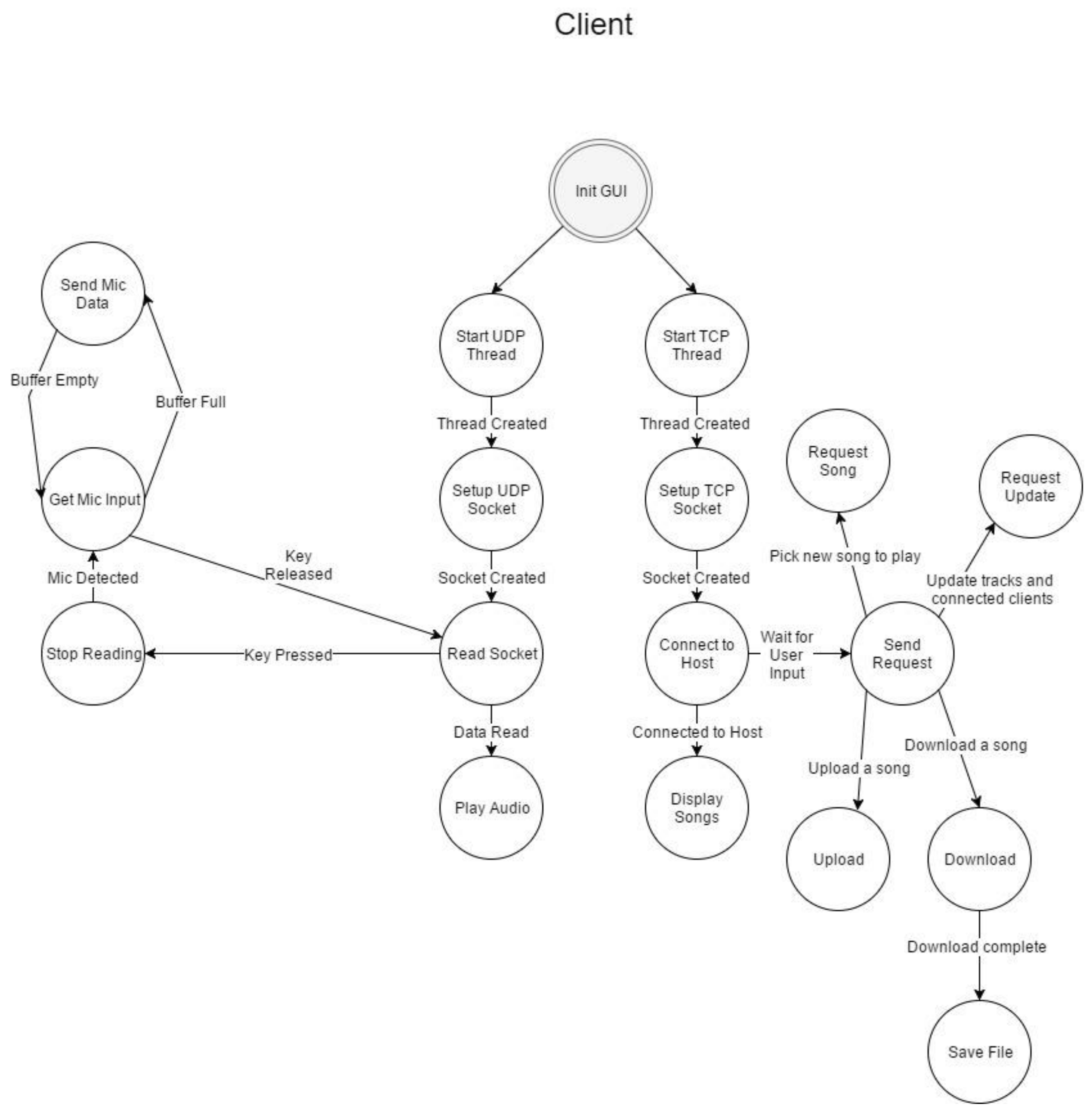
Comm Audio – Design

Robert Arendac, Matt Goerwell, Alex Zielinski

Client FSM


Client
Init GUI
Start UDP Thread
Start TCP Thread
Thread Created
Thread Created
Setup UDP Socket
Setup TCP Socket
Socket Created
Socket Created
Read Socket
Connect to Host
Data Read
Connected to Host
Play Audio
Display Songs
Key Pressed
Key Released
Stop Reading
Mic Detected
Get Mic Input
Buffer Full
Buffer Empty
Send Mic Data
Wait for User Input
Send Request
Pick new song to play
Request Song
Update tracks and connected clients
Request Update
Upload a song
Upload
Download a song
Download
Download complete
Save File

Client Pseudo Code

INIT GUI

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

START UDP THREAD

```
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

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

READ SOCKET

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
    - a) If a Mic signal is received (mic key pressed)
        - i) Go to stop reading State
    - b) If the reading flag is set
        - i) Receive from Multicast IP
        - ii) Store packet content into buffer
        - iii) Update Stream Queue

STOP READING

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

GET MIC INPUT

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
    - a) Start saving mic input to buffer
7) If buffer full
    - a) Go to send mic data state
8) If the start mic button is released
    - a) Stop mic input
    - b) Go to read socket state to read incoming data from server

SEND MIC DATA

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

PLAY AUDIO

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

START TCP THREAD

```
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
```

SETUP TCP SOCKET

```
1. Create TCP socket (ERROR CHECK)
```

CONNECT TO HOST

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

DISPLAY SONGS

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

SEND REQUEST

```
1) Get user request
    a) If request is to play a song
        i) Go to REQUEST SONG[STATE]
    b) If request is to Upload a song
        i) Go to UPLOAD[STATE]
    c) If request is to download a song
        i) Go to DOWNLOAD[STATE]
    d) If request is to refresh lists
        i) Go to REQUEST UPDATE[STATE]
    e) If request is to close connection
        i) Send teardown Request over TCP
        ii) Exit SEND REQUEST[STATE]
```

REQUEST SONG

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

UPLOAD

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

DOWNLOAD

```
1) Extract selected song from song list.
2) Send Request over TCP
3) Receive a message and specify a completion routine
    a) If message is request complete
        i) Close file
```

        ii) Exit DOWNLOAD*[STATE]*
    b) If message is Song not found
        i) close file
        ii) delete file
        iii) Exit DOWNLOAD*[STATE]*
    c) Write Packet data to buffer
    d) Post another receive with the same completion routine
4) Go to SAVE FILE*[STATE]* with the buffer

SAVE FILE

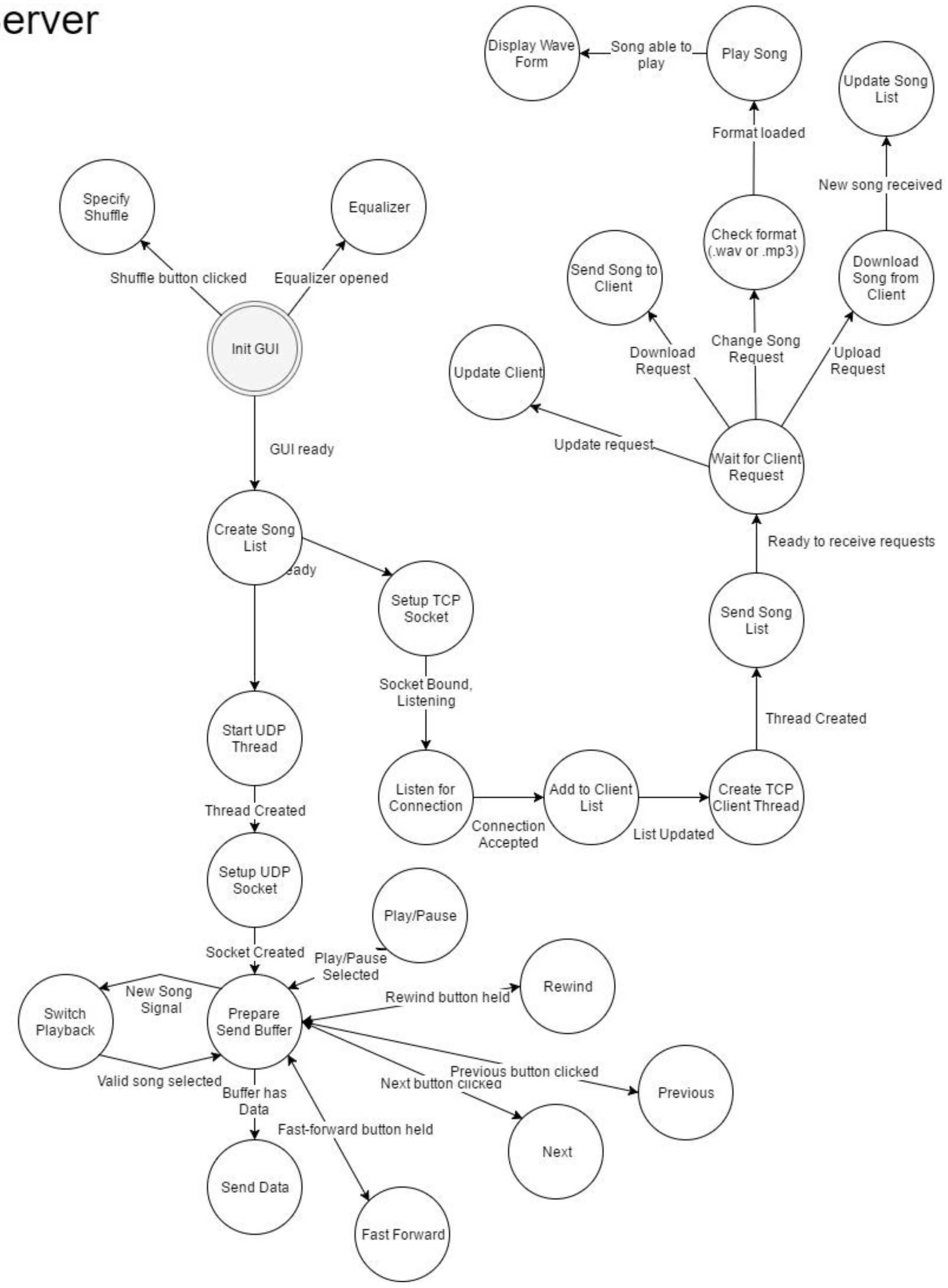
1. Open a Qt file browser
2. Get the save path
3. Open the file for writing only
4. Store the data into the file
5. Close the file

REQUEST UPDATE

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



Server Pseudo Code

INIT GUI

```
1) Add File menu item
   a) Add exit sub menu item
2) Add About menu item
3) Add Server group box
   a) Add Start button
   b) Add Stop button
   c) Add status text
4) Add Client group box
   a) Add Client list display box
5) Add Song group box
   a) Add Current song playing
   b) Add Rewind button
   c) Add Play/Pause button
   d) Add Stop button
   e) Add Fast forward button
   f) Add Track list title text
   g) Add Shuffle radio button
   h) Add Track list display box
   i) Add Play selected track button
6) Start a Winsock session
```

CREATE SONG LIST

```
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"
      a. store the filename in the song name buffer
```

START UDP THREAD

```
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

```
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

```
1)  As a loop:
    a)  If Shuffle is true
        i)  Load song from file name at the index indicated by Current Song Counter in the Shuffle
            Playlist
    b)  Otherwise
        i)  Load song from file name at the index indicated by Current Song Counter in the standard
            song list
    c)  Prepare buffer
    d)  Load initial chunks
    e)  Go to SEND DATA[STATE]
        i)  If return of Send Data is signal code
            (1)  Go to top of loop
    f)  Close Song File
    g)  Increment Song counter
    h)  If song counter is greater than the number of songs available
        i)  Reset song counter
```

SEND DATA

```
1)  In a forever loop:
    a)  Convert chunk to packet
    b)  Send packet to multicast IP address
    c)  Remove chunk from buffer
    d)  Advance buffer forward
    e)  Load next chunk into buffer
        i)  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

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

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

PREV

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

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

REWIND

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

FAST FORWARD

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

LISTEN FOR CONNECTION

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

ADD TO CLIENT LIST

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

CREATE TCP CLIENT THREAD

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

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

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

WAIT FOR CLIENT REQUEST

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

CHECK FORMAT

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

PLAY SONG REQUEST

1) Check Song list for requested song
   a) If song exists on server
      i) Send Signal to streaming thread
      ii) Set Song counter to appropriate index
      iii) Send Request successful message
   b) If song Does not Exist
      i) Send “Song could not be found” message

VISUALIZATION

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

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

1) Check Song list for requested song
   a) If song exists on server
      i) Prepare Buffer
      ii) send each chunk in the buffer
      iii) send request complete message
   b) If song doesn't exist
      i) Send Song could not be found message over TCP

DOWNLOAD SONG FROM CLIENT

```
1) Get name of uploaded song
2) Check existing songs.
3) If song already exists:
    a) Send Song Already exists message over TCP
    b) 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
    a) If transfer complete is indicated
        i) stop receiving
    b) Append data to file
    c) Post another receive with same completion routine
7) Close file
8) Go to Update Song list state with name of new song.
```

UPDATE CLIENT

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

UPDATE SONG LIST

```
    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

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