Comm Audio

Thomas Tallentire

Marc Rafanin

Marc Vouve

Lewis Scott

State Transition Diagrams







Server Client

 

Microphone Transmission

Psuedocode

# Server:

main

{

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

}

music 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

}

}

vlc pre render callback

{

allocate memory chunk for rendering

}

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

}

control socket accept

{

create a new user list entry for the client

send the file list to socket

}

send file completion

{

receive commands from control socket

receive from microphone socket

}

command receive

{

if partial packet

receive until all data got

execute command

otherwise

receive until got command code and size

}

initMicCapture

{

initialize connection from client to get mic data.

}

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

}

# Client:

main

{

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

}

control connect completion

{

receive commands

}

command receive

{

if partial packet

receive until all data got

execute command

otherwise

receive until got command code and size

}

wave callback

{

re-queue the finished buffer

}

initMicCapture

{

Initialize connection to server.

Initialize mic to be ready for capturing PCM data.

}

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.

}

# Client-UI:

Connect function

{

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

}

onPlayPressed

{

if playing music

send server message to pause music

else

send server message to start playing music where it left off.

}

onMicrophoneUsed

{

while microphone is being used.

use microphone connection to send raw pcm to server.

}

sound bar graph

{

clear space

read sound frequencies

set bar heights

draw space

}