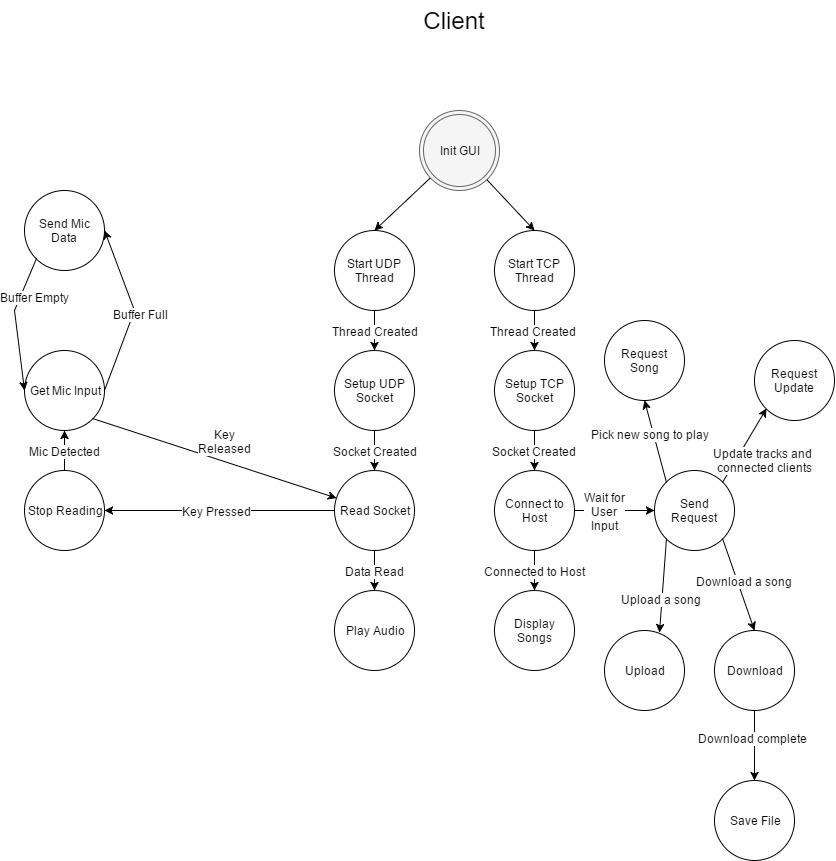
Comm Audio – Design

Robert Arendac, Matt Goerwell, Alex Zielinski

## Client FSM



## Client Pseudo Code

**INIT GUI  
Alex Zielinski – March 18**

1. Add File menu item
   1. Add exit sub menu item
2. Add About menu item
3. Add Station Group box
   1. Add IP text box
   2. Add Port text box
   3. Add Connect button
4. Add Clients group box
   1. Add Client list display box
5. Add Track group box
   1. Add Current track playing text
6. Add Tracklist group box
   1. Add Update button
   2. Add Upload button
   3. Add track list display box
   4. Add Play Selected Song button
   5. Add Download Selected Song button
7. Setup Winsock session

**START UDP THREAD  
Robert Arendac – March 25**

1. Create a new thread off of the main one
2. No information needs to be given to the thread
3. Function should be one that is responsible for streaming audio and sending mic input

**SETUP UDP SOCKET  
Robert Arendac – March 19**

1. Create UDP socket (ERROR CHECK)
2. Set broadcast option (ERROR CHECK)
3. Initialize address information
4. Bind socket (ERROR CHECK)

**READ SOCKET  
Matt Goerwell – March 28**

1. Prepare receive buffer
2. Set buffer as a queue of audio sources (stream)
3. Start *PLAY* *AUDIO[STATE]* thread with receive buffer
4. In a forever loop
   1. If a Mic signal is received (mic key pressed)
      1. Go to stop reading State
   2. If the reading flag is set
      1. Receive from Multicast IP
      2. Store packet content into buffer
      3. Update Stream Queue

**STOP READING  
Alex Zielinski – March 26**

1. Set stop reading flag
2. Go to GET MIC INPUT*[STATE]*

**GET MIC INPUT  
Alex Zielinski – March 24**

1. Get list of available input devices from machine
2. Set Audio codec
3. Set sample rate
4. Set Bitrate
5. Set audio channel
6. If the start mic button is pressed
   1. Start saving mic input to buffer
7. If buffer full
   1. Go to send mic data state
8. If the start mic button is released
   1. Stop mic input
   2. Go to read socket state to read incoming data from server

**SEND MIC DATA  
Alex Zielinski – March 28**

1. Format buffered data
2. Write buffer to socket (ERROR CHECK)

**PLAY AUDIO  
Matt Goerwell – March 29**

1. Wait for initial buffer filling.
2. While buffer has content
   1. Play stream buffer

**START TCP THREAD  
Robert Arendac – March 19**

1. Should be run off the main thread (no need to explicitly create a new one)
2. Purpose is to manage the connection to the server and send requests
3. Create TCP socket (ERROR CHECK)

**CONNECT TO HOST  
Robert Arendac – March 19**

1. Extract IP address from the GUI
2. Check to see that the IP is a valid one
3. If IP is invalid
   1. Exit CONNECT TO HOST*[STATE]*
4. Initialize address information
5. Connect to the server
6. Go to DISPLAY SONGS*[STATE]*

**DISPLAY SONGS  
Robert Arendac – March 27**

1. Read on the socket and specify a completion routine
   1. Make sure a message was received without error
   2. Parse the message and store each delimited song name into a list of songs
2. Update GUI with the song list

**SEND REQUEST  
Matt Goerwell – March 30**

1. Get user request
   1. If request is to play a song
      1. Go to REQUEST SONG*[STATE]*
   2. If request is to Upload a song
      1. Go to UPLOAD*[STATE]*
   3. If request is to download a song
      1. Go to DOWNLOAD*[STATE]*
   4. If request is to refresh lists
      1. Go to REQUEST UPDATE*[STATE]*
   5. If request is to close connection
      1. Send teardown Request over TCP
      2. Exit SEND REQUEST*[STATE]*

**REQUEST SONG  
Matt Goerwell – March 30**

1. Extract selected song from song list.
2. Send a message containing
   1. The type “request song”
   2. The song name
3. Wait for response (Error or success)
4. Display response

**UPLOAD  
Matt Goerwell – March 31**

1. Prompt user to select file to upload
2. If file is not in the correct format (.wav or .mp3)
   1. Display error message
   2. Exit UPLOAD*[STATE]*
3. Prepare buffer of file sections
4. Send request type and file name
5. Wait for response (Error or continue)
   1. If error
      1. Display message
      2. Free buffer
      3. Exit UPLOAD*[STATE]*
6. For each section in buffer
   1. Send section of data
7. Send transfer complete message
8. Free buffer

**DOWNLOAD  
Matt Goerwell – March 31**

1. Extract selected song from song list.
2. Send Request over TCP
3. Receive a message and specify a completion routine
   1. If message is request complete
      1. Close file
      2. Exit DOWNLOAD*[STATE]*
   2. If message is Song not found
      1. close file
      2. delete file
      3. Exit DOWNLOAD*[STATE]*
   3. Write Packet data to buffer
   4. Post another receive with the same completion routine
4. Go to SAVE FILE*[STATE]* with the buffer

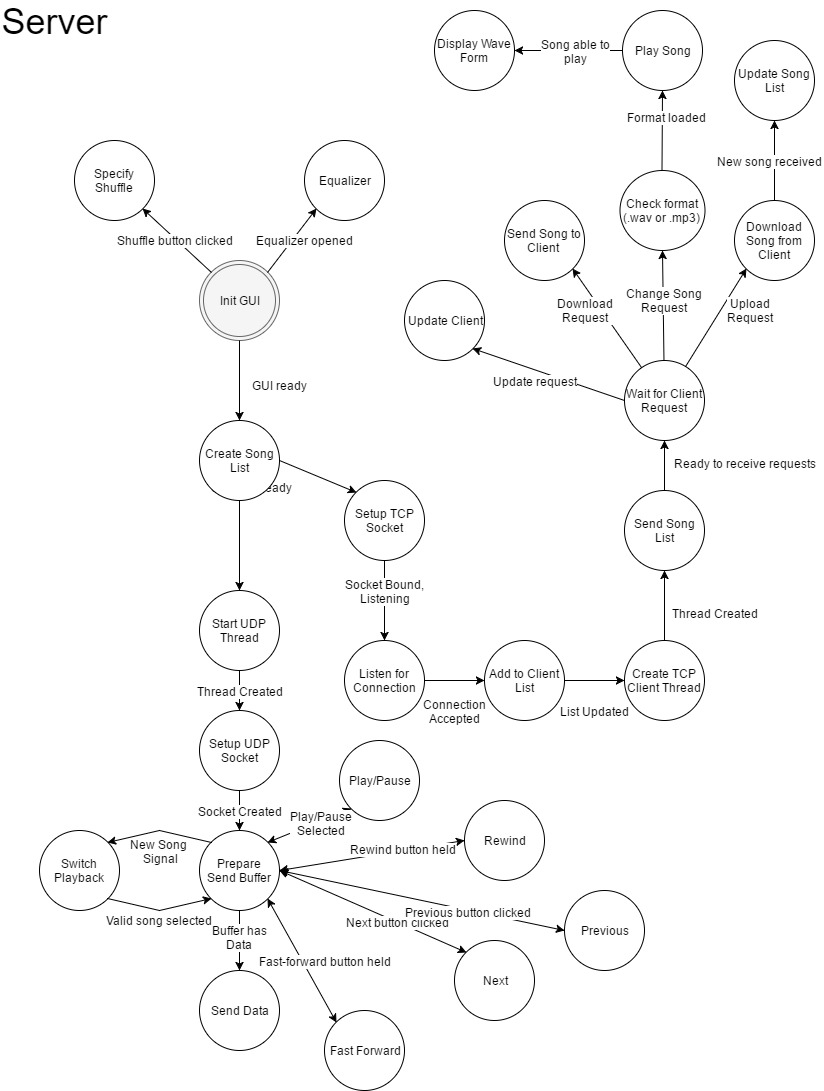
**SAVE FILE  
Robert Arendac – March 30**

1. Open a Qt file browser
2. Get the save path
3. Open the file for writing only
4. Store the data into the file
5. Close the file

**REQUEST UPDATE  
Matt Goerwell – March 30**

1. Generate update request message
2. Send message over TCP
3. Wait for response
4. Parse client list message
5. Update clients list
6. Wait for another response
7. Update songs list

## Server FSM



## Server Pseudo Code

**INIT GUI   
Alex Zielinski – March 18**

1. Add File menu item
   1. Add exit sub menu item
2. Add About menu item
3. Add Server group box
   1. Add Start button
   2. Add Stop button
   3. Add status text
4. Add Client group box
   1. Add Client list display box
5. Add Song group box
   1. Add Current song playing
   2. Add Rewind button
   3. Add Play/Pause button
   4. Add Stop button
   5. Add Fast forward button
   6. Add Track list title text
   7. Add Shuffle radio button
   8. Add Track list display box
   9. Add Play selected track button
6. Start a Winsock session

**CREATE SONG LIST  
Robert Arendac – March 25**

1. Create a buffer to hold all the song names
2. Get the path containing the folder of songs
3. Check for valid directory
4. while there are files containing the extension ".wav" or “.mp3”
   1. store the filename in the song name buffer

**START UDP THREAD  
Robert Arendac – March 25**

1. Create a new thread off the main one
2. Give the multicast address to the thread
3. Purpose is to send audio streaming data

**SETUP UDP SOCKET  
Robert Arendac – March 26**

1. Setup multicast group address and port settings
2. Setup multicast intervals (how often to send)
3. Setup time to live (scope of multicast live)
4. Create UDP socket (ERROR CHECK)
5. Initialize address information
6. Bind socket (ERROR CHECK)
7. Join multicast group

**PREPARE SEND BUFFER  
Matt Goerwell – March 26**

1. As a loop:
   1. If Shuffle is true
      1. Load song from file name at the index indicated by Current Song Counter in the Shuffle Playlist
   2. Otherwise
      1. Load song from file name at the index indicated by Current Song Counter in the standard song list
   3. Prepare buffer
   4. Load initial chunks
   5. Go to SEND DATA*[STATE]*
      1. If return of Send Data is signal code
         1. Go to top of loop
   6. Close Song File
   7. Increment Song counter
   8. If song counter is greater than the number of songs available
      1. Reset song counter

**SEND DATA  
Matt Goerwell – March 26**

1. In a forever loop:
   1. Convert chunk to packet
   2. Send packet to multicast IP address
   3. Remove chunk from buffer
   4. Advance buffer forward
   5. Load next chunk into buffer
      1. If last chunk loaded was last chunk of song:
         1. break loop
2. Prepare/send remaining packets
3. Return how much song was played

**SWITCH PLAYBACK  
Matt Goerwell – March 30**

1. Send signal to stop sending audio packets
2. Find specified song in the song list
3. Open the media file
4. Send contents to PREPARE BUFFER[STATE]

**PLAY/PAUSE  
Designed by Robert Arendac  
To be programmed by Alex Zielinski – March 29**

1. If previous flag from SEND DATA was set to play
   1. switch icon from pause to play
   2. tell SEND DATA to pause
2. If previous flag from SEND DATA was set to pause
   1. switch icon from play to pause
   2. tell SEND DATA to resume

**PREV  
Designed by Robert Arendac  
To be programmed by Alex Zielinski - March 29**

1. Find the current song in the playlist
2. Go back one song (the previous song)
3. Go to SWITCH PLAYBACK with the grabbed song

**NEXT**   
**Designed by Robert Arendac  
To be programmed by Alex Zielinski - March 29**

1. If shuffle is enabled
   1. Go to next song in shuffle playlist
2. Otherwise
   1. find current song in container of alphabetically stored songs
   2. select next song
3. Go to SWITCH PLAYBACK with the grabbed song

**REWIND  
Designed by Robert Arendac  
To be programmed by Alex Zielinski - March 29**

1. While the rewind button is pushed down
   1. Seek backwards from the current song point by a small amount (1?)
   2. If you seek to the beginning of the file
      1. Exit REWIND

**FAST FORWARD  
Designed by Robert Arendac  
To be programmed by Alex Zielinski - March 29**

1. While the fast forward button is pushed down
   1. seek forwards from the current song point a small amount
   2. If you seek to the end of the file
      1. Go to NEXT*[STATE]*

**LISTEN FOR CONNECTION  
Robert Arendac – March 19**

1. While the program is alive
   1. accept an incoming connection on an accepting socket
   2. go to ADD TO CLIENT LIST[STATE]

**ADD TO CLIENT LIST  
Robert Arendac – March 25**

1. Extract the address from the accepting socket
2. Add the address to a client list buffer

**CREATE TCP CLIENT THREAD  
Robert Arendac – March 19**

1. Create a new thread to service each TCP connected client
2. Give the thread the socket the connection was accepted on
3. Purpose of the thread is to handle requests from each client simultaneously

**SETUP TCP SOCKET  
Robert Arendac – March 19**

1. Create TCP socket (ERROR CHECK)
2. Initialize address information
3. Bind the socket (ERROR CHECK)
4. Set socket to listen (ERROR CHECK)

**SEND SONG LIST  
Robert Arendac – March 26**

1. Create message buffer
2. For every entry in the song list
   1. Append entry to the message
   2. Append New line character to message
3. Send message
4. Go to wait for client request State

**WAIT FOR CLIENT REQUEST  
Matt Goerwell – March 28**

1. While connection is open
   1. Poll connection for any TCP requests
   2. If a request is detected:
      1. Parse request type
         1. If request is to play song
            1. Go to CHECK FORMAT*[STATE]* with song name
         2. If request is to download a song
            1. Go to SEND SONG TO CLIENT*[STATE]* with song name
         3. If request is to upload a song with desired filename
            1. Go to DOWNLOAD SONG FROM CLIENT*[STATE]*
         4. If request is to update client’s information
            1. Go to UPDATE CLIENT*[STATE]*
         5. If request is to close connection
            1. Initiate teardown
            2. Close the connection
2. End thread

**CHECK FORMAT  
Alex Zielinski – March 31**

1. If requested file type is MP3
   1. Load MP3 file
2. Otherwise
   1. Load WAV file
3. Go to PLAY SONG with loaded file type

**PLAY SONG REQUEST  
Matt Goerwell – March 28**

1. Check Song list for requested song
   1. If song exists on server
      1. Send Signal to streaming thread
      2. Set Song counter to appropriate index
      3. Send Request successful message
   2. If song Does not Exist
      1. Send “Song could not be found” message

**VISUALIZATION  
Alex Zielinski – April 1**

1. Create the buffer, which is simply a byte array made by Qt
2. Populate the buffer (in the case of “play generated tone” and “play WAV file” modes)
3. Configure the Qt multimedia audio objects.
4. Direct data flow between the buffer and the Qt multimedia audio objects.
5. Manage the calculations – spectrum analysis and level
6. Provide status updates, by emitting signals which are consumed by the main window.

**EQUALIZER  
Alex Zielinski – April 1**

1. Map 5 sliders to 5 frequency ranges
2. If slider is moved up then increase volume of given frequency range
3. If slider is moved down then increase volume of given frequency range

**SEND SONG TO CLIENT  
Matt Goerwell – March 31**

1. Check Song list for requested song
   1. If song exists on server
      1. Prepare Buffer
      2. send each chunk in the buffer
      3. send request complete message
   2. If song doesn't exist
      1. Send Song could not be found message over TCP

**DOWNLOAD SONG FROM CLIENT  
Matt Goerwell – March 31**

1. Get name of uploaded song
2. Check existing songs.
3. If song already exists:
   1. Send Song Already exists message over TCP
   2. Exit DOWNLOAD SONG FROM CLIENT*[STATE]*
4. Prepare file for writing with song name
5. Send continue Message over TCP
6. Post a receive with a specified completion routine
   1. If transfer complete is indicated
      1. stop receiving
   2. Append data to file
   3. Post another receive with same completion routine
7. Close file
8. Go to Update Song list state with name of new song.

**UPDATE CLIENT  
Designed by Matt Goerwell  
To be programmed by Robert Arendac – March 30**

1. Prepare message buffer
2. For each client in client list
   1. Append client IP to message
   2. Append new line character
3. Send client list
4. Clear the message buffer
5. For each song in the song list
   1. Append song name to message
   2. Append new line character
6. Send song list

**UPDATE SONG LIST  
Robert Arendac – March 30**

1. Read the filename that was passed in
2. add it to the song list buffer
3. Add new radio button to GUI with song name

**SPECIFY SHUFFLE  
Matt Goerwell – April 1**

1. If shuffle is false
   1. Prepare Shuffle Playlist out of all songs
   2. Set Current Song Counter to zero
   3. Set shuffle to true
2. Otherwise
   1. Set shuffle to false
   2. Clear Shuffle Playlist
   3. Create a playlist of all songs ordered in alphabetical order
   4. Find the currently playing/selected song
   5. Set the current song counter to be the index of the current song in the container