March 4, 2015

Comm Audio

Thomas Tallentire

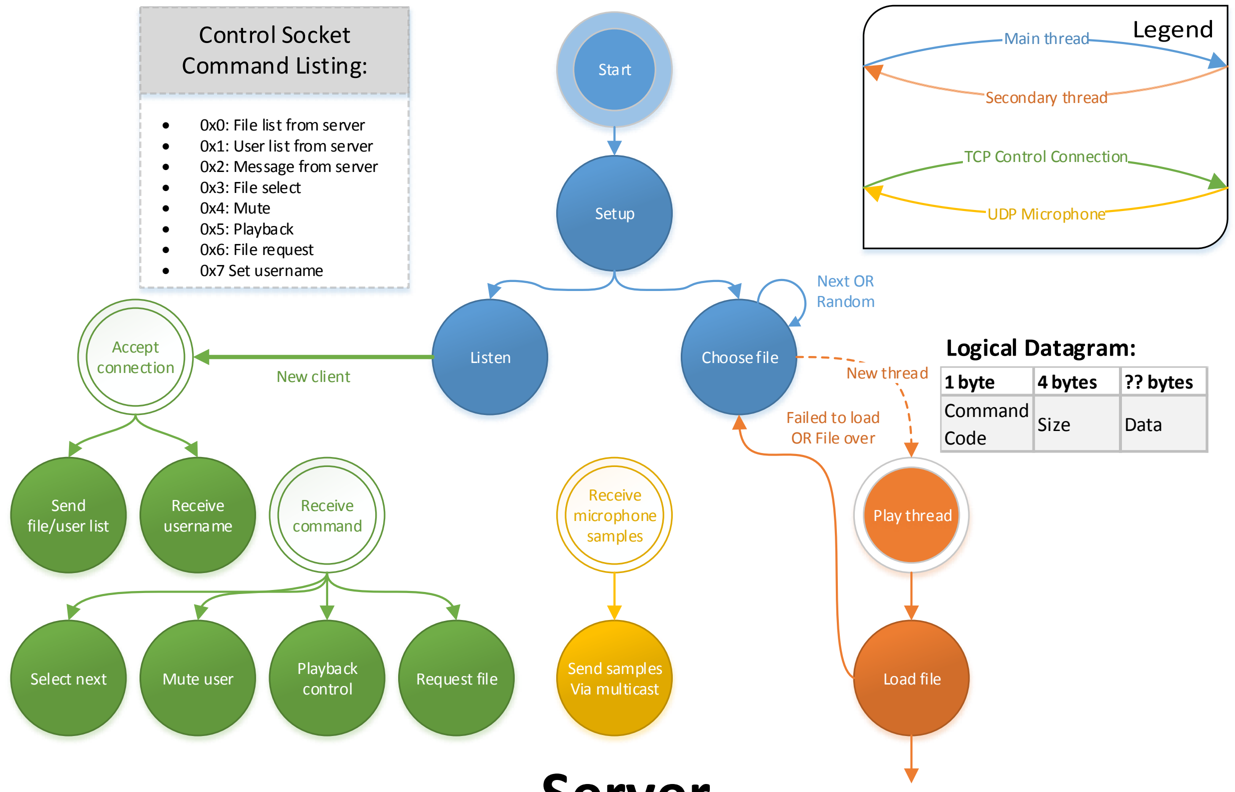
Marc Rafanin

Marc Vouve

Lewis Scott

March 4, 2015

State Transition Diagrams



Control Socket Command Listing:

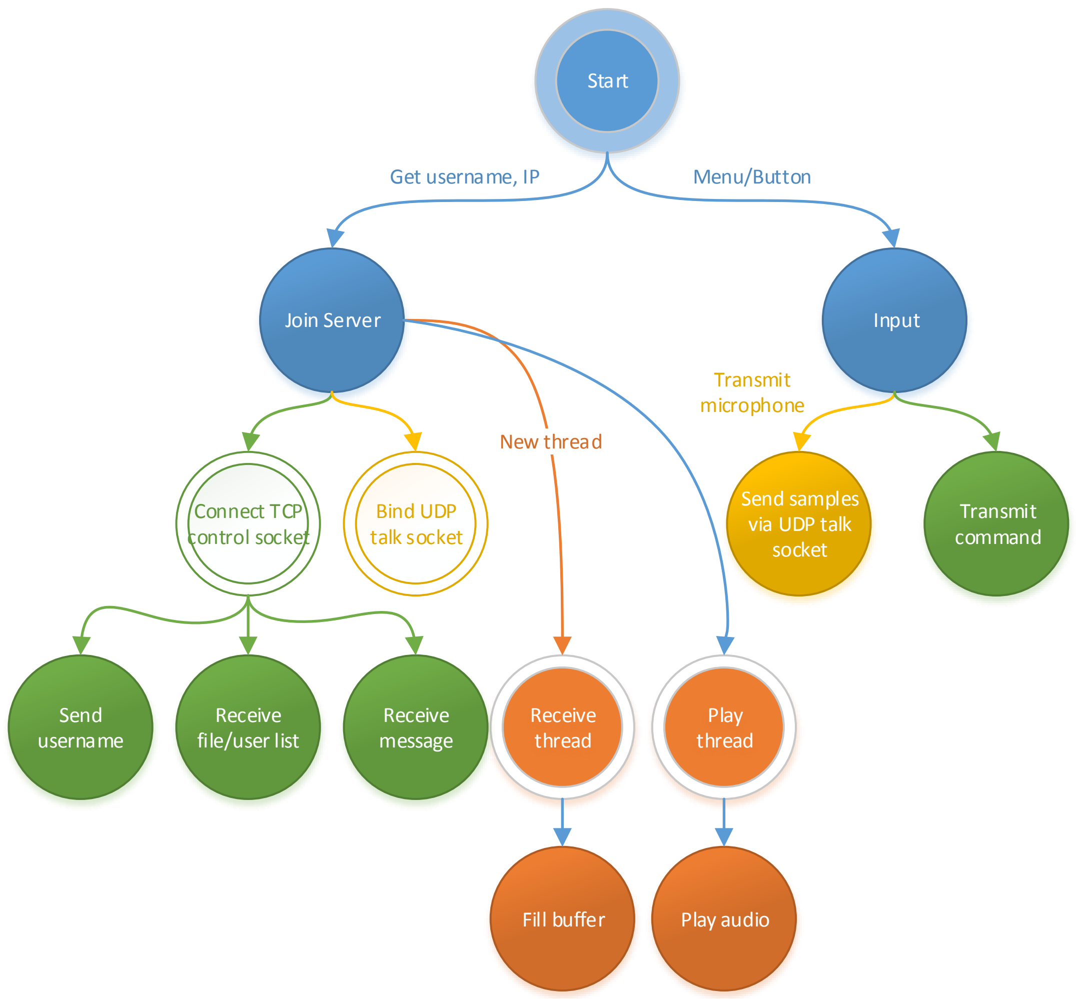
- 0x0: File list from server
- 0x1: User list from server
- 0x2: Message from server
- 0x3: File select
- 0x4: Mute
- 0x5: Playback
- 0x6: File request
- 0x7 Set username

Logical Datagram:

1 byte	4 bytes	?? bytes
Command Code	Size	Data

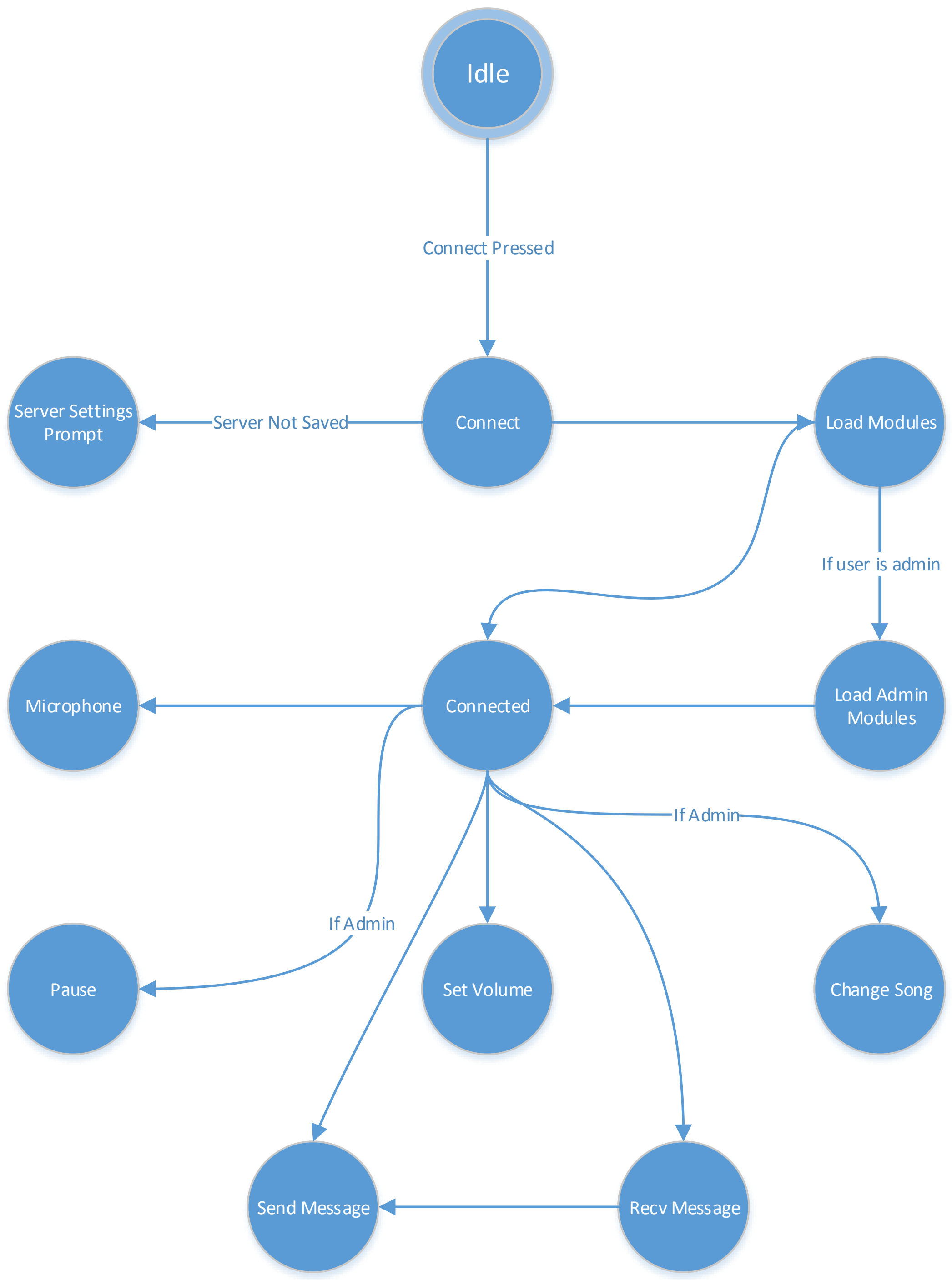
Server

March 4, 2015



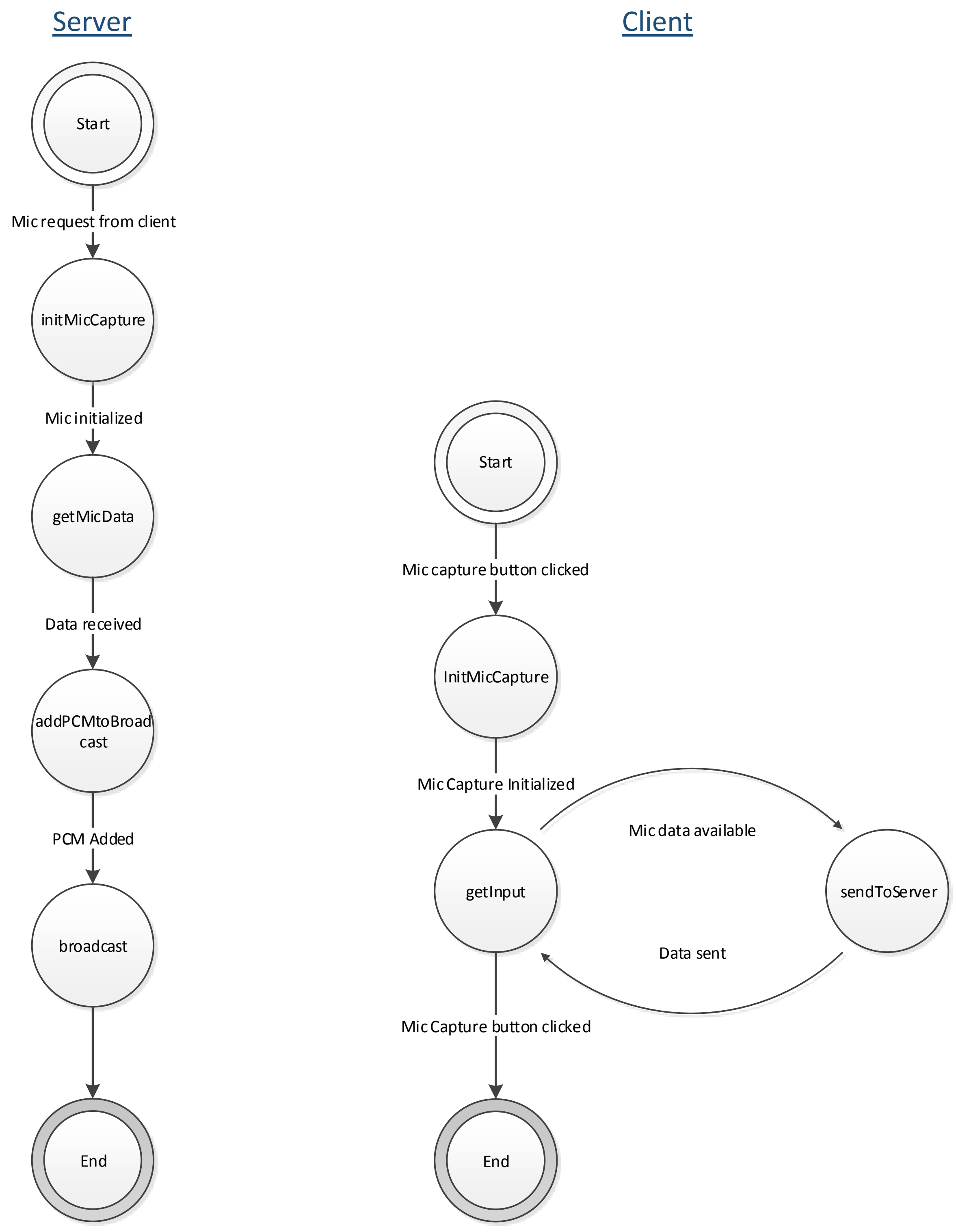
Client

March 4, 2015



Client-UI

March 4, 2015



Microphone Transmission

Pseudocode

Server:

Setup

```
{
        startup WSA
        create a list of all available files

        create a udp multicast socket and bind it
        create a tcp control socket and bind it
        create a udp microphone socket

        start the music thread

        listen on the control socket
        wait for user to press enter

        cleanup
        exit
}
```

Cleanup

```
{
        set vlc instance to stop and wait for it to stop
        de-allocate media buffers
        close sockets
        de-allocate file and user lists
}
```

March 4, 2015

Play thread

```
{
    for ever
    {
        if music list is not empty
            select next track
        otherwise
            select random track

        load media in vlc
        create new vlc instance from media

        apply vlc callback functions
        play the vlc instance

        while vlc instance not finished do nothing

        send a buffers worth of silence
    }
}
```

Play Thread - vlc pre render callback

```
{
    allocate memory chunk for rendering
}
```

March 4, 2015

Play Thread - vlc post render callback

```
{
    while chunk remaining is not zero
        if chunk remaining is greater than or equal to the size of a packet
            create a full packet from the chunk and send it
        otherwise
            create a partial packet from the chunk and send it
}
```

Accept Connection - control socket accept

```
{
    create a new user list entry for the client
    send the file list to socket
}
```

Receive Command - send file completion

```
{
    receive commands from control socket
    receive from microphone socket
}
```

Receive Command - command receive

```
{
    if partial packet
        receive until all data got
        execute command
    otherwise
        receive until got command code and size
        interpret command
```

```
                call command function
}
```

Select Next

```
{
        Add song to the end of the queue
}
```

Mute User

```
{
        Block specified user chat
}
```

Playback Control

```
{
        Interpret playback control
        If control is pause
                Pause music
        If control is play
                Play music
}
```

Request File

```
{
        Send song file through TCP socket to client that requested it.
}
```

Server Microphone

getMicData

```
{
  read Mic Data from client and put to buffer.
  When buffer is full, pass data to addPCMToBroadcast to add the data to
  the music stream
}
```

addPCMToBroadcast

```
{
    add PCM data to the music stream
}
```

broadcast

```
{
    use base broadcast function to broadcast music stream with the mic data
}
```

March 4, 2015

Client:

```
Start
{
        startup WSA
        create the gui

        create a socket to receive UDP multicast
        create a TCP socket for control connection

        create a play thread
        connect to the server via TCP
}

Play thread
{
        open the wave device using the wave callback
        allocate and zero a circular buffer
        create and prepare 3 wave headers for the circular buffer
        queue and play the 3 buffers
}

Connect TCP control socket
{
        receive commands
}
```

March 4, 2015

Connect TCP control socket - command receive

```
{
    if partial packet
        receive until all data got
        execute command
    otherwise
        receive until got command code and size
        interpret command
        call command function
}
```

Send username

```
{
    Transmit username over control socket
}
```

Receive file/user list

```
{
    Read from control socket
    If file
        Display available songs on screen
    If user list
        Display current  users on screen
}
```

Receive Message

```
{
    Display message in chat window
}
```

Receive Thread - wave callback

```
{
    re-queue the finished buffer
```

}

Client Microphone

getInput

```
{
   Get mic input as PCM and put it into send buffer.
   When buffer is ready for sending call sendToServer().
}
```

sendToServer

```
{
   send data from buffer to server.
}
```

March 4, 2015

Client-UI:

Connect

```
{
    if there is no server saved in settings.
        open up server setting dialog
    else
        connect to configured server
        if connection fails
            pop up error.
        else
            load sound chat playlist and sound visualisations
            if IP is on admin list
                load song bank tab, enable play/pause functionality
}
```

Play - onPlayPressed

```
{
    if playing music
        send server message to pause music
    else
        send server message to start playing music where it left off.
}
```

March 4, 2015

```
GUI update - sound bar graph
{
    clear space
    read sound frequencies
    set bar heights
    draw space
}
```