapshotRecieveMsg(String snapshot, int port)
+ String getSystemSnapshot()
+ void startElection()
+ String initLeaderMsg(int port)
+ void leaderRecieverMsg(ArrayList<Integer> newValues)
+ void leaderReachedMsg()
+ int getSystemLeader()
+ void set_pass(int port)
+ void pass()
FileSharing
+ static void main(String[] args)
+ void sendingfile(String filename, int port)
+ void reciving(long filesize, String recivingfilelocation)
1..1
send
<<interface>>
connected
+ void usersupdate(int port)
+ int newuser(int port)
+ void set_pass(int port)
+ void pass()
+ int getuser(int current)
+ boolean filecheck(String fileName)
+ long filesize(String fileName)
+ void sharefile(String name)
+ void recieveMessage(String msg)
+ String timestampMsg(int otherPort)
+ void incrementTimestamp()
+ String initialSnapshotMsg(int sender)
+ void snapshotRecieveMsg(String snapshot, int port)
+ void debug()
+ void leaderRecieverMsg(ArrayList<Integer> newValues)
+ void leaderReachedMsg()
<<implements>>
fileSharingGUI
- int portNo
- peer remote
- Timer timer
- String peerString
+ fileSharingGUI()
- // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-BEGIN:initComponents void initComponents()
- void joinBttnActionPerformed(java.awt.event.ActionEvent evt)
+ static void main(String args)
- void createSystem()
- void joinSystem(int otherPort)
+ void actionPerformed(ActionEvent e)
1..1
remote