SOFT336: Technical Documentation

Prepared by: Sam Sutton

2 November 2014

Student Number: 10365944

GUI DESIGN

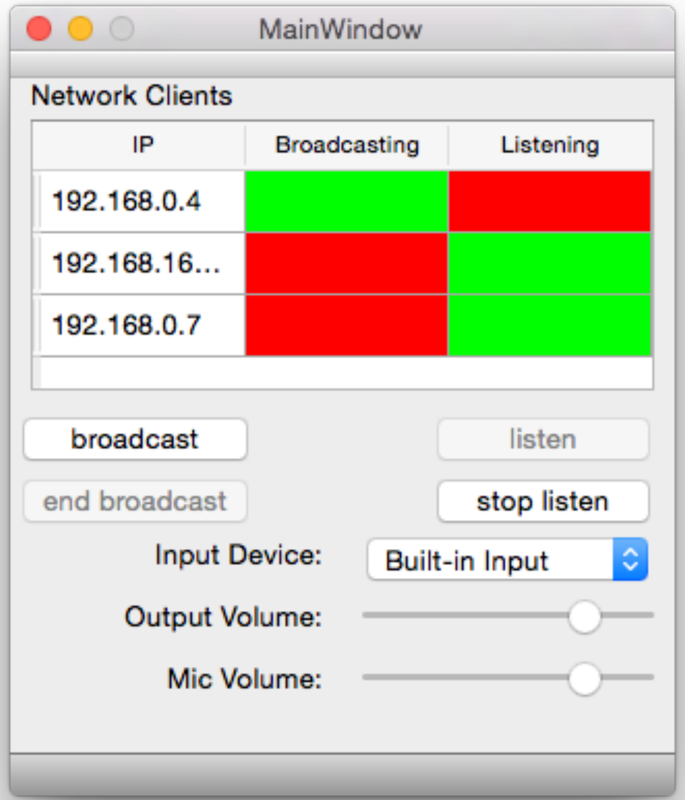
The GUI was based on a simplistic design, meant to be as simple to use as possible.

The Network Clients table displays a list of current clients on the network, and displays their status i.e. if they are broadcasting or listening.

If the user wishes to broadcast to other users on the network they click the "broadcast" button, which initiates audio recording and transmits this across the network, similarly, to stop they click the "end broadcast" button.

If the user wishes to listen to other users on the network, they click on the "listen" button, which writes the read network packets to the audio device. Users can stop this by clicking on the "stop listen".

The "Input Device" combo box can be used to change the input audio device (if there is more than one on the system). This must be initiated before the user begins broadcasting (and can be changed once they stop broadcasting).



Network Clients

IP	Broadcasting	Listening
192.168.0.4	🟩	🟥
192.168.16...	🟥	🟩
192.168.0.7	🟥	🟩

Input Device: Built-in Input

Finally, the volume sliders can be used to control the audio output and input volumes.

CODE DOCUMENTATION AND NOTES

SEE THE ATTACHED DOXYGEN DOCUMENTATION FOR THE MAIN DESCRIPTION OF THE CLASSES AND THEIR RELATIONSHIPS, INHERITANCE, ETC.

Some of the audio portions of the code are based on the audio output and input examples provided by Qt. The code has been examined heavily for any memory leaks etc, and makes use of destructors to ensure that pointers are cleaned up. However, one memory leak was identified (which is also present in the Qt examples) which was unclear on how to clean up, despite multiple attempts. Specifically, when the audio input device is re-assigned (in startDevice()) the memory does not appear to free itself.

The networking portions of the code are based loosely on the documentation from Qt. An early version of the project utilised a TCP socket in addition to UDP. Its purpose was to send broadcast and control strings. This was dropped in favour of UDP for the control strings, although this seems counter-intuitive, broadcast packets are sent regularly enough that information is not lost, similarly, the information sent in these control strings (i.e., whether the client is listening or broadcasting) is not deemed mission critical, so it doesn't matter if the occasional packet gets dropped. This works well because the server sends out a broadcast packet regularly.

The Phonon multimedia library was considered, but deemed inappropriate (not very cross compatible), and instead uses QTMultimedia libraries.

The code is well modularised, using separate classes for the audio and networking components. Makes proper use of the Q model object hierarchy, ensuring that objects are deleted appropriately etc. Uses Qt's data stream to serialise the audio information. (based from the QDataStream tutorial).

The clients list, and the associated network clients table uses a custom data model, based on QAbstractTableModel to store the clients. The code is extended from the Qt Model/View tutorial.

No external dependencies exist as no outside libraries were used, so therefore the program only requires the core Qt libraries.

A simplified high-level description of how the program works is as follows:

- A UDP server and client is started on the server.
- UDP server frequently sends broadcast messages, which also state whether the server is broadcasting or listening.
- UDP client receives these messages, adds client to the client list if if doesn't exist, or updates it status if it does.
- When broadcasting, UDP server reads audio from input device, and sends this to all clients in the client list.
- When listening, UDP client receives audio, and writes this to the output device.