**Program:** Fun File Share [Self-Scaling P2P Network with Central Directory Server]

**Authors**: Ben Durao and Khaled Alsaqri

**User Documentation**: describes how to compile and run your programs

**System Documentation**: describes data structures, design choices, and important algorithms used;

UDP – package built to keep organization of this protocol – Client and Server will use the class methods to communication for informAndUpdate, search, handle download requests, exit.

ReceiverUDP – Simple receiver thread that starts up and endlessly listens for packets on the specified and delivers them.

* stopListening - Closes out the Receiver Socket to stop communications.
* deliverData - Removes the header from the packaged data, resets Sequence number
* checkPacketSeq - Determines if the sent packet is the one that was expected
* run - Initiates the thread to begin listening for incoming packets
  + opens port for communication
  + sets up a packet to receive from sender
  + checks for correct sequence number
  + Build ACK for appropriate sequence number received
  + Send ACK to sender
  + Delivers data
* setPortNum - Assign a port number to the Receiver
* startSlowMode - add in Thread.sleep Methods that will cause packet loss for testing

Sender UDP - Sender thread that receives data from above (Main.java) and makes a packet(s) then senders the packet(s) to the receiver and waits for a response ACK.

* startSender - Opens Port to communicate over
* stopSender - Closes Port to stop communication.
* rdtSend - Stores Appropriate Status Code, Receives data, Pass data to build packet, Send Packet to Receiver
  + main loop in this method reading in data from the input stream to build into packet then send out to receiver waiting for transmission
* adjustTimeout - formula to estimate the current round trip to balance with packet loss
* checkACK - Determine if the packet with sequence number we sent matches the seq num the receiver was expecting with corresponding ACK sent to us
* recACK - Listen on our open port to receive ACK for our sent packet to make sure the Receiver got what we sent.
* makePacket - Builds the packet header and attaches it to the data portion of the packet [includes status code,sequence num, flag]
* makePacketData - Read in the data from the input stream to build the data portion of the packet to the appropriate length according to MTU
* sendPacket - When not listening for an ACK, and have data to send - Push whole packet built over the open port to receiver then starts the timer [and adjusts it], then sends packet to specified IP Address and Open Port of Receiver
* setPortNum - Assign the Sender a Port Number
* setTarPort - Set the Receiver's Open Port that we will send our Packet through
* setTarIPAddress - Identify the IP address we will send our Packet to
* setSenderPort - Set the port we will open to allow communication over
* startSlowMode - add in Thread.sleep Methods that will cause packet loss for testing

MAIN – the driver of the program will allow for 2 different startups SERVER START and CLIENT START. Server start will initiate the directory server to handle all requests by clients and should be launched and setup first. Client threads can then be starts so they can informAndUpdate to the server for connection then query the server and download files that are available to share.

Client

* startSenderUDP - Initiate the client to start sending data to Server with its designated IP Address and the port it is listening on
* startReceiverUDP - Initiate the client as a Receiver for UDP connections from the Server
* slowMode - Put the client in slow mode to test for Timeout recovery
* informAndUpdate - Send the selected files to the Server to show what files are available for sharing
* search - Query the server for a certain file to see if available for sharing
* download - Send download request to the server to ask for TCP connection info of peer of queried file
* exit - Tell the server that the Client is closing out the program and that files will not be available.
* clearTableModel – update UI to wipe the table clean of rows
* rcvServerOk - Accept Server Status Code that message was received
* rcvServerErr - Accept Server Status Code that an error occurred
* rcvSearchResponse - Accept Server response that a file was searched for
* rcvExitResponse - Successful Exit from Server to end the Client
* startTCPSender - Initiate the client to start sending data to client receiver with its designated IP Address and the port it is listening on
* startReceiverTCP - After successful download request - Receive the connection info for the peer to accept the download transfer from.
* updateTableModel – update UI for new data
* getFileInfo – get information about selected file in UI [Name, Size, FilePath]
* breakMessage - Break apart packet to check for the status code as defined by the server
* isExpectedMessage - breakMessages helper to check for expected Status Code [True if expected]

Server

* startSenderUDP - Initiate the server to start sending data to client with its designated IP Address and the port it is listening on
* startReceiverUDP - Initiate the server as a Receiver for UDP connections from the client
* updateSender - update sender with new IP address and port number as specified
* setSlowMode - Put the server in slow mode to test for Timeout recovery
* refreshDirectory – update the directory if a file has changed
* addClient – update amount of clients running
* rmvClient – update amount of clients running
* updateTableModel – update UI for new data
* clearTableModel – update UI to wipe the table clean of rows
* sendHostClientReq - send connection info to the Host so they can link up for TCP transmission of files
* clientConnectionInfo - Retrieve necessary info for TCP connection for both hosts and send server response with status code
* rcvInformAndUpdate - handle informAndUpdate request by showing those files specificed as available for sharing the directory and send appropriate status code to sender
* rcvClientSearchReq - search through directory for file that matches the String in the query
* rcvDownloadReq - retrieve connection info necessary for Hosts to connect over TCP for file transfer
* rcvExitReq - remove user from directory
* clientSearchResponse - Send the client the results of its query
* informAndUpdateResponse - server tells client it has updated its directory to include shared files
* exitResponse - Send status code that server is exiting
* breakMessage - Break apart packet to select appropriate status code to be sent by the server

Main

* main – Initiate program through GUI

recMessages - Interface to be used by Receiver and Sender so that they can break up the messages by their appropriate delimiters. This will allow them to receive and use the correct data.

* breakMessages – will allow client and server to scan through messages for appropriate delimiting to retrieve needed information to use for connection information