CommAudio Design Document

COMP4985 – FINAL PROJECT

Gabriel Lee | Eunwon Moon | Oscar Kwan | Krystle Bulalakaw
GLOKKBEWM | HTTPS://GITHUB.COM/GLOKKBEWM/COMMAUDIO

Table of Contents

Introduction

The Comm Audio project utilizes Windows sockets programming techniques to connect clients to a server. The server has the ability to stream songs over a network via multicast UDP messages.

Client users must interact with the Qt GUI to optionally enter a host address and IP; otherwise default values are used. Once connected, clients can see a list of connected users and available songs from the server to stream.

The server behaves like a radio station, in which it can select songs to stream to the channel, handle song requests, etc.

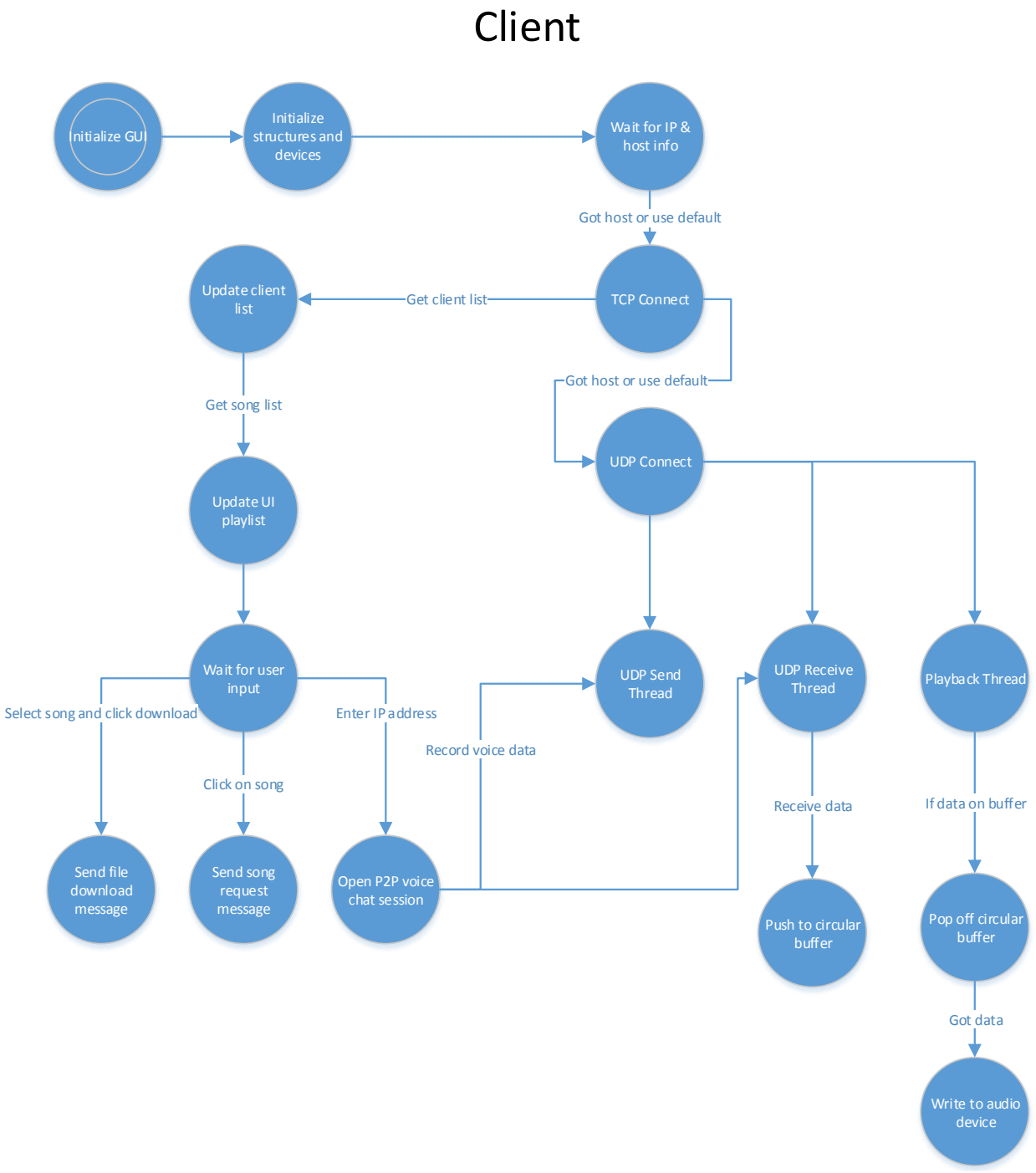
TCP and UDP sending and receiving are done via multi-threaded completion routines. A large amount of data is coming in and out of the server, so buffering must be handled effectively as to keep the transfers fast and smooth. One solution for this is to implement circular buffers, a superior method to process asynchronous I/O. Data is transferred between threads and/or the client and server in blocks of bytes in order to keep a cohesive stream available.

This document outlines our approach to meeting the requirements of this project:

- Connecting to a known remote server
- Capability to transfer and play sound on either the client or server
- Options to save and retrieve sound files
- Default sound file .wav format
- Two-way microphone support
- Multicasting capability

Design

Client



Pseudocode

Client

Initialize GUI
Setup Qt GUI Widget styles
Connect Qt signals and slots
Go to **Initialize structures and devices**

Initialize Structures and Devices
Search for device
Open device
Initialize circular buffers (size, head, tail)
Go to **Wait for IP & Host Info**

Wait for IP & host info
Validate IP and host
If valid Go to **TCP Connect**

TCP Connect
Create socket
Bind address to socket
If connect succeeds
    Go to **Update Client List**
    Go to **UDP Connect**

Update Client List
Receive message of all connected clients
For each client in the message, add to GUI client list
Go to **Update UI Playlist**

Update UI Playlist
Receive message of all songs available on server
For each song in the message, add to GUI playlist
Go to Wait for User Input

UDP Connect
Create UDP Socket
Initialize address structure
Bind address to structure
Set multicast settings
Create **UDP Send, UDP Receive, Playback Threads**

Wait for User Input
If selected song and clicked download, **Send File Download Message**
If double clicked song, **Send Song Request Message**
If entered IP address, **Open P2P Voice Chat Session**
    Go to **UDP Send Thread**

UDP Send Thread

Forever loop
    If in P2P voice session and recorded voice data
        Format voice data
        Write voice data to buffer
        Send buffer

UDP Receive Thread

Forever loop
    If received data
        Go to **Push to Circular Buffer**

Push to Circular Buffer

Push data to circular buffer head
Increment head index

Playback Thread

Initialize audio input / output settings
Forever loop:
    If there is data on circular buffer
    Pop data off circular buffer

Pop off Circular Buffer

Pop data off ring buffer tail
Increment tail index
Write data to audio output device

Open P2P Voice Chat Session

Establish TCP connection to desired peer address
Go to **UDP Send Thread**
Go to **UDP Receive Thread**

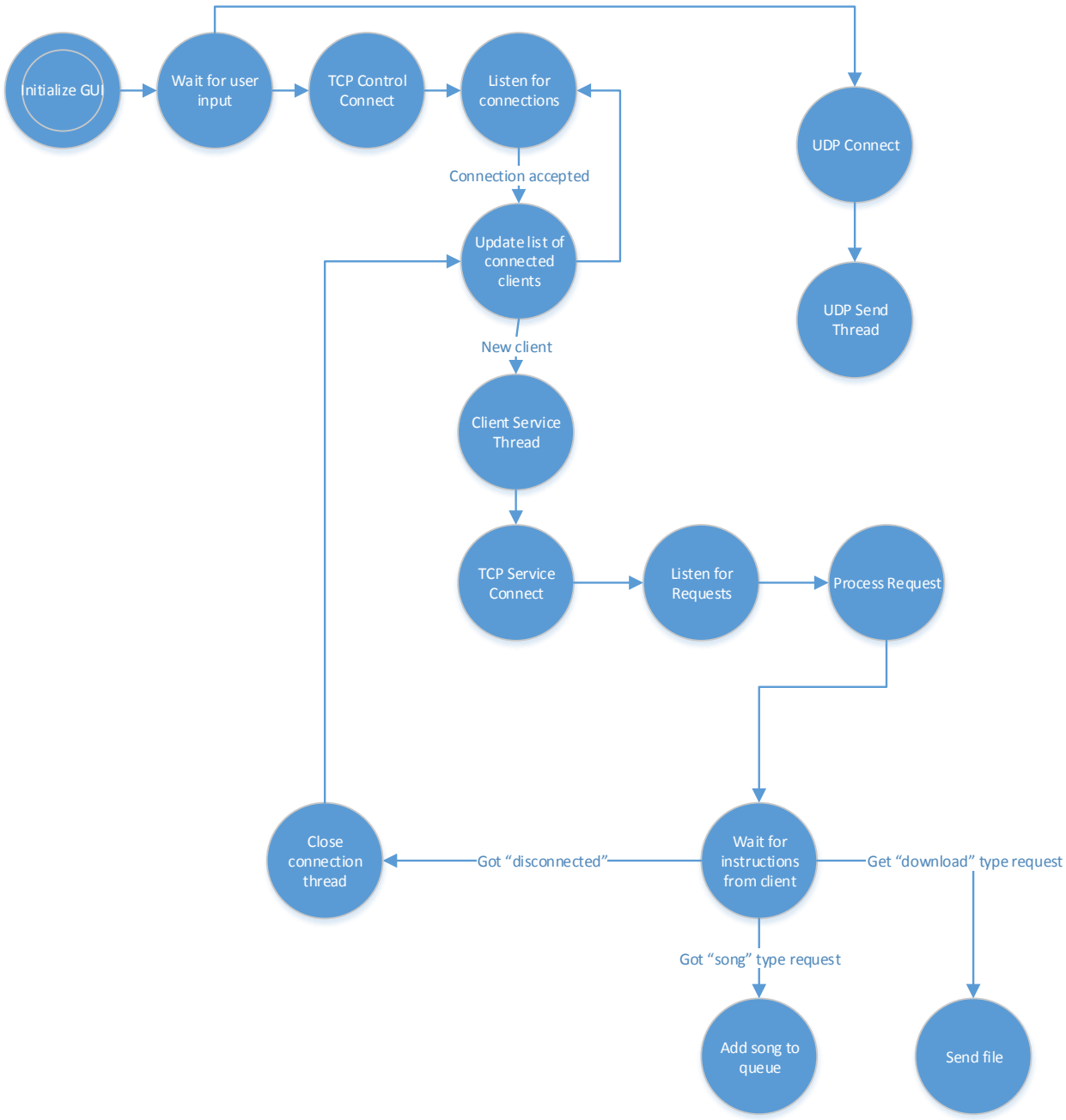
Send request message

Append request type to message
Append request name to message
Append request data to message
Send message to TCP socket

Write to Audio Device

Copy received data to QBuffer
Write QBuffer to audio output device

Server


Initialize GUI
Wait for user input
TCP Control Connect
Listen for connections
UDP Connect
Connection accepted
Update list of connected clients
UDP Send Thread
New client
Client Service Thread
TCP Service Connect
Listen for Requests
Process Request
Close connection thread
Got “disconnected”
Wait for instructions from client
Get “download” type request
Got “song” type request
Add song to queue
Send file

Pseudocode

Server

Initialize GUI

Set up Qt GUI Widgets style
Connect Qt slots and signals
Initialize global variables
Go to **Wait for User Input**

TCP Control Connect

Create socket
Bind address to socket
If connect succeeds
    Go to **UDP Connect**
    Go to **Listen for Connections**

UDP Connect

Create UDP Socket
Initialize address structure
Bind address to structure
Set multicast settings
Create **UDP Send, UDP Receive Threads**

UDP Send Thread

Forever loop
    If in P2P voice session and recorded voice data
        Format voice data
        Write voice data to buffer
        Send buffer

Listen for connections

Forever loop
    Listen for connections
    If connection accepted
        Go to **Update list of clients**

Update list of clients

Add/remove client name and IP to list of connected clients
Send list to all clients
If new client
    Go to **Client Service Thread**

Client Service Thread

Create thread to listen on TCP socket
Go to **TCP Service Connect**

tcp Service Connect

Create socket
Bind address to socket
Go to **Listen for Requests**

Listen for Requests

Forever loop
    Receive message from client
    Deserialize message:
        Get request type
        Get song request name
        Get request data
    Go to **Process Request**

Process Request

If got “download” type request
    Send File
        Open & read file
        Copy readbytes to buffer
        Send buffer to TCP socket
If got “song” type request
    Add Song to Queue
        Get song name
        If not playing any song, play this song
        If playing song, add to queue
If got “disconnected”
    Close connection thread
        Close socket
        End thread
        Cleanup