CP252 Final Project

Research Team Project and Protocol Specification

**Group Members**

 Matt Zawojski

 Forrest Surprenant

 Zach Boynton

 Derek Lamonica

 Paul Ouellette

**Record of Changes**

4/4/13 – Initial document created.

4/24/2013 – Document Updated.

**Functional Specifications**

**Overview**

The research team will develop a protocol that can be used for the transmission of audio data. The team will also create a library that implements the protocol and provides additional abilities to read audio and playback audio.

**Major Features**

* Ability to read audio chunks from a source (microphone or file)
* Compress audio chunks using GSM6.10 compression
* Ability to play audio from a certain position in the file
* Ability to send those audio chunks over the network in UDP packets
* Ability to receive and buffer audio chunks in UDP packets
* Ability to playback received audio chunks

**Major Components**

The library created by the research team will contain several components.

Microphone Reader

 Microphone reader objects will have the ability to record incoming audio from the microphone hooked to the system. The audio captured from the microphone will be compressed and sent out over the network as close to real time as possible. As the audio comes in, it is taken in 1600 byte chunks which are compressed and packed into UDP packets which are then sent over the network.

File Reader

 File reader objects will have the ability to read the audio data out of a wave file on the system. The audio sample data is read from the file in 8000 byte chunks of audio data, which are then compressed and packed into UDP packets, and then sent over the network.

Audio Player

 Audio player objects will have the ability to playback the audio chunks received in the UDP packets generated by microphone or file reader objects. The audio player will decompress the data and keep a buffer of several chunks to ensure smooth playback.

Playback UI

 The playback UI will be a Graphical User Interface (GUI) that can be used with the audio player. When the GUI is used with an audio player object that is receiving audio from a file source, the duration of the file, the file name, and the sample rate will be known. This will allow the GUI to display a playback bar, file duration and current playback time.

**Software Architecture**

 The library will be built with code written in C#. We will use Microsoft Visual Studio 2010 with version 4 of the .Net Framework. The created C# code will make use of the NAudio C# audio library.

**The User Experience**

 We will assume that the user of the library already has a working client server application and is handling any other processes such as token passing that are needed. The user adds the DLL created by the research team to their project and uses the objects and methods of the library to add whatever audio functionality is needed to their application.

 The user will create microphone or file reader objects depending on the application on the sender side of the application. On the receiver side of the application the user will create audio player objects.

**Future Features**

* Implement a seek function from the player by means of the playback bar in the GUI.
* Add support for multiple file formats.

**Conflicts and communication problems**

This section will list possible things that can go wrong and the way in which those problems will be handled.

|  |  |
| --- | --- |
| **Problem** | **How it will be handled** |
| The client stops receiving data packets, but never receives an end of transmission packet. | The client will assume end of transmission if no data packets are received for a certain duration. |
| The data packets are not arriving at exactly the right time for real time playback on the receiver side. | The built in buffer in the audio player should compensate for small inconsistancies in arrival time of the packets. Any delay that would be too long for the buffer to handle would cause the client to assume end of transmission . |

**Technical**

**Design tools and environment**

* Operating System: Windows 7
* IDE: Microsoft Visual Studio 2010
* Programming Language: C#
* Dependencies:
	+ NAudio C# audio library

**Protocol Definition**

File Reader - WaveFileManager

When a file reader object is created it initializes variables

Files will be 8 bit GSM 6.10 compressed wave files sampled at 8000Hz.

The file reader will read the contents of the file into a queue from a starting position, default start position is zero.

WaveFileManager.ReadWaveFile() method tell the manager to open a file and takes 3 parameters.

1. string filePathIn – full path of the file
2. ushort startPosition – The position in the file, in tenths of seconds, to start reading from
3. bool isCompression – Is the data going to be compressed.
* The contents of the file will be separated into chunks.
	+ If the file is being compressed the chunk size will be 8000 bytes, or 1 full second of audio data
	+ If there is no compression the chunk size will be 800 bytes, or 1/10 of a second of audio.

UDPSender

Handles sending the audio data

* First a packet announcing a new stream will be sent to the Destination.
* Packet Format:
	+ 1 byte SOH
	+ 2 bytes to indicate uncompressed chunk size
	+ 2 bytes to indicate duration of audio in tenths of seconds (only used for file source)
	+ 2 bytes to indicate the sample rate
	+ 1 byte to indicate the channels
	+ 1 byte to indicate the bits per sample
	+ 32 bytes for the file name
* The UDPSender then waits for an ACK response from the PlaybackObject
* If ACK is received then transmission begins
* Packet Format:
	+ 800 bytes of audio data for uncompressed data
	+ 1625 bytes for compressed audio data from a file
	+ 325 bytes for compressed audio data from the microphone

MicroManager

When a microphone reader object is created it is given three parameters.

1. Int Rate – sample rate of the audio
2. Int bits – number of bits taken per sample
3. Int channels – number of channels used

The NAudio library will be used to read the audio from the microphone in the form of 8 bit PCM uncompressed audio sampled at 8000Hz.

MicroManager.StartRecording() method will cause the object to start reading from the microphone and transmitting the data to the Destination.

MicroManager.StartRecording() method will cause the object to stop reading and transmitting audio from the microphone.

Audio Player

When an audio player object is created it is given one parameter.

1. Audio Duration – A value of 0 will tell the player that we are playing a live source (microphone) and the duration is unknown. Any other valid positive duration indicates we are playing from a file source.

Upon successful creation of an audio player object an ACK packet is sent back to the Source.

The player will then start receiving data packets.

* Incoming packets are placed into a buffer.
* Once the buffer contains 1 full second of audio playback can begin.
	+ The audio format will always be the same 8 bit GSM 6.10 @ 8000Hz so header information is generated programmatically
	+ Audio data is pulled from the buffer to be played while the incoming packets continue to be added
	+ Playback continues until an End of Transmission packet is received or no new packets are received for the timeout duration

Port numbers

Audio transmission will take place on UDP port 54321.