Literature Review

Submitted in partial fulfilment of the requirements of the degree

Bachelor of Science (Honours)
Computer Science

Rhodes University - Grahamstowm

Mitchell Hedges

2012

Contents

1. Introduction

The field of sound control is a popular area of research as there are many different ways that this may be done. It was conventionally done by having fixed physical point-to-point cabling in which signals could be sent from a source to a destination. As this is progressing into digital control, there is software which can be used to control Ethernet AVB (Audio/Video Bridged) devices from a GUI (Graphical User Interface). This paper explores various components that can be used in unison to control devices from a 3D (3 Dimensional) GUI. There are a total of four different fields that will be looked at in-depth, and reason will be given such as to why these particular ones have been chosen. These are sound system control protocols, graphical sound installation, Google Sketchup, and linking Visual C++ to Google Sketchup. Alternative methods will also be looked at with their respective advantages and disadvantages with reason given as to why they have not been chosen for this project.

2. Sound System Control Protocols

This section will look at three different sound control protocols for monitoring and controlling distributed audio devices. These three protocols (in respective order) are the XFN, Harman HiQNet, and IEEE 1722.1 AVDECC.

2.1 Harman Pro HiQNet

The HiQNet protocol, developed by the Harman group, is aimed at a single solution for all Harman networked devices based on standard networking technologies [8]. It is controlled by a single application, System Architect, which has been developed to the specification of this protocol in order to optimize both tour sound and sound installation environments [10]. This protocol has a tiered approach with the top level representing the actual device (node) where each node must the have at least one virtual device; this acts as a node manager.
Each virtual device (as well as the objects in the virtual devices) can contain objects and parameters which make up a useful unit which allows designers a convenient method of segmenting the product. At each level in the hierarchy there are also attributes, which are member variables containing data about the virtual device, parameters or objects. Attributes can be classed into three

different types, which are static, instance, or instance & dynamic. Static variables contain the same values across all the different devices of the same model. Instance variables are initialized on start up and have their values set, while the instance & dynamic are set on start up and can have their values changed throughout their lifetime.
Each item (Virtual device, object, or parameter) has its own unique Class ID and Class Name. This allows each one to be uniquely identified in order for the designer to work with them individually [23].

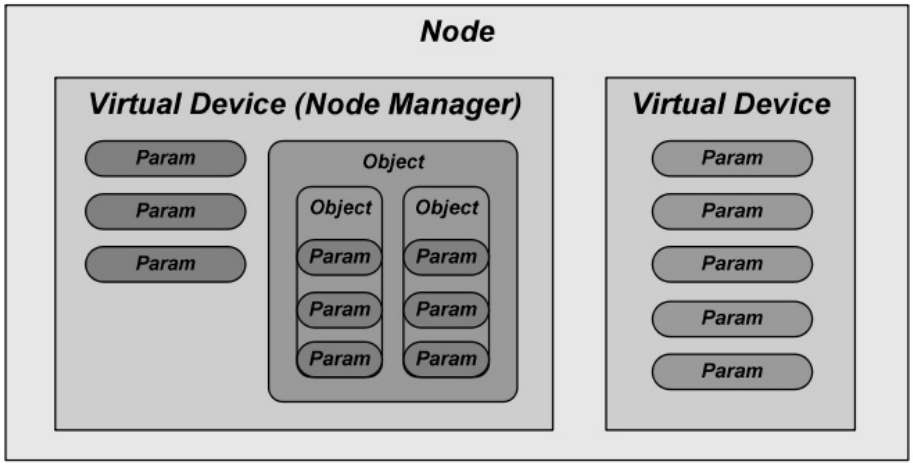

Figure 1: HiQNet Device Architecture

Objects in HiQNet are collections of parameters or other objects conveniently grouped together. The smallest modifiable parameter is actually held within a HiQNet parameter as these contain various variables of the audio objects such as frequencies or fader values. The number of parameters contained within the product will differ depending on the complexity of the product, products such as mixing consoles will have many parameters while the simpler products will only have a few. These parameters support several primitive data types and when sending a HiQNet message, it is important to use the appropriate data format [23].

2.2 XFN

The XFN (Cross Fire Network) Protocol is an IP based peer-to-peer network layer [26] command and control protocol [17] [21]. This protocol has two broad functionalities which are connection management, and monitoring and controlling devices [25].
A device must satisfy three prerequisites in order to use the XFN protocol, which are as follows:

- It must have at least one fully functional networking interface
- It must conform to the TCP/IP stack
- It must be capable of hosting an implementation of the stack of the device

Each XFN device within a network is represented as a hierarchical, seven level tree of device parameters. Due to the structure of the tree, each parameter in an XFN network must be addressed via the fixed seven level address. Each level has a unique value assigned to it, with an associated alias in order to describe its functionality [21].
The seven levels of the hierarchy are described below, in descending order.
1. Section Block: Any single device is may be a composition of various sections such as output or patching [21]. This part of the hierarchy is used for identifying the section in which parameters are contained.
2. Section Type: This identifies the distinct type of the section. The output of a mixing console may have digital or analogue output so it is important to distinguish the type [21]
3. Section Number: The section number is used to identify specific instances of the section [21]. A mixing console may have an output section to various channels that need to be individually addressed, which is what the section number is used for [25].
4. Parameter Block: The parameter block is used to describe which section the parameter is found in. This section may be less critical for simpler devices which contain a single block of various parameters [25]
5. Parameter Block Index: The parameter block index is used to refer to different subsections of the parameter block [21]. This may follow the *Parameter Block* closely and is also negligible for simpler devices [25].
6. Parameter Type: The parameter type describes the type of parameter

being addressed [21]. There can be many different types of parameters within a single device such as frequency or gain, which needs to be addressed individually [25].

7. Parameter Index: The parameter index is the lowest level in the hierarchy, which addresses the different parameters individually in their lowest level of grouping.

2.2.1 XFN Messages

Any XFN devices are allowed to send and receive XFN messages between their networked peers. The XFN stack provides facilities for the transmission and receipt of these messages according to the predefined XFN specification [19]. Every XFN message can be classed as either a request or a response. A request is a message sent from one XFN device to another requesting an action, while the response is a "reply" to a request message [25]. Not every request message has a response; this is purely determined by the nature of the message. In some of the XFN messages such as the *setRemoteParamValue_nonb_fdb* [1], there is boolean parameter named *responseRequired* which may be set to true or false depending on whether the sender requires a response or not [19].
Contained within these messages is the *Command Executive* and *Command Qualifier* [25] [21]. The *Executive* describes the fundamental nature of the message, while the *Qualifier* allows it to direct at a certain attribute of a parameter [21].

2.3 AVDECC

The Institute of Electrical and Electronics Engineers (IEEE) 1722.1 specifies protocols for device management on an Ethernet AVB network which use Audio Video Transport Protocol (IEEE 1722 [17]) for streaming [27]. This protocol is named Audio/Video Discovery Enumeration, Connection management and Control (AVDECC). Devices that make use of the ADVECC protocol are known as ADVECC devices, and are thus capable of sending AVDECC compliant layer 2 AVTP control messages [27].

[1] A non blocking method which sends a request to set a remote parameter value [19]

AVDECC devices can be classed according to their functionality, as one or more of the following [14]:
Entity: An end station which allows the facilitation of transmission and receipt of AVDECC messages within an Ethernet AVB network.
Controller: An AVDECC entity which initiates message exchange between other entities. There may be more than one controller on a network, however, the *LOCK_ENTITY* command must be called for atomic operations.
Listener: An AVDECC entity which allows transmission and receipt of AVDECC messages and is also a listener on the AVB network.
Proxy: An AVDECC controller which forwards AVDECC messages. This can be done between both network layer 2 and network layer 3 for completeness.
Talker: An AVDECC entity which allows transmission and receipt of AVDECC messages and is also a talker on the AVB network.

2.3.1 Sub Protocols

The AVDECC protocol is comprised of three sub-protocols (where each is responsible for the functions suggested in the names) [27]:

- AVDECC Discovery Protocol (ADP)
- AVDECC Connection Management Protocol (ACMP)
- AVDECC Enumeration and Control Protocol (AECP)

The AVDECC Discovery Protocol (ADP) is a Layer 2 protocol which is responsible for the discovery of networked, AVDECC compliant devices. This dynamically identifies the different AVDECC devices as they are added or removed from the network. It uses three different notification messages to indicate whether entities are available, unavailable, or searching for available [14].
The AVDECC Enumeration and Control Protocol (AECP) is responsible for controlling parameters of an entity, from another entity. Controls for this protocol would include setting or getting values of parameters such as volume or muting. A requirement for this is that the device receiving the control message is able to understand it [27].
The AVDECC Connection Management Protocol (ACMP) is responsible for the management of connections between the sources and sinks [14]. This is achieved

by the transfer of single packets known as AVDECC Connection Management Protocol Data Units (ACMPDU). These structures utilize the feature of time-outs and contain a sequence identification for reliability purposes. [27]

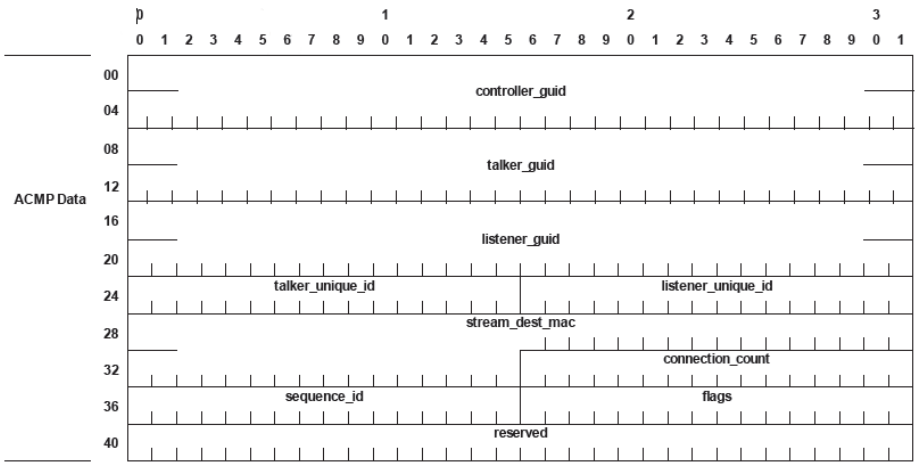

Figure 2: ACMPDU Structure [14]

2.3.2 WinPcap

WinPcap (**Win**dows **P**acket **cap**ture) is a standard tool for link layer (layer 2) packet access in windows environments [29]. This library is able to capture "raw" packet data, data which is usually accessed from higher levels in which the operating systems take care of the lower level data processing. This is useful as it is entirely independent from the host protocols, as it simply "sniffs" the packets as they pass [29].

The Pcap library would be useful in the field of digital sound as it can enumerate and gather information about devices on the network. It is particularly needed for controlling devices that conform to the IEE 1722.1 standard as they both operate on the link layer. The Pcap library is natively used in C [29] but can also be used in Visual C++ within Microsoft Visual Studio 2010 [13]. It requires the installed Pcap library with the `"#include <pcap.h>"` in the code header of the file in order to successfully compile [13].

3. Graphical Sound Installation

3.1 UNOS Creator

UNOS Creator is sound installation software developed by Universal Media Access Networks (UMAN). It's core functionality is the ability to create custom interfaces by drag and drop instead of writing lines of code for an interface [16]. UNOS Creator provides the user with a desk item editing capability, which allows the user to specify various images for the different desk item controls. Desk item controls are controls that are found on a desk top such as sliders and meters [20]. Each desk item has a parameter associated with it which is known as a "control parameter", this effectively controls the value associated with the desk item [20].

The UNOS Creator GUI editor can be accessed from the "tools" menu within UNOS Vision. UNOS Vision displays different devices for the various subnet configurations. The devices can then be selected to display the matrix mixer which is responsible for control and patching of the devices [20].

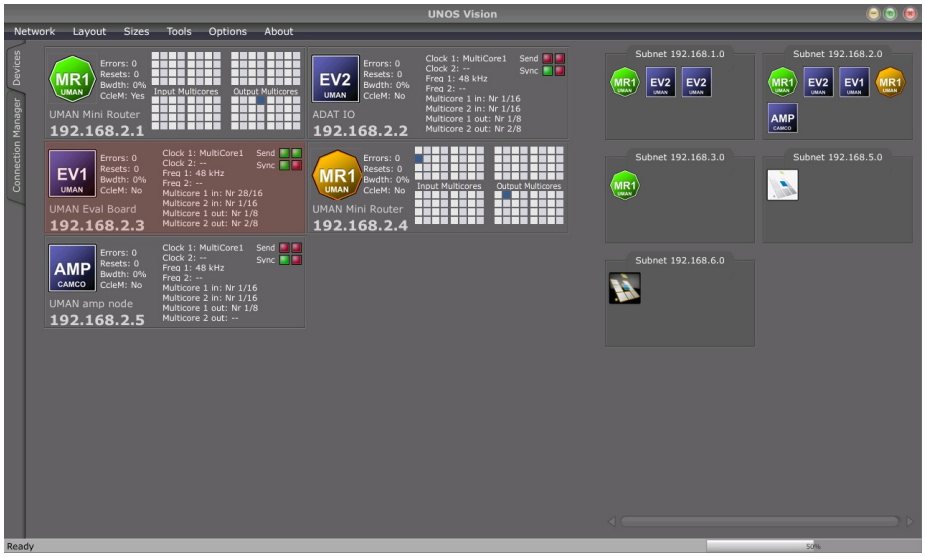

Figure 3: Small network configuration as it appears in UNOS Vision [20]

3.2 Harman System Architect

System Architect is the software application developed by the Harman group to control HiQNet systems. It is the user interface for the HiQNet protocol, discussed earlier. Sound control can take place after setting the software up according to your network configuration. The network can be set up to be used in four steps as shown below [22].

3.2.1 Defining the venue

The user is required to graphically design the venue for the sound installation. This can be done by using the preset shapes provided by System Architect which can be shaped into a realistic representation of the venue. A crucial feature provided whilst doing this is the ability to have a background image whilst creating the layout, effectively allowing the user to "trace" the layout of the venue.

3.2.2 Adding devices by their location

After the venue design is completed, the user will then add the distributed audio devices to the venue. These can be specifically placed as they will correspond to their physical location within the venue. The list of devices provided by System Architect is preset and includes various devices by AKG, BSS, Crown, dbx, JBL, and Lexicon [24]. The physical layout of the devices is important as the interface will indicate which rooms have problems (if they occur).

3.2.3 Associating the devices by their location

Although amplifiers have only one physical location, they typically end where the loudspeakers begin [22]. This stage of the setup involves associating the user specifying which amplifiers are connected to speakers in different sections of the venue. System Architect allows a drag and drop technique to allow the user to specify which amplifier channels to connect on installation.

3.2.4 Routing the audio

Audio routing is a highly complex step within general sound installation. System Architect has aimed to simplify it by allowing the user to route to specific zones

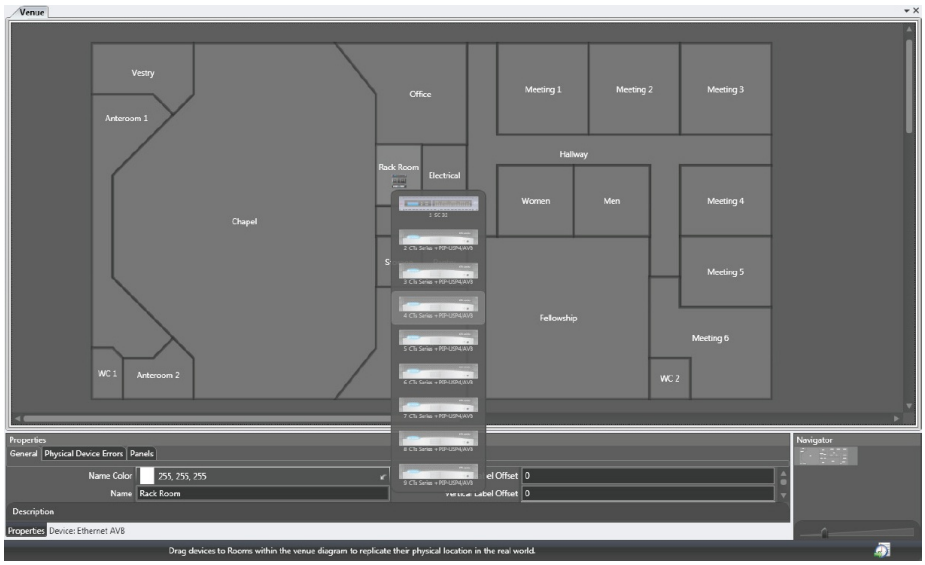

Figure 4: Placing devices in a custom made venue with System Architect [22]

within the venue, instead of specific devices. This is a high level abstraction as the "behind the scenes" device routing is all taken care of.

3.3 BSS London Architect

London Architect is software which allows graphical sound installation, developed by BSS Audio. BSS Audio is part of the Harman group and also uses the HiQNet protocol [9]. London Architect is a powerful tool which makes use of a user orientated drag and drop GUI. It provides standard sets of audio devices for the user to add to the interface and simply connect to other devices. This software provides a wide range of functions will allow scheduling as well as automatic system adjustments according to the state of the devices [9].
Despite the difference between the interface appearances, this software is in many ways similar to System Architect. Despite the similarities, there is one major difference in that System architect allows connection between zones, while London Architect allows explicit connection between devices. The London Architect device interface shows all the input and output jacks for the list of preset devices which allow the user to specify exactly which inputs and outputs are connected between devices. System Architect does all this for the user "behind

the scenes", which is ideal from a user friendly point of view, but essentially gives less direct control over the devices.

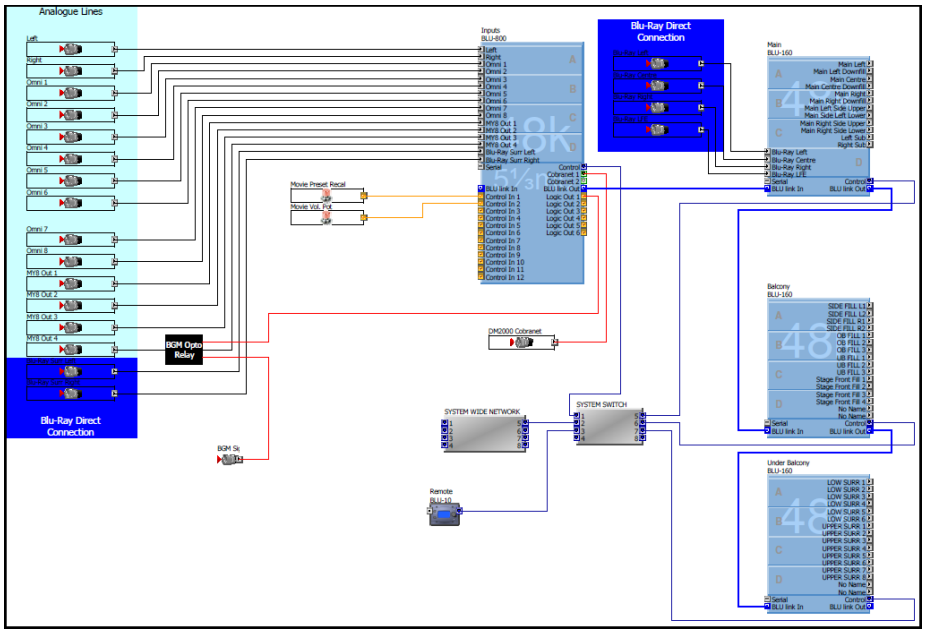

Figure 5: London Architect user interface, showing connections between various devices

4. Google Sketchup

Google Sketchup is an open source software tool developed by Google. It is a tool that allows 3D construction of objects using a standard set of tools provided. These tools are categorized into principle, drawing, modification, camera, construction, walkthrough, sandbox, and 3D Warehouse. By using these tools, one is able to create any possible 3D structure within the 3D interface of Sketchup. [12]. Plugins for Sketchup can be written to provide a wide range of functions and adapt the user interface to suit the user.

4.1 Ruby

Ruby is a functional programming language created by Yukihiro Matsumoto as a blend of different languages (Perl, Smalltalk, Eiffel, Ada, and Lisp) in

order to create a programming language that balanced functional and imperative programming [1]. His main aim was to develop a language more powerful than Perl, yet more object orientated than Python. Everything in Ruby is treated as an object, with its own properties and actions. Encapsulating everything into the object type is one of ruby's main ease of use objectives, as everything is treated in a similar fashion with a standard set of rules being followed.

4.1.1 Ruby in Sketchup

Sketchup has not only the capabilities of 3D imaging, it also has an embedded programming feature, the Ruby API (Application Programming Interface). Ruby is associated with Sketchup in the form of "plugins", which are loaded on start up from the "Sketchup/Plugins" folder. Plugins extend the functionality of Sketchup tremendously as they can perform actions such as adding tools, simplify multi-step operations, and otherwise improve the way you work with Sketchup [15]. Ruby scripts can be written in any basic text editing program, and saved in the "plugins" folder as ".rb" files. These files are then loaded and the appropriate actions will take place whether they are function calls, or user interface customization. The functions within the script can then be called from various different user defined places, or the standard Ruby console which is provided in Sketchup. [2]

4.2 User Interface

The "User Interface" or UI classes contains various methods which can add to the user interface, which are called upon start up. They are able to create and manage various elements such as drop down menus, timers, message boxes, input boxes, panels, and URLs [7]. Custom submenus and items can be added to existing menus by assigning a name to a menu, and then adding it via the built in methods. These must be added to the Sketchup menus, which are not all visible on start up (such as the plugins menu). The menus available in Sketchup are "File", "Edit", "View", "Camera", "Create", "Tools", "Page", "Window", "Plugins", and "Help". [7]

Consider the example below (Adapted from [7], [6])

```
plugins_menu = UI.menu("Plugins")
```

```
submenuCamera = plugins_menu.add_submenu("Cameras")
submenuCamera.add_item("View Camera Properties") { cameraStats }
```

This section of code initializes a variable to the "Plugins" menu, add a submenu called "Cameras", and then add an item called "View Camera Properties". This example includes two of the three methods for editing a menu, the other is "add_separator" which simply adds a menu separator to the given menu or submenu. A function called "cameraStats" is then bound to that item and called when the user clicks on the item. This function will be written elsewhere in the script, and can also be called from other menu items, as well as the console window.

4.3 Cameras

The camera class in Sketchup is the class in which the methods are contained that manipulate the camera. The camera is the current "point of view" that the user is observing the 3D model from. Cameras are able to be moved around the 3D environment in two ways, the first is interacting with the 3D environment by moving it around using the mouse or tools, the second is manipulating it's attributes from a Ruby function. A camera object has 3 attributes which have to be defined in order for a fully functional camera to be created, the "Eye", "Target", and "Up Vector". The "Eye" is the physical position of the camera within the 3D environment, given as a distance from the origin in the form of a 3D co-ordinate. The "Target" is a specific co-ordinate within the environment that is in the center of the camera's vision. The "Up Vector" is a vector that describes the direction in which the top of the camera is facing. This is comprised of 3 vectors (x, y, z) in which each vector can be described as an element given by u in $\{u \mid u \in \mathbb{R}, -1 \leq u \leq 1\}$ [3]

Consider the example below (Adapted from [3])

```
eye = [X1, Y1, Z1]
target = [X2, Y2, Z2]
up = [X3, Y3, Z3]
my_camera = Sketchup::Camera.new eye, target, up
view = Sketchup.active_model.active_view
view.camera = my_camera
```

Presuming that [X1..Z3] have been previously defined somewhere in the program, this section of code creates 3 array type data structures, each containing 3 elements for the "eye", "target", and "up" (where X3, Y3, and Z3 must obey the rule mentioned above). The current Sketchup camera is then changed to the newly created camera, effectively "moving" the camera to the new point of view created by the user [3].

4.4 Entities

The entities class serves as a container class for all the entities in a Sketchup 3D model [4]. This container is continuously updated as more entities are added and/or removed from the model. A single entity is a member of the base class "Entity" which is anything that is contained in the model such as edges, planes, component instances, and groups [5]. Entities can be referenced from the container class and actions can then be performed on them which will be methods from the "Entity" class (It is important to distinguish between the "Entities" and "Entity" classes as they are separate).

Consider the example below (Adapted from [4], [5])

```
entities = Sketchup.active_model.entities
entitycur = entities[0]
puts entitycur.typename
puts entities.count
```

The code above makes use of both the "Entity" and "Entities" class. It assigns the current active model's entities to an array called "entities", and then assigns a current entity named "entitycur" to the first entity in the model (entity at position 0 of the entity container array construct). The entity's type will then be printed out in the form "Face", "Edge", "ComponentInstance", etc., along with the count of the total entities in the model.

5. Linking Visual C++ to Google Sketchup

The controlling aspect of this project will be based on the .NET framework's Visual C++ platform. This programming is done through Visual Studio, a

development program which provides a user friendly Integrated Development Environment (IDE) as well as coding interface [11]. This will have to communicate with Sketchup in order to send and receive the control protocol messages from the 3D user interface to the control interface. This will be achieved by creating a parallel running Transmission Control Protocol (TCP) socket server within the C++ application which will be responsible for the transmission and receipt of messages from the 3D user interface. TCP is connection oriented layer 3 protocol which provides a full duplex service between two processes [26]

5.1 Socket programming in Visual C++ and Sketchup

The server structure will be set up on the Visual C++ side of the application. This essentially contains methods for listening in on a socket for connections from other programs. Once the data stream is transmitting and receiving, there is no fundamental difference between the client and server [26].
Setting up a socket server in most languages, follows the process of 4 steps [28]

1. Create a socket
2. Bind the socket to a network interface
3. Listen on the socket
4. Accept connections

The creation of the socket will create a TCP/IPv4 socket in which the program will listen on. The 3D interface will provide a method in order to connect to this socket to transfer data. The listening method will have to be run in parallel to the main program as this will have to keep listening in case Sketchup is disconnected and needs to reconnect. This will have to be synchronized with the main thread so that no transmission occurs whilst the socket is not connected to a client [25].
Sketchup has a socket class named "SKSocket" which is officially unsupported and undocumented [18]. This class has 4 methods which allow basic read and write socket operations. These methods are *connect, disconnect, add_socket_listener* and *write* [18]. These methods will be bound to calls from the user interface in order to perform the appropriate control actions with callbacks to the main interface.

Bibliography

[1] About Ruby. Online. Available from: http://www.ruby-lang.org/en/about/.

[2] Google Sketchup Ruby API. Online. Available from: https://developers.google.com/sketchup/.

[3] Google Sketchup Ruby API - Cameras. Online. Available from: https://developers.google.com/sketchup/docs/ourdoc/camera.

[4] Google Sketchup Ruby API - Entities. Online. Available from: https://developers.google.com/sketchup/docs/ourdoc/entities.

[5] Google Sketchup Ruby API - Entity. Online. Available from: https://developers.google.com/sketchup/docs/ourdoc/entity#entityID.

[6] Google Sketchup Ruby API - Menu. Online. Available from: https://developers.google.com/sketchup/docs/ourdoc/menu#add_submenu.

[7] Google Sketchup Ruby API - UI. Online. Available from: https://developers.google.com/sketchup/docs/ourdoc/ui.

[8] HiQnet Protocol and Harman System Architect. Online. Available from: http://www.akg.com/site/products/powerslave,id,948,pid,948,nodded,2,_language,EN.html.

[9] About London Architect. Online, 2010. Available from: http://www.bssaudio.com/LA_Features.php.

[10] What is System Architect? Online, 2010. Available from: http://hiqnet.harmanpro.com/about/.

[11] Microsoft Visual Studio 2010 Products. Online, 2011. Available from: http://www.microsoft.com/visualstudio/en-za/products.

[12] Sketchup Tools. Online, February 2011. Available from: http://support.google.com/sketchup/bin/answer.py?hl=en&answer=73815.

[13] Visual c++ 2010 express and winpcap compiling issues. Online, January 2011. Available from: http://social.msdn.microsoft.com/Forums/sk/Vsexpressvc/thread/22fd5e36-084f-4a3d-896b-4aed820066e2.

[14] Draft standard for standard device discovery, connection management and control protocol for ieee 1722 based devices - draft d19. Online, April 2012.

[15] Ruby Scripts. Online, 2012. Available from: http://sketchup.google.com/intl/en/download/rubyscripts.html.

[16] UNOS Creator. Online, 2012. Available from: http://www.unosnet.com/unosnet/index.php/unos-core.html.

[17] Dibley, J. Audio networks, March-April 2012.

[18] Foltz, J. Class: SKSocket. Online, May 2012. Available from: http://rubydoc.info/github/jimfoltz/SketchUp-Ruby-API-Doc/master/SKSocket.

[19] Foss, R. *XFN Application Programming Interface*. Universal Media Access Networks, November 2009.

[20] Foss, R. *UNOS Creator User Manual*. UMAN, February 2011.

[21] Foulkes, P. *An investigation into the control of audio streaming across networks having diverse quality of service mechanisms*. PhD thesis, Rhodes, September 2011.

[22] Harman. *HIQNet System Architect 2 Workflow Overview*.

[23] Harman. *HiQnet Third Party Programmer Documentation*. Harman, December 2010.

[24] Harman. HiQNet System Architect. Online, 2012. Available from: http://hiqnet.harmanpro.com/general/system_architect.

[25] Haw, S. The x170 protocol as a vehicle for 3d sound control. Thesis, November 2011.

[26] Kurose, J., and Ross, K. *Computer Networking*, vol. 5. Pearson, 2010.

[27] Otten, F. *Overview of Sound System Control and X-170*. PhD thesis, Rhodes, 2012.

[28] Wilson, J. Socket Programming with MFC (Part 1). Online, November 2005. Available from: `http://www.codeproject.com/Articles/12209/Socket-Programming-with-MFC-Part-1`.

[29] WinPcap. WinPcap Documentation. Online, 2008. Available from: `http://www.winpcap.org/`.

Plan of action

Date	**Event**
28th May	Literature review and Plan of Action due in.
4th - 18th June	Examinations
19th - 22nd June	Vac Project work
30th June - 7th July	Away (SAU Squash)
8th July - 16th July	Vac Project work
16th - 20th July	Fieldtrip
24th/31st July & 7th August	Second oral presentations
23rd July - 13th August	Advanced Architecture and Graphics modules
14th August	Begin main programming component
20th August - 17th September	Real Time Multimedia Module
29th-31st October	Third oral presentations
2nd November	**PROJECT DUE**
5th November	Website due
21st/22nd November	Final research oral examinations

During the June/July vac I will be in Grahamstown for a total of about 10 days, which I will fully dedicate to project work. During this time I will try and complete the majority of the underlying research before the time comes to start the main programming component. By the time I am busy with the main component, I will have completed 5/6 modules for the year. The last module will be Real Time Multimedia which I think will be useful to be doing whilst doing the main component of the project as they will most likely compliment each other. At the moment I feel that I am a bit behind as things have come to a halt with exams and the literature review due in. I plan to be back to the original time frame after the July vac.