Computer Networks Lab: Final Project

Yashvanth Kondi
IMT2012053

May 18, 2014

1 Functions of Each File

- **audio.h:** Contains function signatures for initializing the soundcard to either play or record sample.
- **recorder.c:** Function definition to intialize soundcard to read audio, as in audio.h.
- **playback.c:** Function definition to intialize soundcard to play audio, as in audio.h.
- **netconfig.h:** Function signatures to create sockets for a UDP server or UDP client.
- **netconfig.c:** Function definitions for netconfig.h.
- **netstream.h:** Function signatures to setup a server to stream a wave file over the network, stream what the mic picks up, and a listening client to receive audio being streamed from a source in the network.
- **netstream.c:** Function definitions for netstream.h.
- **master.c:** Contains the main function; takes user input and calls appropriate methods.

2 Calls Used

audio.h

The header `audio.h`, defined by `playback.c` and `recorder.c`, is primarily about initializing the sound card to play or record audio. In this regard, it includes function calls to open a pulse code modulated interface with the soundcard, set the rate at which samples have to be played, number of channels, buffer size to be read on each interrupt, period size, etc.

These functions are called in order to get a valid PCM handle to play/ record audio using.

netconfig.h

The `netconfig.c` file, that defines `netconfig.h` contains function calls to create a UDP socket, initialize its parameters and bind it. These functions provide a readymade way of getting a UDP server/client socket descriptor.

netstream.h

The defining file for this library, `netstream.h` basically contains a composition of the above functions. It sets up a UDP server/ client, then streams audio to/from the current device, as per the situation.

In case of streaming from the device, it initializes the soundcard to record audio, and retreives a UDP server socket descriptor. It then writes audio samples (either gotten from the mic, or a given wav file) to this socket upon being pinged by a client. In case of streaming to the device, it initializes the soundcard to play audio samples, and creates a client UDP socket. Samples are read from this socket, and then written to the audio device using the PCM handle.

In addition to using the above mentioned functions, the socket is read from/written to using `recvfrom` and `sendto` respectively. The soundcard is read from/written to from a buffer using `snd_pcm_readi` and `snd_pcm_writei` respectively (defined in alsa/asoundlib.h, along with the other soundcard-related function calls).

master.c

All this file contains is the main function. It takes user input and calls the user-defined functions from the above mentioned header files.

3 Compilation

Running the command `make master` will compile all the necessary files and produce an executable called `astream`.

NOTE: The audio card initializing function calls assume that the default audio device is 1;0, as that is the case on the author's system. As automatic detection of the correct hardware device is a tedious procedure outside the scope of this project, it has been hard-coded. On any other system, a list of audio devices may be obtained running the command `aplay -l` in the terminal. The correct one has to be noted (card x; subdevice y), then line 10 in both, `playback.c` and `recorder.c` has to be modified; the string in the `snd_pcm_open`

function call is to be changed to `"plughw:x,y"`, where x and y were the required card and subdevice numbers obtained from earlier [1].

The command `make all` will even add the `astream` executable as a system command, so that it may be accessed from anywhere in the system[2].

4 Execution

Execution of `astream` with incorrect / without any arguments will print the list of valid options.

Sample executions:

- `astream -s 100` : Serve mic output to any client requesting for the next 100 seconds.
- `astream -s test.wav` : Serve "test.wav" over network.
- `astream -l 192.168.112.53 10` : Listen to the stream from host 192.168.112.53 for the next 10 seconds.
- `astream -d 192.168.112.53 10 20` : Listen to the stream from host 192.168.112.53 for the next 10 seconds. Also serve mic output to any requesting client for the next 20 seconds.

5 Inputs During Execution

No inputs are required during execution.

6 Important Aspects

This application was created using a very low-level audio interface. Hence, while control over the interface is excellent, there are some glitches during playback. While streaming from a low-bandwidth network, for instance, the audio device will be closed and reopened many times (as keeping the filehandle active despite no new input will corrupt it and cause a buffer underflow when the audio device reads it). This may be noticeable in playback.

Also, the audio streamed is fairly high quality. The sampling frequency is hardcoded, but can be adjusted if required.

[1] Usually 0;1 , 0;0 , 1;0 or 1;1
[2] Requires sudo access