Connection Management Applications for High-Speed Audio Networking

A thesis submitted in fulfilment of the requirements for the degree of MASTER OF SCIENCE in Computer Science

At

RHODES UNIVERSITY

By

PHATHISILE SIBANDA

December 2007

RHODES UNIVERSITY

Where leaders learn

ABSTRACT

Traditionally, connection management applications (referred to as *patchbays*) for high-speed audio networking, are predominantly developed using third-generation languages such as C, C# and C++. Due to the rapid increase in distributed audio/video network usage in the world today, connection management applications that control signal routing over these networks have also evolved in complexity to accommodate more functionality. As the result, high-speed audio networking application developers require a tool that will enable them to develop complex connection management applications easily and within the shortest possible time. In addition, this tool should provide them with the reliability and flexibility required to develop applications controlling signal routing in networks carrying real-time data. High-speed audio networks are used for various purposes that include audio/video production and broadcasting. This investigation evaluates the possibility of using Adobe Flash Professional 8, using ActionScript 2.0, for developing connection management applications. Three patchbays, namely the Broadcast patchbay, the Project studio patchbay, and the Hospitality/Convention Centre patchbay were developed and tested for connection management in three sound installation networks, namely the Broadcast network, the Project studio network, and the Hospitality/Convention Centre network. Findings indicate that complex connection management applications can effectively be implemented using the Adobe Flash IDE and ActionScript 2.0.

ACKNOWLEDGEMENTS

I would like to express my utmost gratitude to everyone who honestly contributed to the successful completion of this investigation.

Firstly, I would like to thank God for the strength he has given me throughout this investigation. I thank Him for having given me the opportunity to work with all the people at Rhodes University who I must thank for being such an inspiration to me, I am truly grateful.

Secondly, I would like to thank my supervisors **Professor Richard Foss** and **Professor Greg Foster** for whom I have great respect. They were always ready to help and give me technical or moral support throughout the investigation in spite of their busy schedules. *Thank you, may God bless you abundantly.*

This project could not have succeeded without the assistance from the Audio Engineering Group (AEG) members at Rhodes University Computer Science Department. I also thank **John Ebden** and my sister **Sibusisiwe Sibanda** for proof reading this thesis work, *Thanks and may God Bless you*.

I would also like to thank **ANDREW MELLON PRESTIGIOUS SCHOLARSHIP** for their financial support, which made it possible for me to study towards my MSc degree at Rhodes University.

Lastly, I would like to take this opportunity to acknowledge the financial support from Telkom SA, Business Connexion, Converse SA, Verso Technologies, Stortech, Tellabs, Mars Technologies, Amatole Telecommunication Services, Bright Ideas Project 39 and THRIP through the Telkom Centre of Excellence at Rhodes University which assisted the department in maintaining and giving us good service in terms of purchasing software and hardware for us, *Thank you*.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

1 Introduction

The evolution of legacy Analogue studios to Firewire (IEEE1394 standard) based technology brought with it many complications with respect to audio and control data routing between communicating studio devices. In a simple legacy studio configuration, a large number of cables (of different types) are used to connect audio devices [*Figure 1.1*, Foss, 2005].

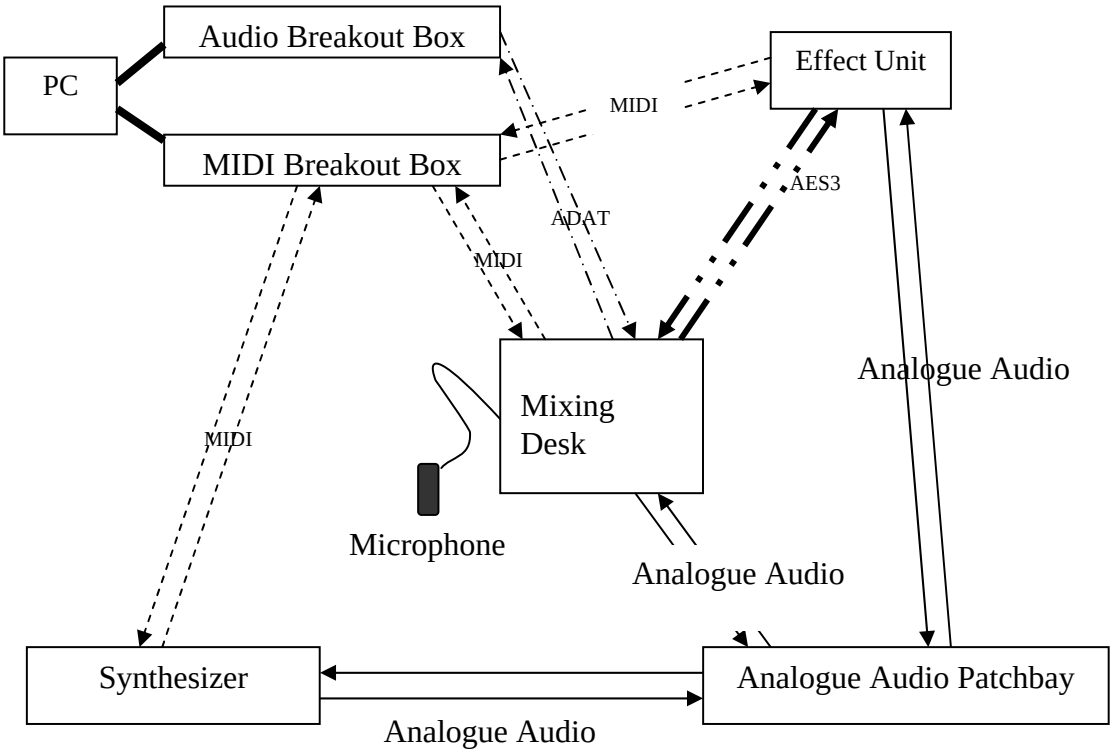

Figure 1.1: Simple Legacy Studio Configuration
[Foss, 2005]

Each cable type carries different types of audio data. *Figure 1.1* shows a simple analogue studio configuration with cables carrying:

- MIDI (Musical Instrument Digital Interface) data – For example, cables between the MIDI Breakout Box device and the Effect Unit device.
- Analogue audio data – For example, cables between the Mixing Desk device and the Analogue Audio Patchbay device.

- ADAT (Alesis Digital Audio Transmission) – For example, cables between the Mixing Desk device and the Audio Breakout Box device.
- AES3 (An audio Engineering Society Standard for the transmission of stereo digital audio, often termed AES/EBU) – For example, cables between the Mixing Desk device and the Effect Unit device.

As observed in *Figure 1.1*, many cables (of different types) are required to route audio, MIDI and control data between audio devices. As a result, adding one device requires an addition of more cables into the network, resulting in undesirable cable clutter. This makes connection management costly in terms of time and the money required for purchasing many cables of different types, and difficult. It is also labour intensive for sound engineers, since audio routing for these networks is performed by physically switching cables between devices. Moreover, additional costs are incurred through purchasing hardware Analogue Audio Patchbays, which are used to increase the flexibility of the network to allow fast and convenient access to audio signals at all strategic points in the signal paths [Sound On Sound Ltd, 1999]. Audio signals converge to a single, convenient location for effective routing using hardware Analogue Audio Patchbays.

In 1993, Yamaha Corporation initiated a project called mLAN in which Firewire was chosen as the networking standard for small and large professional studios to reduce cable clutter and provide a flexible peer-to-peer networking standard. mLAN stands for music Local Area Network. It is a Firewire-based protocol for high-speed transmission and control of multiple channels of audio and MIDI streams over a network [Yamaha Corporation, 2004c]. It involves intelligent connection management that allows the audio engineer to make connections and have full control over the entire music network without having to plug or unplug a single cable [Yamaha Corporation, 2004c]. *Figure 1.2* shows the studio devices displayed in *Figure 1.1*, this time they are daisy-chained using a single Firewire cable in an mLAN network. Daisy-chaining audio devices using a single Firewire cable significantly reduces cable clutter and the cost of installing and managing the studio since there is no need for purchasing many cables (of different type) and hardware Analogue Audio Patchbays.

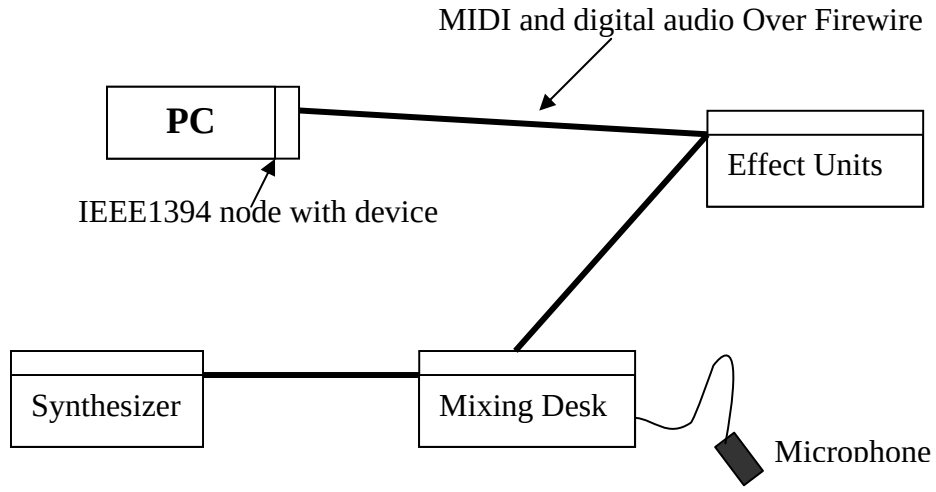

Figure 1.2: Simple Studio with Firewire Configuration

In the mLAN network configuration [*Figure 1.2*], the use of one Firewire connector for carrying data means that audio, MIDI and control data traverse the same cable using time-division multiplexing techniques with each signal sent occupying a dedicated channel. The number of audio channels supported by mLAN is influenced by the speed of the bus, which directly affects the bandwidth of the network, and how many channels the total network can support. mLAN can support thousands of MIDI cables and thus tens of thousands of MIDI channels. It is however, very common for manufacturers to make equipment with 8 MIDI ports supporting 128 MIDI channels per device [Yamaha Corporation, 2004c].

Audio routing in mLAN networks is done on a central network controlling application, referred to as a patchbay, running on a workstation that is connected to the mLAN network. Many audio solution companies have developed patchbays using third-generation languages for use with their proprietary hardware. The Audio Engineering Group (AEG) of the Rhodes University Computer Science Department is working in collaboration with Yamaha Japan, to improve the original mLAN architecture and develop a flexible mLAN Client/Server architecture that is discussed in *chapter 2*.

1.1 Project Motivation

Only the Windows Explorer - style patchbay (also known as NAS Explorer patchbay – *Figure 1.3*) has been developed for the client side of the mLAN Client/Server architecture described in *section 2.3*. The NAS Explorer patchbay uses two collapsible/expandable tree lists that display the source and destination devices on the network. It is easy to use even for first time users and allows the viewing of many devices on the network. However, the NAS Explorer patchbay cannot be used in all sound installation networks (Broadcast networks, Project studio networks and Hospitality/Convention Centre networks – these networks are discussed in detail in *section 3.1*). For instance, although the NAS Explorer patchbay allows for the display of many network devices at a time and is easy to use, it does not provide a graphical representation of the connection status for individual device plugs (whether a plug is connected or not connected). This feature is essential in complex sound installations such as Broadcast networks. Broadcast networks are complex, distributed in nature, and deal with many connections at a time. Due to this complexity, it is important for the sound engineer to readily view the connection status of plugs.

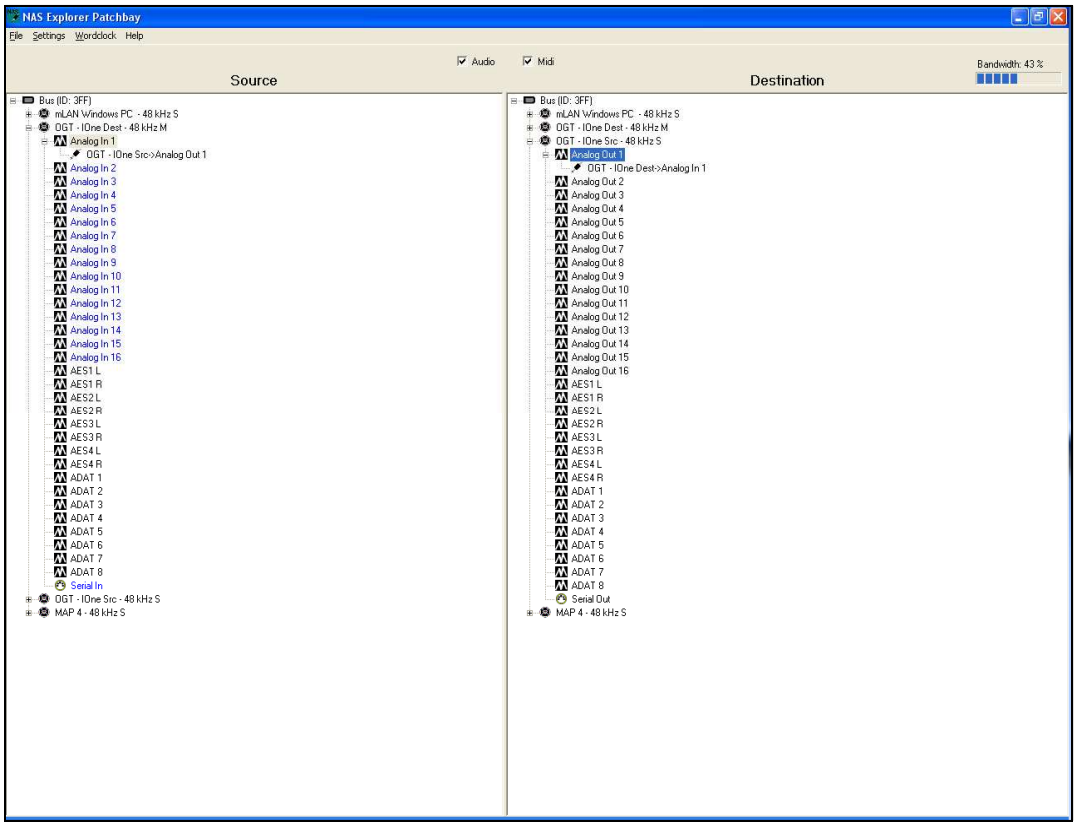

Figure 1.3: NAS Explorer Patchbay

Music producers and musicians in Project studio networks prefer a patchbay that displays, visually, devices on the network such as graphic-based patchbays instead of tree lists used for the NAS Explorer patchbay. As a result, the NAS Explorer patchbay is difficult to use in Project studio networks. Hospitality/Convention Centres networks are usually controlled by inexperienced personnel who do not need to see the whole list of plugs as exposed by the NAS Explorer patchbay [*Figure 1.3*], but only those (plugs) that have devices connected to them. Furthermore, the naming convention for devices and plugs used by the NAS Explorer patchbay is too complex for novice users in Hospitality/Convention Centres networks. The NAS Explorer patchbay makes use of names such as *OGT - I/One Dest* and *AES1L* for device nodes and plugs respectively. These do not make sense to Hospitality patchbay users; they prefer working with simple familiar names such as speaker, microphone and radio that make sense to them.

Table 1.1 shows examples of current non-mLAN patchbay types that are developed using third-generation languages such as C, C++ and C# for Broadcast, Project studio and Hospitality/Convention Centre networks. Non-mLAN patchbays do not utilise the mLAN Client/Server architecture described in *section 2.3*.

Table 1.1: Current non-mLAN Patchbays Examples

Patchbay Type	**Patchbay Examples**
Grid-Based Patchbays	• OTARI mLAN Control Software , the CobraNet™ Manager • Digigram's ESControl Management Software for EtherSound networks
List-Based Patchbays	• mLAN Patchbay • PathfinderPC Router Control Software
Graphic-Based Patchbays	• mLAN Version 2 Devices patchbay • Digigram's Hospitality Audio Manager

Chapter 3 gives a detailed description of these non-mLAN patchbays. They are categorised into three groups namely; grid-based patchbays, list based patchbays and graphic-based patchbays.

1.2 The Problem Statement

Due to the rapid increase in distributed audio/video Firewire-based network usage in the world at the time of this investigation, mLAN patchbays that control audio routing over these networks have also evolved in complexity to accommodate more functionality. Furthermore, the inclusion of Firewire as a networking standard has resulted in a steady increase in the use of Graphic User Interface (GUI) as more and more people move away from physically switching audio cables to perform audio routing to software based audio routing. More functionality needs to be exposed on the patchbay interface for the user. This functionality can be incorporated into patchbay applications using third-generation languages as explained in the preceding section but at a high cost in terms of the effort and time required to develop them. In addition to this, current software development techniques require that software be developed iteratively in cycles, alternating the development phases with testing. This means software development tools and networks used should support fast development of prototypes and parts of the system for testing as well as provide a way of quickly modifying these, once they are tested at each cycle level. This has led to a need among high-speed audio networking patchbay developers for a development tool and environment that will:

- Enable the development of connection management applications easily and within the shortest possible time.
- Provide strong graphic control component capabilities that will allow the developers to incorporate as much functionality as possible on the patchbay interface with least effort without compromising the quality of the software.
- Provide the reliability and flexibility required for developing applications controlling signal routing in networks carrying real-time data.
- Allow for the implementation of connection management applications that satisfy specific studio network requirements for different sound installation networks.

The main aim of this investigation was to evaluate the possibility of using a high-level graphic tool that supports scripting technology for developing mLAN connection management applications instead of the traditional third-generation languages. Five

possible scripting technologies that could be used include; the Microsoft Silverlight, the Sun Microsystems JavaFX, the Adobe Systems Adobe Flash Professional 8, the Adobe Systems Adobe Flex 2, and the Laszlo Systems OpenLaszlo [*Chapter 4*].

Adobe Flash Professional 8 using ActionScript 2.0 was chosen as the Integrated Development Environment (IDE) of choice for developing mLAN connection management applications for mLAN Client/Server networks. It was chosen for this investigation because of its support for Object Oriented Programming (OOP) concepts and its XML capabilities. In addition to this, in the beginning of this investigation, there was Adobe Flash expertise in the Rhodes Computer Science department and the software was already available locally. No costs were incurred to use Adobe Flash and the developer had access to experienced Adobe Flash users, which accelerated his learning of the Adobe Flash toolkit. To achieve this goal, three connection management applications were developed for controlling audio routing in three sound installation environments, namely Broadcast networks, Project studio networks and Hospitality/Convention Centre networks. Two usability techniques were then employed for evaluating the usability of the three applications to see if they fulfilled the requirements of each audio. The two usability techniques used include a heuristic evaluation that was done to find usability problems and missing functionality on the patchbays that were fixed before the applications were sent to potential users for a further usability testing based on a usability questionnaire.

The secondary aim of this investigation was to determine, using specific sound installation requirements, which patchbay design (grid-based, list-based and graphic-based) would best suit the three sound installation networks. Two prototypes, a computer prototype and a paper prototype, were designed and tested by real users to gather requirements, help decide on the best design and layout for the Project studio and the Hospitality/Convention Centre patchbays respectively. Since the nature of Broadcast networks are well defined and well known within the audio industry, the best design and layout for the Broadcast patchbay was chosen from analysing current Broadcast network patchbays [*Chapter 3*].

1.3 Chapter Summaries

Chapter 2 describes the current state of the mLAN project. It discusses the main mLAN components, namely the Enabler/Transporter architecture, the mLAN Client/Server configuration, and the XML communication protocol that was used for communication between the mLAN patchbays and the server (known as the mLAN Connection Management (mCMS) server).

Chapter 3 describes current non-mLAN Client/Server and mLAN Client/Server connection management applications. This chapter identifies and describes the three basic sound installation network categories that include:

- Broadcast networks.
- Project Studio networks.
- Hospitality/Convention Centre networks.

Chapter 4 discusses five alternative development environments for developing mLAN client patchbays:

- The Microsoft Silverlight.
- The Sun Microsystems JavaFX.
- The Adobe Systems Adobe Flash Professional 8.
- The Adobe Systems Adobe Flex 2.
- The Laszlo Systems OpenLaszlo.

The Adobe Systems Adobe Flash Professional 8 was chosen for developing mLAN Client/Server connection management applications. A detailed description of the Adobe Flash Professional 8 programming interfaces is also given.

Chapters 5, *6* and *7* describe the development process of a grid-based patchbay for Broadcast networks, as well as two graphic-based patchbays for Project studio and Hospitality/Convention Centre networks, respectively. Use case, Sequence diagrams, and Object models are used to describe the functionality incorporated into each

patchbay discussed. Each application was evaluated and tested using a heuristic evaluation checklist and a usability test questionnaire.

Chapter 8 describes Adobe Flash Professional 8 and ActionScript 2.0 capabilities that aided the development of the three mLAN Client/Server patchbays described in *chapters 5*, *6* and *7*.

Chapter 9 gives the conclusion to this research, and briefly discusses the benefits of using Adobe Flash and ActionScript 2.0 for developing mLAN Client/Server patchbays.

CHAPTER 2

2 The Current Status of mLAN and the mLAN Client/Server Architecture

A number of audio networking technologies have been developed and deployed by various audio solution providers across the audio industry, to deal with end-to-end connection management in various sound installations. These (audio networking technologies) are a combination of hardware and software protocols that are designed to control audio and MIDI data routing over various mediums such as Firewire and Ethernet. True end-to-end connection management provides the capabilities of routing audio and MIDI data from a hard-end plug of a device or a data-bus line implemented within a device, onto an audio network, and vice-versa [Okai-Tettey, 2005]. This is done using a user-level application that exposes virtual plugs of devices on the network. The sound engineer performs audio routing tasks on this user-level application. Examples of current audio networking solutions include:

- The Yamaha Corporation's mLAN Digital Network Interface Technology [Yamaha Corporation, 2004c].
- The Aviom's A-NetTM Pro64 Technology [Aviom Inc, 2007].
- The Axia Audio's Livewire Technology [Axia Audio/TLS Corporation, 2005].
- The Cirrus Logic's CobraNetTM technology [Cirrus Logic, 2007].
- The Digigram's EtherSound technology [Digigram, 2007].
- The Intelligent Media's SmartBuss Technology [IMT Inc, 2005].

The Yamaha Corporation's mLAN Digital Network Interface Technology was chosen for this investigation as its extensive development was done by the Rhodes Audio Engineering Group in collaboration with Yamaha Japan. This provided the researcher with easy and cheap access to mLAN resources such as the mLAN source code and documentation. Furthermore, the researcher had the opportunity of interacting and learning directly from experienced mLAN developers at no cost. mLAN uses its architecture (the Enabler/Transporter architecture – *section 2.2*) to support true end-

to-end connection management with reasonable Quality of Service (QoS) and it implements mechanisms that control bandwidth allocation within the network, therefore optimising its utilisation and its distribution of audio to network devices. This chapter defines mLAN and describes the mLAN Enabler/Transporter architecture and its components that enable successful connection management in mLAN networks.

2.1 The current mLAN Architecture

According to Fujimori and Foss (2003), mLAN can be described as a networking technology that allows the transport of audio and music control data between audio devices[1]. mLAN uses Firewire, also known as IEEE 1394, as its base networking technology. Firewire is a high-speed serial-bus standard that offers enhanced connectivity and data transfer for video, audio and storage peripheral applications through a universal input/output (I/O) interface [Anderson, 1999]. It was chosen as the mLAN base networking technology for the following reasons [Anderson, 1999, Foss, 2005]:

- It has low latency transmission of audio (i.e. does not have long delays in audio transmission) and determinism (i.e. provides guaranteed transmission within a particular time frame) required by a network carrying real-time data.
- It has power-carrying capabilities that enable the connection of devices with no power supply.
- It supports hot-plugging and plug-and-play performance capabilities, which eliminate the need for a host workstation when attaching or detaching devices. This is because when attaching/detaching devices to/from the network, an automatic bus reset forces network re-enumeration, a process which automatically detects new devices without the need for a controlling workstation.
- It provides peer-to-peer data transfer capabilities. This eliminates the need for routing audio and MIDI data to a controlling workstation in order for one device to communicate with another. This reduces processing overhead and improves communication speeds between devices.

[1] mLAN network devices are also referred to as nodes , IEEE 1394 nodes or units.

Figure 2.1 displays the current mLAN version 2 architecture that was utilised for this investigation. The diagram shows two mLAN compatible devices, the *Synthesizer* and the *Mixer*, connected to a controlling host *Workstation*. On the *Workstation* is a module known as the Enabler that enables connections between mLAN plugs to be made [Foss, 2005]. The Enabler comprises a Hardware Abstraction Layer, an A/M (Audio and Music data) Manager layer, and an mLAN Plug Abstraction Layer. Each mLAN device incorporates a Transporter, which is responsible for the transport of audio and music data in a manner that is compliant with the Audio and Music data transmission protocol [Fujimori et al. 2003]. The Transporter comprises a Node Controller and the firmware that facilitates audio and music data encapsulation and transmission between communicating mLAN devices.

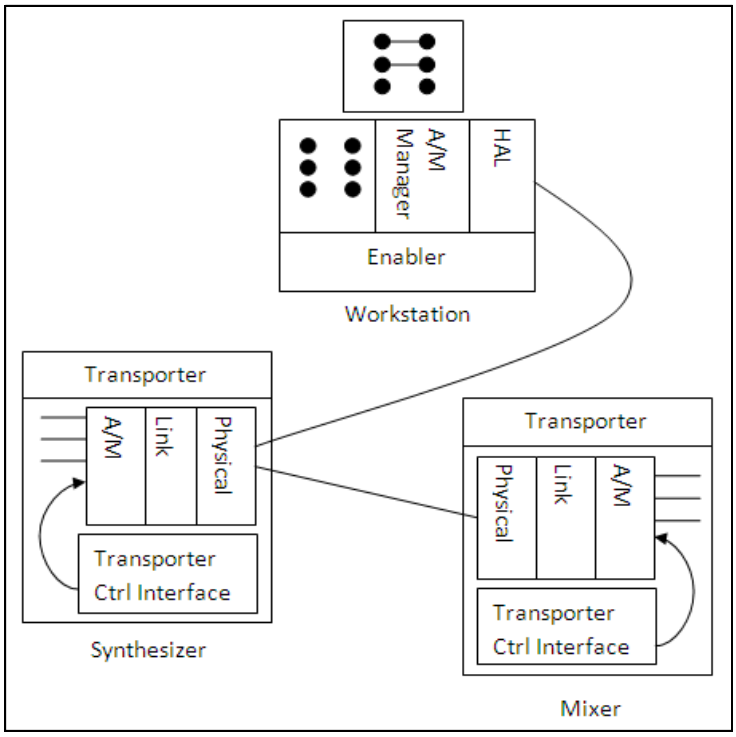

Figure 2.1: mLAN Version 2 Architecture [Fujimori et al. 2003]

2.1.1 The Enabler

The Enabler is responsible for setting audio and music data parameters for transmission and reception of audio and music data sequences for all Transporters under its control [Foss, 2005]. Many Enabler modules may exist within one mLAN network but each Transporter is controlled by only one Enabler at a time [Fujimori et al. 2003].

2.1.1.1 mLAN Plug Abstraction Layer

The top most layer of the Enabler module is the mLAN Plug Abstraction Layer, which is responsible for implementing input and output mLAN plug abstractions for all possible "hard" end points of all Transporters under the Enabler's control [Fujimori et al. 2003]. These mLAN plug abstractions can be viewed as the terminators of the audio and music data sequences that are transmitted and received by the Transporters. This layer also provides an API (Application Programming Interface) that can be used by connection management applications (patchbays) for performing tasks such as making and breaking audio connections and word clock synchronisation [Foss, 2005].

2.1.1.2 A/M Manager Layer

Below the mLAN Plug Abstraction Layer is the A/M Manager layer, which is responsible for reading audio and music data transmission and reception parameters from the associated Transporter, and for updating these parameters in response to requests from the mLAN Plug Abstraction layer [Fujimori et al. 2003]. *Figure 2.2* shows the Enabler's layers and interfaces. It can be seen from the diagram that each of the Transporters under the control of an Enabler has a Transporter object that keeps its state information and handles requests from both the Plug Abstraction Layer and the actual Transporter [Fujimori et al. 2003]. The A/M Manager layer also provides an API that is used by the Plug Abstraction Layer to perform its tasks.

2.1.1.3 Hardware Abstraction Layer (HAL)

The bottom layer of the Enabler is the Hardware Abstraction Layer (HAL) whose sole purpose is to hide away the hardware implementations of the Transporters. This layer and the Transporter Control Interface allow non-mLAN chip manufacturers to acquire mLAN compliance [Fujimori et al. 2003, *Figure 2.2*]. Each chip manufacturer can

provide a plug-in that bridges between the A/M Manager and their vendor specific Transporter Control Interface [*Figure 2.2*].

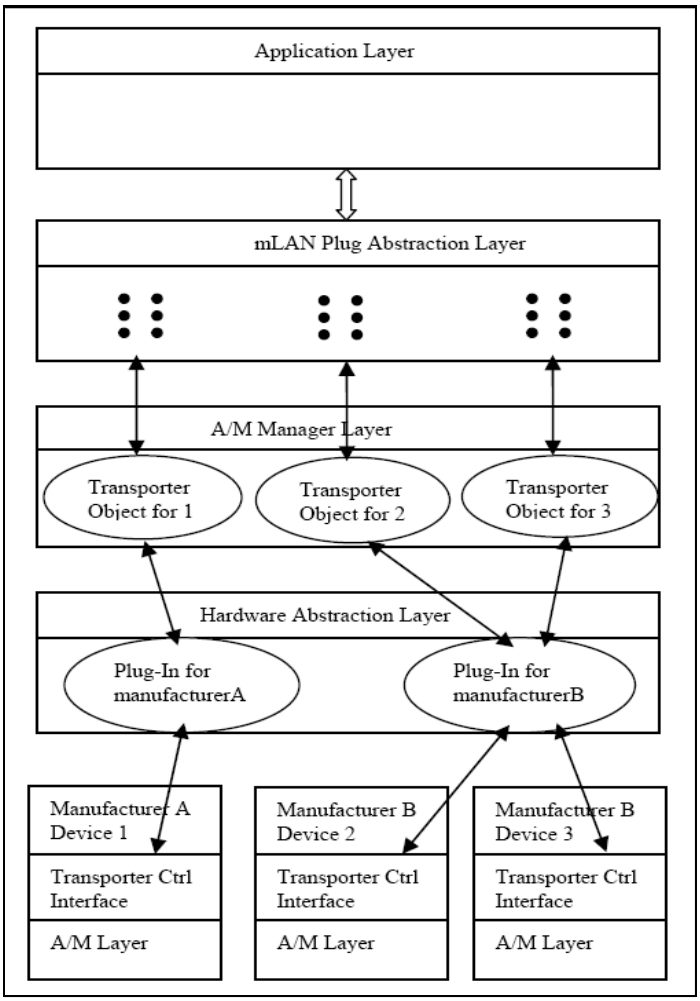

Figure 2.2:The Enabler's Layers and Interfaces [Fujimori et al. 2003]

Figure 2.2 shows three Transporters; one of which contains a Node controller from Manufacturer A, and two of which contain Node Controllers from manufacturer B. Both Manufacturer A and B have created workstation plug-ins and embedded Transporter Control Interface software. In addition to this, Manufacturer A and B will have created their own proprietary messaging protocols between their respective plug-ins and Transporter Control Interfaces. The A/M Manager shown in *Figure 2.2* has instantiated three Transporter objects that provide for A/M related control, and access to their respective hardware Transporters. The plug-ins are responsible for fulfilling these control and access requests. Via interaction with these Transporter objects, the

mLAN Plug Abstraction layer can build mLAN plug objects, which in turn are accessible to connection management applications [Fujimori et al. 2003].

2.1.2 The Transporter

The Transporter is composed of hardware chips and firmware that allow mLAN devices to transport isochronous streams within the mLAN network as well as detect and interpret instructions passed by the controlling Enabler via asynchronous transactions [Foss, 2005]. These Transporter hardware chips and firmware collectively form a Node Controller. All devices participating in mLAN network transactions host a Node Controller. Hardware chips that have been implemented for the mLAN project include the PH1 chip, the NC1 chip, and the PH2 chip [Fujimori et al. 2003, Foss, 2005].

2.1.2.1 mLAN Sequence Encapsulation and Extraction Hardware

The components of an mLAN connection management system are designed to control the encapsulation of sequences into streams on the transmitting device side and their subsequent extraction on the receiver's side [Fujimori, et al, 2003]. Data structures, formats and hardware have been developed for this encapsulation and extraction of sequences in mLAN networks. *Figure 2.3* displays two Node Controllers for a *Synthesiser* (the transmitting device shown in *Figure 2.1*) and for a *Mixer* (the receiving device shown in *Figure 2.1*).

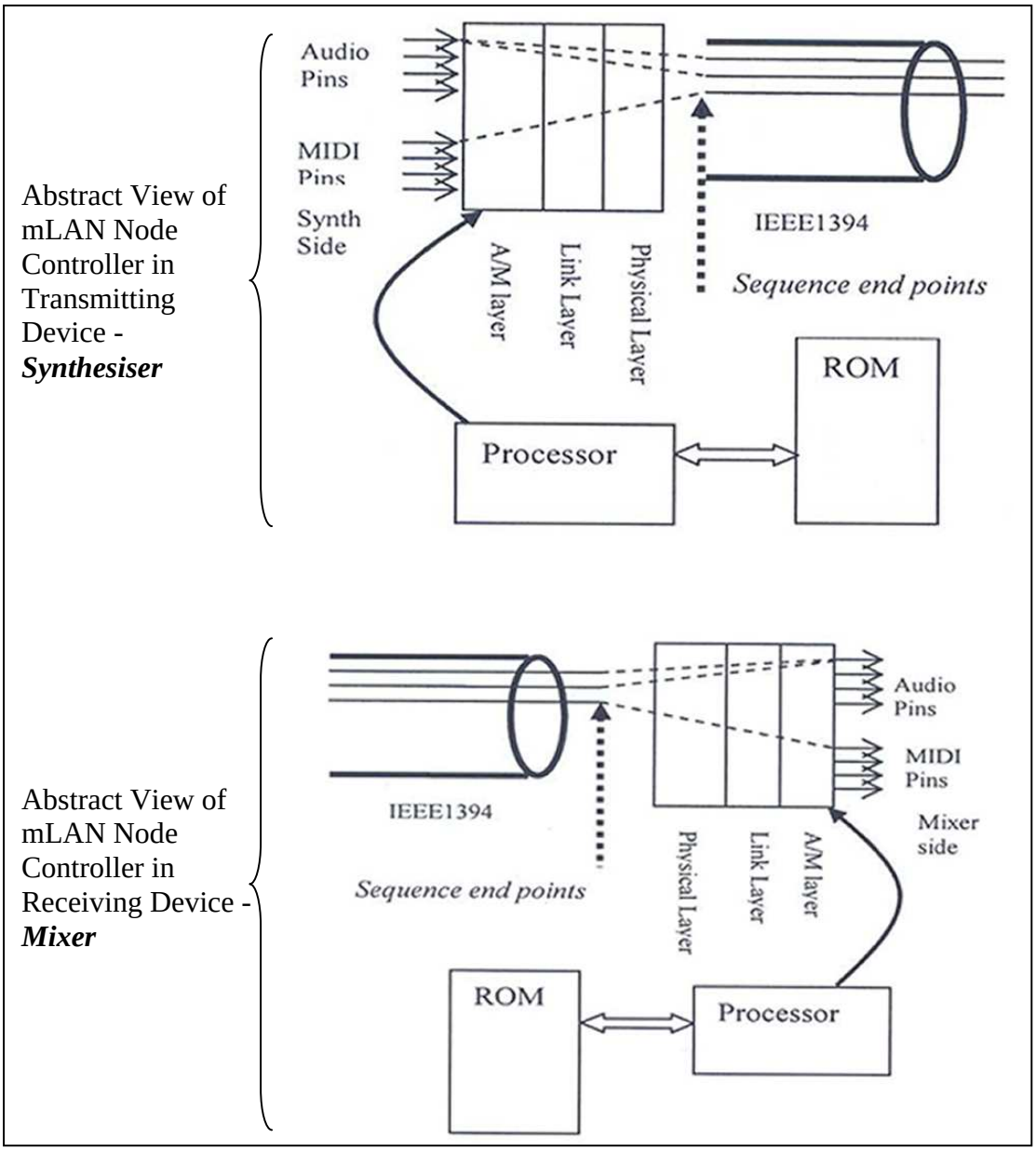

Figure 2.3: mLAN Node Controllers in a Transmitting Device (Synthesiser) and a Receiving Device (Mixer) [Fujimori et al. 2003]

Each Node Controller comprises the following components [Foss, 2005]:

- An mLAN 2.0 Configuration ROM (Read Only Memory) – It keeps and provides information for the device enumeration process. Information provided by the ROM is utilised for:

    a) Identifying software drivers for network devices.
    b) Identifying diagnostic software for network devices.
    c) Specifying capabilities for network devices.

d) Optionally specifying module, node and unit characteristics for network devices.

- The A/M Manager Layer – This layer is responsible for reading audio and music data transmission and reception parameters from the associated Transporters, and for updating these parameters in response to requests from the Enabler.

- The Link Layer – This layer provides the interface between the transaction layer (A/M Manager Layer) and the physical layer during asynchronous transactions.

- The Physical Protocol Layer – This layer is the bottom layer of the Transporter and provides the interface to the IEEE 1394 bus. It is this layer that is responsible for detecting packet bits from the bus and transmitting them packet bits to the network. This layer is implemented in hardware.

- The Processor – It controls the functioning of the A/M Manager Layer by translating to it the asynchronous commands it receives from the controlling Enabler.

2.1.2.2 mLAN Audio and Music data Transmission

Audio and control data in mLAN networks can be transmitted either isochronously or asynchronously.

a) Isochronous Transactions

Isochronous streams transmitted and received by mLAN devices constitute isochronous packets, which form sequences that transport successive samples of an audio channel, or successive MIDI commands of a MIDI command stream [Fujimori et al. 2003, *Figure 2.4*].

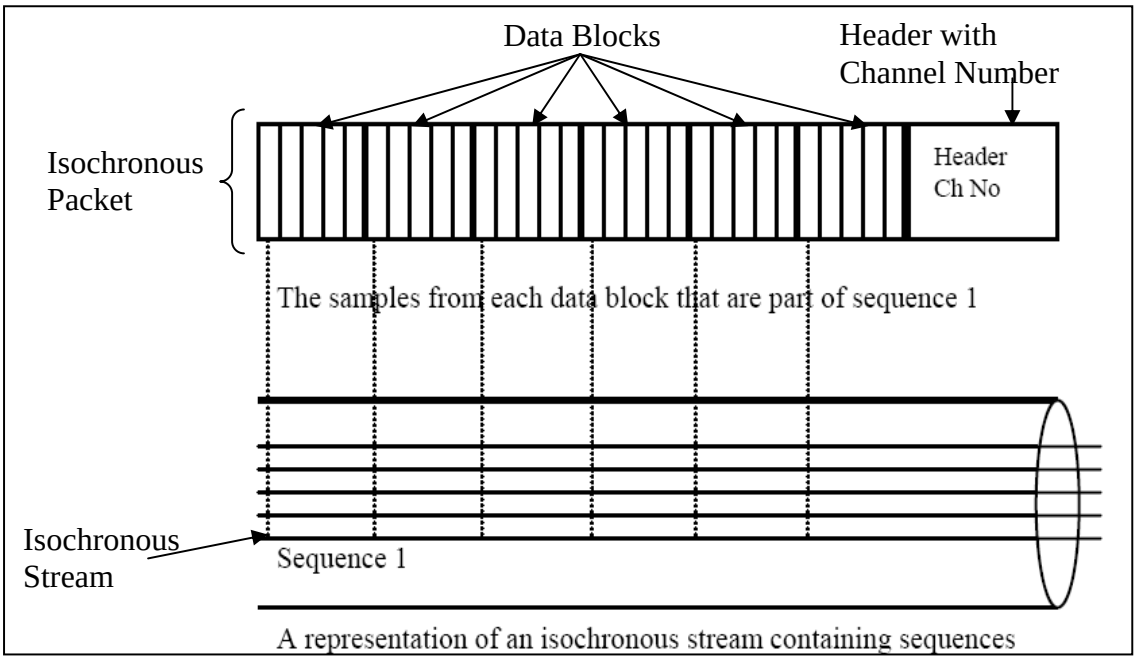

Figure 2.4: An Isochronous Stream with Sequences
[Fujimori et al. 2003]

Figure 2.4 displays an *isochronous stream* with isochronous packets. Each packet contains a number of *data blocks*. A *data block* comprises audio samples and MIDI data from different sequences that occur at the same point in time. The sampling rate of the transmitting device determines the number of *data blocks* in an isochronous packet. The isochronous packet [*Figure 2.4*] consists of six *data blocks* and a *header* that contains a *channel number*. All packets with the same *channel number* form an *isochronous stream*. Each *data block* contains 32-bit quadlets that carry audio and MIDI data. The quadlets are ordered within the *data blocks*, and those (quadlets) that occur at a particular *data block* position are referred to collectively as a sequence [Fujimori et al. 2003]. *Figure 2.5* displays the packet format of an *isochronous packet* [Foss, 2005].

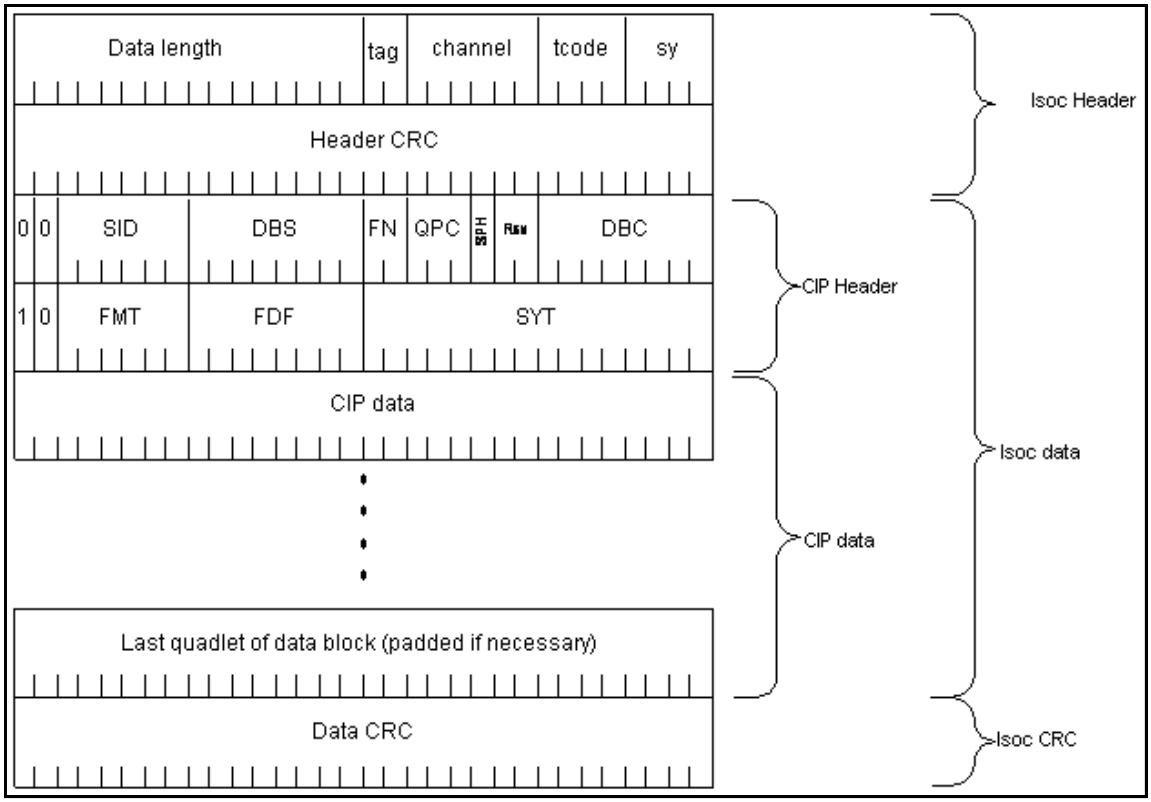

Figure 2.5: The Isochronous Packet Format
[Foss, 2005]

This section describes only a few *isochronous packet* fields shown in *Figure 2.5* that are relevant to this investigation [Foss, 2005]:

- The *data length* field - Holds a value that specifies the number of bytes of data carried in the packet.
- The *channel* field - Holds the isochronous *channel number* assigned to the packet and can contain any number from 0 to 63, which is the number of nodes allowed on a single bus in mLAN networks. To achieve packet transmission between two communicating devices, both devices should transmit and receive packets at the same *channel number*.
- The *SID* (Source node ID) field - Holds the source device identifier, which is a 6-bit physical ID of the node transmitting the packet.
- The *DBS* (Data Block Size) field - Holds a number that indicates the data block size.

- The *DBC* (Data Block Count) field - Holds a sequence counter and continuity checker that is used by receivers for audio synchronization and to detect any lost data blocks.
- The *SYT* (Time Stamp) field - Indicates the time that a particular data block within the packet should be presented at the receiver. This is also used for audio synchronization.

In isochronous transactions, isochronous packets are transmitted every 125 microseconds which is equivalent to 8000 packets per second.

b) Asynchronous Transactions

Asynchronous transactions enable an IEEE 1394 node to write/read commands to/from another node's addressed memory location. According to Foss (2005), an asynchronous transaction is initialised by a requester node and received by a responder node. Each transaction consists of two subsections that include [Foss, 2005]:

- A request subaction - Transfers the address, command and data (for writes) from the requester to the responder.
- A response subaction – Returns completion status (writes) back to the requester node, or returns data during read transactions.

Asynchronous transactions are performed using asynchronous packets.

2.1.3 mLAN Device Synchronisation Mechanism

In order for two communicating devices to successfully transmit and receive audio data from each other, they need to transmit and receive data at the same sample rate to avoid duplication of audio samples and buffer overflows, which introduces undesired audio breaks that affect the quality of the transmitted audio [Foss, 2005]. Current supported sample rates are 32 kHz, 44.1 kHz, 48 kHz, 88.2 kHz, 96 kHz, 176.4 kHz and 192 kHz. mLAN utilises the device synchronisation mechanism defined in the IEC 61883-6 specification [IEC, 2005], which depends on two key points:

- A cycle master/cycle slave relationship.
- The inclusion of the Common Isochronous Packet (CIP) header below the IEEE 1394 isochronous header in each isochronous packet transmitted.

Each mLAN node implements a Cycle_Time register, which is incremented via a 24.576 MHz clock. The Cycle_Time registers of all nodes are synchronized by *cycle start* packets sent by the cycle master node every 125 microseconds. These *cycle start* packets contain the Cycle_Time value of the Cycle_Time register of the cycle master device. All isochronous packets transmitted are time stamped with a 25-bit sample rate of the transmitting device's Cycle_Time register value and a delaying offset added to it. This value is stored in the SYT field of the CIP header [*section 2.1.2.2*]. At the receiving device, the SYT field time stamp value is extracted and used to re-create the sampling rate of transmitted audio samples [*Figure 2.6*].

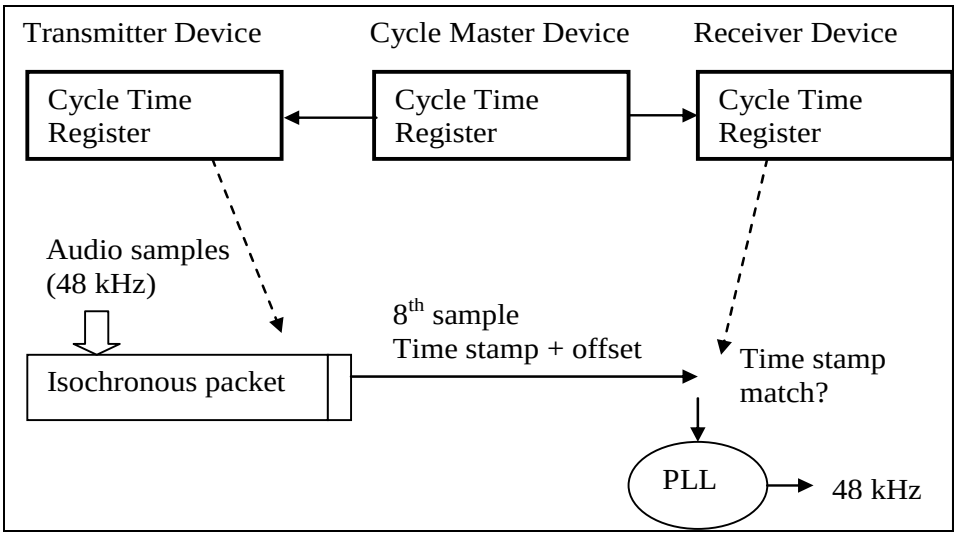

Figure 2.6: Sample Clock Synchronisation
[Foss, 2005]

In *Figure 2.6,* a transmitter device is generating audio samples at 48 kHz, and formatting them into *data blocks* within isochronous packets. At each 8th sample rate clock tick, the Cycle_Time value of the transmitter device is sampled and an offset added before its transmission within the next packet in the SYT field of the CIP packet header. The value in the SYT field is intended to be the presentation time of a *data block* (event) received by the receiver. The offset is known as the TRASNFER_DELAY, and takes care of the delay incurred in the transfer of an

isochronous packet from its transmitter to its receiver, and allows packets travelling along different paths to be presented at the same time. It also takes into account bus resets that may occur during transmission. At the receiver's side, the time stamp is read. If this value is the same as the receiver's Cycle_Time register value, it indicates a match to the Phase Locked Loop (PLL) [*Figure 2.6*]. A CIP packet may contain multiple events and only one time stamp. In order to associate the SYT time stamp with a particular event, a unit known as the SYT_INTERVAL is used that varies according to the sampling rate [*Table 2.1*]. The SYT_INTERVAL is defined as the number of events between two successive SYT time stamps.

Table 2.2: SFC (Nominal Sampling Frequency Code) Definition

SFC (Nominal Sampling Frequency Code) definition		
Value (decimal)	Sample transmission frequency	SYT_INTERVAL
0	32 kHz	8
1	44.1 kHz	8
2	48 kHz or not indicated	8
3	88.2 kHz	16
4	96 kHz or not indicated	16
5	176.4 kHz	32
6	192 kHz or not indicated	32
7	Reserved	

The transmitter prepares the time stamp for the event using the following condition:

$$\mathrm{mod}\ (DBC, SYT\_INTVAL) = 0$$

where *DBC* is the Data Block Count and *SYT _ INTVAL* is the number of events between two successive SYT time stamps [*Table 2.1*].

At the receiver's side, the index within a packet of the event to which the SYT time stamp applies can be calculated from:

$$\text{index} = \mathrm{mod}((\text{SYT\_INTVAL} - \mathrm{mod}(\text{DBC}, \text{SYT\_INTVAL})), \text{SYT\_INTVAL})$$

where DBC is the Data Block Count and SYT _ INTVAL is the number of events between two successive SYT time stamps [*Table 2.1*].

The receiver is responsible for estimating the timing of events between valid time stamps. Using this formula, the receiver determines the time stamp of a received packet as well as its *data block*. If the time stamp does not match its register value, it will not process this *data block*, and following *data blocks* until the cycle time is matched [Foss, 2005]. Using these mechanisms two mLAN devices can be synchronised to transmit and receive audio and MIDI data from each other at the same sample rate.

2.2 mLAN Client/Server Configuration

Figure 2.7 displays the mLAN Client/Server configuration that is used by the mLAN networks and that was utilised by connection management applications developed in this investigation.

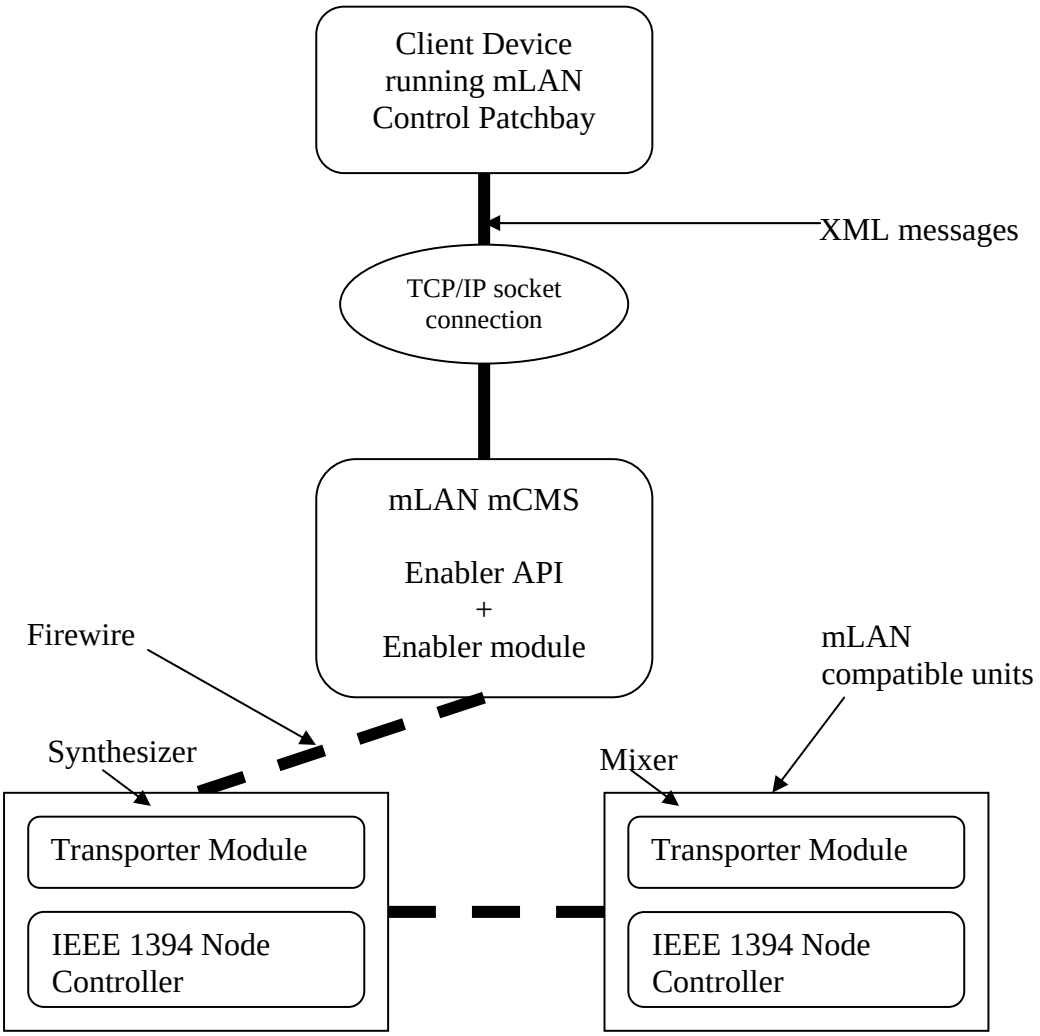

Figure 2.7: mLAN Client/Server Configuration
[Fujimori et al. 2003]

The mLAN Client/Server configuration [*Figure 2.7*] shows two mLAN compatible devices (the *Synthesizer* and the *Mixer*) connected to the mLAN Connection Management Server (mCMS) workstation using a Firewire cable. The mCMS workstation integrates the Enabler module and the Enabler API. Each device on the mLAN network hosts a vendor specific Transporter module [*section 2.1*] and an IEEE 1394 Node Controller. It is a requirement that all devices on the mLAN network incorporate an IEEE 1394 Node Controller, which enables mLAN devices to communicate with each other and the mCMS workstation over Firewire.

A communication protocol has been defined by Klinkradt (2004) that utilises Extensible Markup Language (XML) messages. This communication protocol enables mLAN client applications to communicate with the mCMS server through a TCP/IP socket [*Figure 2.8*]. The current socket number is 52941. Any communication medium can be used between the client and the server as long as they are able to communicate over the TCP/IP socket.

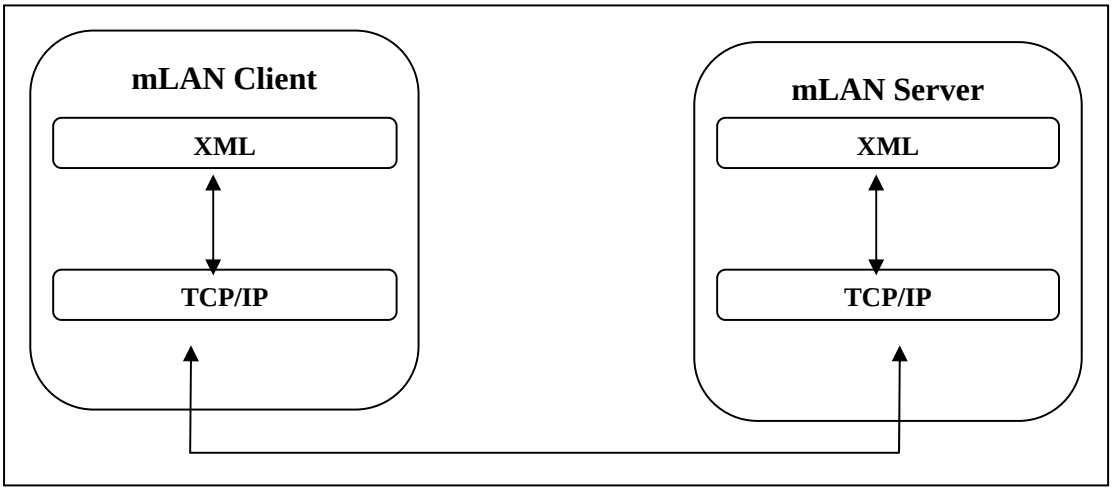

Figure 2.8: Client Server Communication Model
[Klinkradt, 2004]

The Client/Server approach allows for the decoupling connection management between the controlling workstation that runs the client applications and the mCMS server. In addition to this, communication using generic XML messages allows for multiple applications, developed in any language to connect and communicate with the mCMS server. The network can use any medium (physical Firewire cables, wireless or Ethernet technologies) as long as the communicating devices can pass

messages via TCP/IP [Fujimori et al. 2003]. As a result, client applications can be deployed for a wide range of devices that include Laptops and standalone workstations. When a user performs a connection management request (e.g. making a connection or disconnection between two plugs) on the connection management application, a request-specific XML document is created by the application and sent to the mCMS server. *Listing 2.1* shows a typical connection request XML document that requests a connection between an output plug with *sourcePlugID* "1" (on a device with device *GUID* "0013f00400011" and of type "audio") to an input plug with *destinationPlugID* "33" (on a device with device *GUID* "0013f00400000014" and of type "audio"). When the request is fulfilled, audio is routed from the output plug to the newly connected input plugs.

```
<?xml version="1.0" encoding="utf-16" ?>
- <mLANServerCommand version="1.0">
  - <object name="patch">
    - <method name="connect">
        <parameter name="sourceGUID" value="0013f00400400011" />
        <parameter name="sourcePlugType" value="audio" />
        <parameter name="sourcePlugID" value="1" />
        <parameter name="destinationGUID" value="0013f00400000014" />
        <parameter name="destinationPlugType" value="audio" />
        <parameter name="destinationPlugID" value="33" />
      </method>
    </object>
</mLANServerCommand>
```

Listing 2.1: XML "connection" Request Document

On the server side, the Request Server module processes the information in the XML document and forwards the request to the Devices Interface module. The Devices Interface uses the Enabler API to access the Firewire bus and implement the request in collaboration with the Transporter modules on each communicating device [*Figure 2.9*].

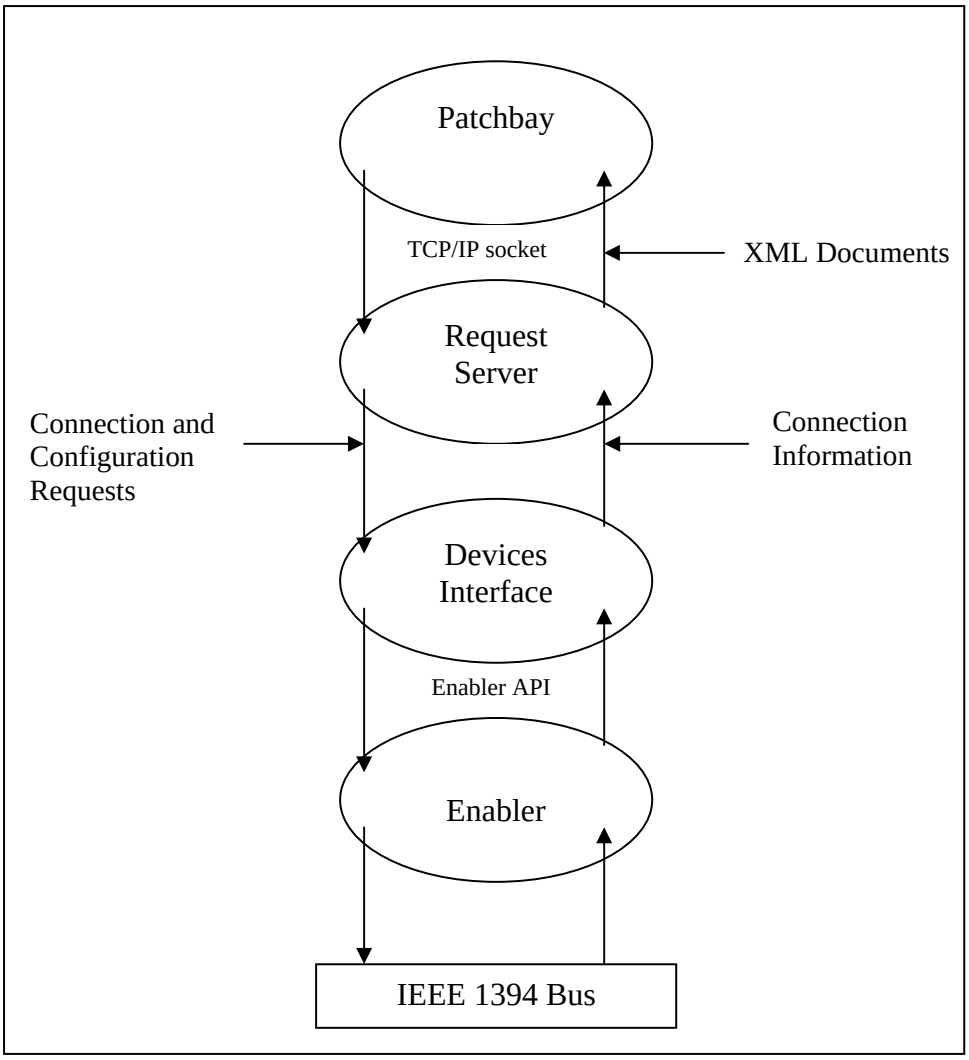

Figure 2.9: mLAN Client/Server Communication Interfaces

mCMS server responses are sent by the server to the connection management application using XML documents. *Listing 2.2* shows a section of an XML "configuration" document that is sent to the mLAN client application at start-up, and each time the sound engineer requests an application update from the mCMS server.

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
- <mLANClientCommand>
  - <object name="patch">
    - <method name="refresh">
      - <parameter name="configuration" value="Tue Feb 27 11:10:43 2007">
        - <mLANConfiguration>
          - <IEEE1394Network>
            - <IEEE1394Bus bandwidthAvailable="2756" busName="3FF">
              + <IEEE1394Device GUID="0013f00400400011" firmware="DICE II OGT 0.1" model="DICE II Evaluation Board" nickname="IOne Connects-left"
                  nicknameIsWriteable="yes" numPossibleDeviceConnections="4" vendor="WaveFront">
              - <IEEE1394Device GUID="0013f00400000014" firmware="X1 OGT 070222" model="Firewire Digital Audio Snake" nickname="IOne Connects-right"
                  nicknameIsWriteable="yes" numPossibleDeviceConnections="3" vendor="I/One Connects">
                - <mLANDevice>
                  - <mLANDevicePlugs>
                      <plug direction="out" id="66" isDangling="no" nameIsWriteable="no" plugName="Analog In 1" plugType="audio" />
                      <plug direction="in" id="115" isDangling="no" nameIsWriteable="no" plugName="MIDI Out" plugType="midi" />
                    </mLANDevicePlugs>
                  - <mLANDevicePlugLayouts currentPlugLayoutID="1" numPlugLayouts="3">
                      <plugLayout id="0" nameIsWriteable="no" plugLayoutName="tx1: 8 ADAT 8 AES. tx2: 16 Anal" />
                      <plugLayout id="1" nameIsWriteable="no" plugLayoutName="tx1: 8 AES 8 Analog" />
                      <plugLayout id="2" nameIsWriteable="no" plugLayoutName="tx1: 8 Analog" />
                    </mLANDevicePlugLayouts>
                  - <mLANDeviceSyncSources numSyncSources="4">
                      <syncSource currentSampleRate="0000bb80" id="0" nameIsWriteable="no"
                        supportedSampleRates="00007d00|0000ac44|0000bb80|00015888|00017700|0002b110|0002ee00|" syncMode="1"
                        syncSourceName="SYT" />
                      <syncSource currentSampleRate="0000bb80" id="2" nameIsWriteable="no"
                        supportedSampleRates="00007d00|0000ac44|0000bb80|00015888|00017700|0002b110|0002ee00|" syncMode="3"
                        syncSourceName="AES RX" />
                    </mLANDeviceSyncSources>
                  - <mLANDeviceWordClockOutputs numWordClockOutputs="1">
                      <wordClockOutput currentSyncSourceID="0" id="0" masterGUID="0013f00400400011" masterWordClockOutputID="0" />
                    </mLANDeviceWordClockOutputs>
                  </mLANDevice>
                </IEEE1394Device>
              </IEEE1394Bus>
            </IEEE1394Network>
          - <Connections>
              <Patch destEndPointLocator="NODE_GUID='0013f00400400011',MLAN_PLUG_ID='Audio In 1'"
                srcEndPointLocator="NODE_GUID='0013f00400000014',MLAN_PLUG_ID='Audio Out 2'" />
            </Connections>
            <session end="2007-02-27 10:56:54" sessionID="0" sessionName="dummy" start="2007-02-27 10:56:54" />
          </mLANConfiguration>
        </parameter>
      </method>
    </object>
```

Listing 2.2: XML "cconfiguration" Document

The XML "configuration" document elements correspond to the actual physical components that constitute the mLAN network, and include:

- The *IEEE1394Network* element.
- The *IEEE1394Bus* element.
- The *IEEE1394Device* element.
    a) The *AudioPlug* element.
    b) The *mLANDevicePlugLayouts* element.
    c) The *mLANDeviceSyncSources* element.
    d) The *mLANDeviceWordclockOutputs* element.
- The *connections* element.
- The s*ession* element.

The following sub-sections describe these elements in detail.

2.2.1 The *IEEE1394Network* element

The *IEEE1394Network* element is the top-level element that contains zero or more *IEEE1394Bus* elements, one *connections* element and one *session* element.

2.2.2 The *IEEE1394Bus* element

The *IEEE1394Bus* element contains zero or more *IEEE1394Device* element(s). The number of bridges on the mLAN network determines the number of IEEE1394 buses that the network has. *Listing 2.*2 shows an IEEE1394 bus named "3FF", which represents the local IEEE1394 bus connected to the host workstation. The bandwidth attribute value "2756" is the amount of unused bandwidth on the mLAN network.

2.2.3 The *IEEE1394Device* element

The *IEEE1394Device* element constitutes of four sub-elements, zero or more *AudioPlug* element(s), one *mLANDevicePlugLayouts* element, one *mLANDeviceSyncSources* element, and one *mLANDeviceWordclockOutputs* element. It maps directly to a particular physical mLAN device on the mLAN network. Globally Unique Identifier (GUID) numbers are used to uniquely identify each mLAN device on the network. *Listing 2.*2 displays two *IEEE1394Device* elements, which represent two devices with GUIDs "0013f00400400011" and "0013f00400000014". Other attributes of the *IEEE1394Device* element(s) include:

- Firmware – The firmware of the device.
- Model – The model number of the device.
- Nickname – The nickname of the device.
- NicknameIsWriteable – The variable which specifies whether a device nickname can be changed ("yes") or not ("no").
- NumPossibleDeviceConnections – The number of possible connections for the device.
- Vendor – The vendor of the device.

2.2.3.1 The *AudioPlug* element

Each *IEEE1394Device* element contains one *mLANDevice* element, which in turn contains zero or more *AudioPlug* elements. The *mLANDevice* element indicates that the current device is an mLAN device. There are two types of *AudioPlug* elements; the *AudioInPlug* element and the *AudioOutPlug* element. The *direction* attribute identifies the type of *AudioOutPlug* element. The *AudioInPlug* element represents an input audio plug, and the *AudioOutPlug* element represents an output audio plug. Other *AudioPlug* element attributes include:

- Id – This attribute is the identification number of the plug.
- IsDangling – This attribute states whether the plug has a dangling connection or not.
- NameIsWriteable – This attribute specifies whether the name of the plug can be changed.
- PlugName – This attribute specifies the name of the plug.
- PlugType – This attribute specifies the type of the plug (Audio or MIDI).

2.2.3.2 The *mLANDevicePlugLayouts* element

Each *IEEE1394Device* element contains one *mLANDevicePlugLayouts* element, which contains device Plug Layouts that are supported by the device.

2.2.3.3 The *mLANDeviceSyncSources* element

Each *IEEE1394Device* element contains one *mLANDeviceSyncSources* element, which contains synchronisation sources that are supported by the device.

2.2.3.4 The *mLANDeviceWordclockOutputs* element

Each *IEEE1394Device* element contains one *mLANDeviceWordclockOutputs* element, which contains the currently set word clock for the device.

2.2.4 The *connections* element

The *connections* element constitutes of one or more *patch* element(s), which represent individual connections that exist between mLAN devices on the mLAN network.

Each *patch* element identifies a single connection using two attributes; the *destEndPointLocator* attribute and the *srcEndPointLocator* attribute.

- The *destEndPointLocator* attribute holds information that identifies a particular destination plug. *Listing 2.2* shows a *patch* element with *destEndPointLocator* attribute value of :

  "NODE_GUID='0013f00400400011', MLAN_PLUG_ID='Audio In 1'"

  Where the *NODE_GUID* variable holds the GUID of the destination device (*0013f00400400011*) and the MLAN*_PLUG_ID* variable holds the plug name (*Audio In 1*) of the destination plug receiving audio from the source plug.

- The *srcEndPointLocator* attribute specifies the information that identifies the source plug such as:

  "NODE_GUID='0013f00400000014', MLAN_PLUG_ID='Audio Out 2'"

  Where the *NODE_GUID* variable holds the GUID (*0013f00400000014*) of the source device transmitting audio and the *MLAN_PLUG_ID* attribute holds the textual name of the source plug (*Audio Out 2*).

2.2.5 The *session* element

The *session* element holds session information for managing time-based sessions, and is not used in the current implementation of mLAN.

2.3 Chapter Summary

This chapter discussed the current status of mLAN and its Enabler/Transporter and mLAN Client/Server architectures. mLAN is defined as a networking technology that is based on Firewire, which allows the transport of audio and music control data between audio devices. Firewire was chosen as the base networking technology for mLAN for the following reasons:

- It has low latency and determinism required by a network carrying real-time data.
- It has power-carrying capabilities and supports hot-plugging and plug-play performance capabilities which eliminate the need of a controlling host workstation

mLAN was chosen for this investigation because most of its development was done by the Rhodes University Audio Engineering Group in collaboration with Yamaha Japan. This provided the researcher with easy and cheap access to mLAN resources. Furthermore, the researcher had the opportunity of learning from mLAN developers themselves. mLAN, using the Enabler/Transporter architecture [*section 2.1*] supports true end-to-end connection management with reasonable Quality of Service (QoS). Data formats and hardware, such as the PH1, the NC1 and the PH2 chips, have been developed for sequence encapsulation and extraction in mLAN networks. Collectively these components described in this chapter ensure effective connection management in mLAN networks and form the basis for this investigation. The mLAN Client/Server architecture using XML messages enables the decoupling of client applications from the underlying Enabler module. Therefore, mLAN clients can be developed in any language and deployed on mLAN networks as long as they can communicate with the server over TCP/IP.

The next chapter surveys current connection management applications for three sound installation networks (the Broadcast networks, the Project studio networks and the Hospitality/Convention Centre networks).

CHAPTER 3

3 Connection Management Applications for Sound Installations Networks

A connection management application (patchbays) can be regarded as the nerve centre for audio signal routing within high-speed audio/video networks [Axia Audio, 2007]. It facilitates connection management by allowing the user to perform many signal routing tasks on a single application. This presents the user with centralized signal control capabilities in small personal studios and large professional audio networks. Tasks that can be performed on a patchbay application include:

- Establishing and breaking audio connections between device plugs.
- Setting and clearing word clock Master/Slave configurations.
- Viewing network topology at different levels (Network, Bus and Device levels).
- Saving/loading routing settings to/from a text file.
- Enabling the sound engineer to edit device properties such as device and plug names.
- Enabling the sound engineer to create or delete user accounts as well as manage resource distribution within the audio network.
- Enabling network users to book devices for their private use.

Connection management applications have been developed and deployed for various sound installation networks, each having different audio routing requirements. For purposes of this investigation, three sound installation networks were identified, and these include:

- Broadcast networks.
- Project studio networks.
- Hospitality/Convention Centre networks.

The following sub-sections discuss in detail each of these sound installation networks.

3.1 Types of Sound Installation Networks

3.1.1 Broadcast Networks

Broadcast networking solutions involve a high degree of complexity. In Broadcast studios, digital audio/video is distributed as a stream of digital data bits over a single or combination of two or more networks. As a result, Broadcast networks span large areas and involve the use of complex mixers and bridge units to route various kinds of audio/video data between audio devices and/or networks. Network Administrators for these networks are usually highly skilled sound engineers who understand how the audio/video data is routed on the network. These networks involve patching of many plugs, therefore requiring a complex connection management application that can expose as many plugs as possible for effective connection management.

3.1.2 Project Studio Networks

Project studio networks are much smaller than Broadcast networks, which tend to be distributed in nature. Project studio networks are usually located in one building or one room with only one administrator, and relatively fewer devices. These networks usually consist of only one network (routing only one type of audio or video data) while Broadcast networks combine one or more audio networks. It follows therefore that they typically deal with fewer connections. Due to the small nature of Project studio networks, sound engineers for these networks tend to develop a mental model of the layout and topology of their studio. This is because there are shorter distances between devices to be connected or disconnected. A the result, network administrators for these networks tend to physically switch the cables connecting network devices to perform audio routing. Sound engineers for these networks are usually music producers, who have enough technical knowledge of their studio network to route audio while Broadcast studio sound engineers are not necessarily music producers but specially trained audio network administrators.

3.1.3 Hospitality/Convention Centre Networks

Hospitality/Convention Centre networks are the least complex of the three sound installation networks, and can be operated with minimal experience and training.

They are usually operated by inexperienced users who have no time to spend learning or understanding the inner workings of the network, and how audio is routed and transmitted by the network. All they are concerned about is performing the audio routing tasks on the patchbay without having to know how these tasks are implemented on the network. As a result, connection management applications for these networks should hide the physical implementation of the network that is involved in the routing of audio and expose only devices to be connected in a manner that can be understood by an inexperienced user.

3.2 Current non-mLAN Client/Server Connection Management Applications

mLAN-based and Non-mLAN connection management applications have been developed that do not utilise the mLAN Client/Server architecture described in *section 2.1*. Traditionally, these are predominately developed using third-generation languages such as C, C++ and C#. Three categories of non-mLAN Client/Server connection management applications can be identified, and they include:

- Grid-based patchbays.
- List-based patchbays.
- Graphic-based patchbays.

The following sections discuss these connection management application categories in detail, and give current examples for each.

3.2.1 Grid-Based Patchbays

Grid-based patchbays are the most complex. They display devices and their plugs as well as visually display their connection status. Grid-based patchbays combine a tree-like structure (collapsible/expandable tree nodes) to display devices on the audio network, and a grid-matrix that displays a visual representation of the connection status of individual plugs on the network (whether a plug is connected, disconnected or not usable). Grid-based patchbay examples include:

- The OTARI mLAN Control Software.

- The CobraNet™ Manager.
- The Digigram's ESControl Management Software.

3.2.1.1 OTARI mLAN Control Software

Figure 3.1 displays a screenshot of the OTARI mLAN Control Software for Otari ND-20B networks that was developed by Otari using C++. It displays five Otari ND-20B devices, namely *mLAN ND-20B (54) – (1)*, *mLAN ND-20B (57) – (2)*, *mLAN ND-20B (59) – (3)*, *mLAN ND-20B (FE) – (4)* and *mLAN ND-20B (FF) – (5)*[*Figure 3.1*]. OTARI (2001) defines an OTARI ND-20 unit as a "network distribution unit which provides a networking solution for audio signals". Multiple OTARI ND-20 units can be connected via an IEEE 1394 network to convert supplied audio signals into different formats or distribute audio signals within the network. OTARI ND-20 units support 24-bit quantization and 96 kHz sample rate therefore can be used as a high quality analogy-to-digital and/or digital-to-analogy converter, or a digital signal sample rate converter. It has a maximum of 32-channel audio capacity (16 channels at 96 kHz) and a maximum of 96 channels for the entire ND-20 network system [OTARI, 2001].

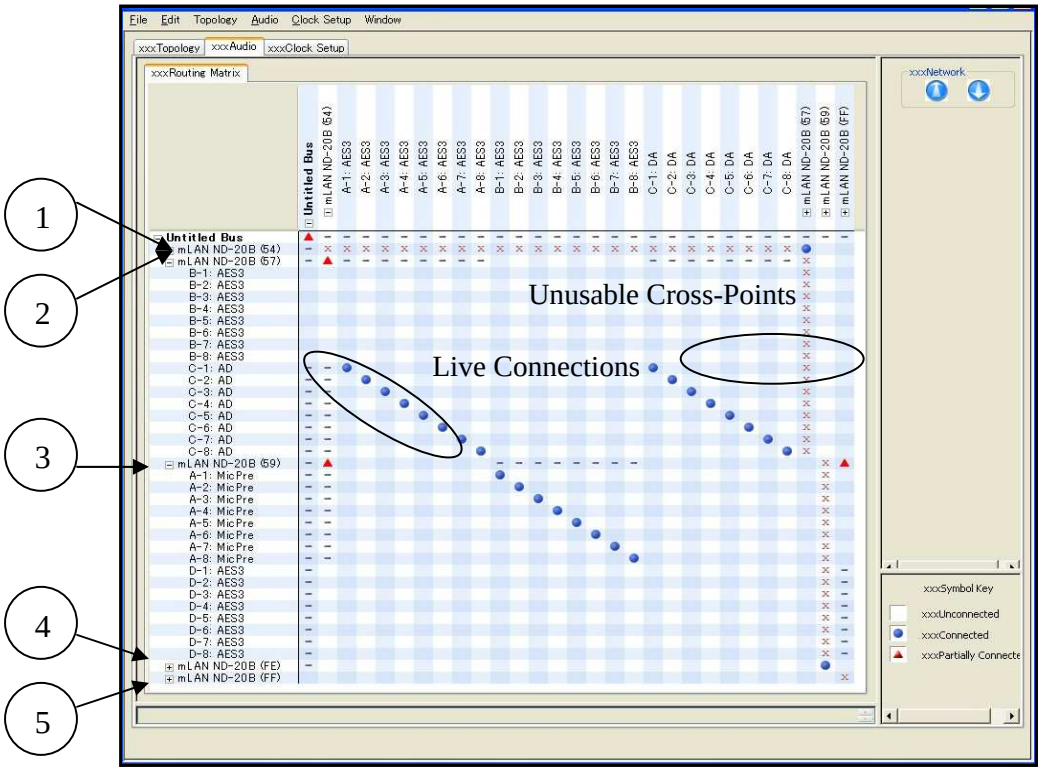

Figure 3.1: OTARI mLAN Control Software [OTARI, 2005]

The OTARI mLAN Control Software application comprise of three panels on which connection management tasks are performed. These include:

- The Topology pane.
- The Audio Routing Matrix pane.
- The Clock Setup pane.

a) Topology Pane

The *Topology* pane enables the sound engineer to view the layout of the IEEE 1394 network at three different topology levels, namely the Network level, the Bus level, and the Device level. Network level topology is displayed on the *Network* pane, which displays all buses and bridges connected to the host workstation. If there is no bridge unit on the IEEE 1394 network, only one bus is displayed on the *Network* pane. Bus level topology is shown on the *Bus* pane, which shows all devices on a particular bus on the IEEE 1394 network [*Figure 3.2*].

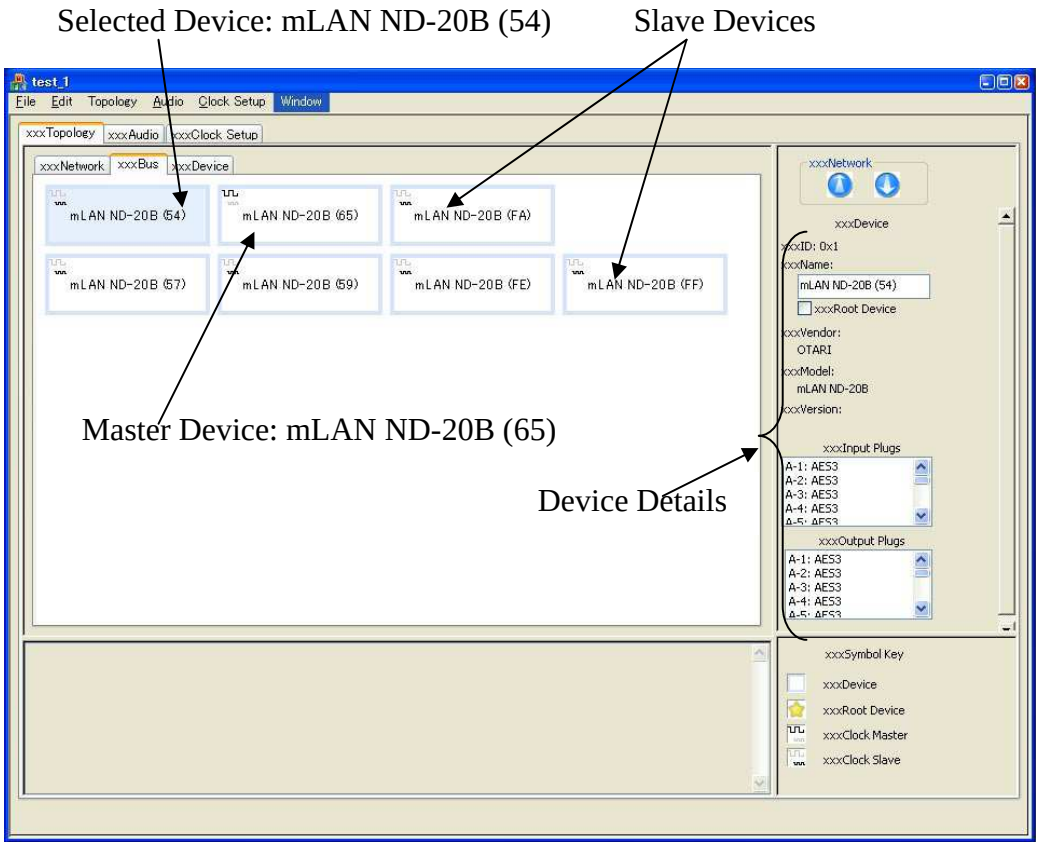

Figure 3.2: OTARI mLAN Control Software – Bus Pane

The number in parenthesis affixed to each device name is the Device ID. In *Figure 3.2*, the device *mLAN ND-20B (65)* is configured as the word clock master for the network, whilst all other devices are slaves. The right hand section of *Figure 3.2* displays the *Symbol Key*, and the selected *Device Details*, which shows information for a particular device, which is selected on the bus *Topology* pane. The *Device Details* section currently shows details for the device *mLAN ND-20B (54)*, which is selected on the display. Device information displayed includes:

- Its device name – mLAN ND-20B (54).
- Its device vendor – OTARI.
- Its device model – ND – 20B.
- Its device Input plugs – A-1: AES3 to A-5: AES3.
- Its device Output plugs – A-1: AES3 to A-5: AES3.

The *Device* pane displays input and output plugs for a particular device on the IEEE 1394 network [*Figure 3.3*].

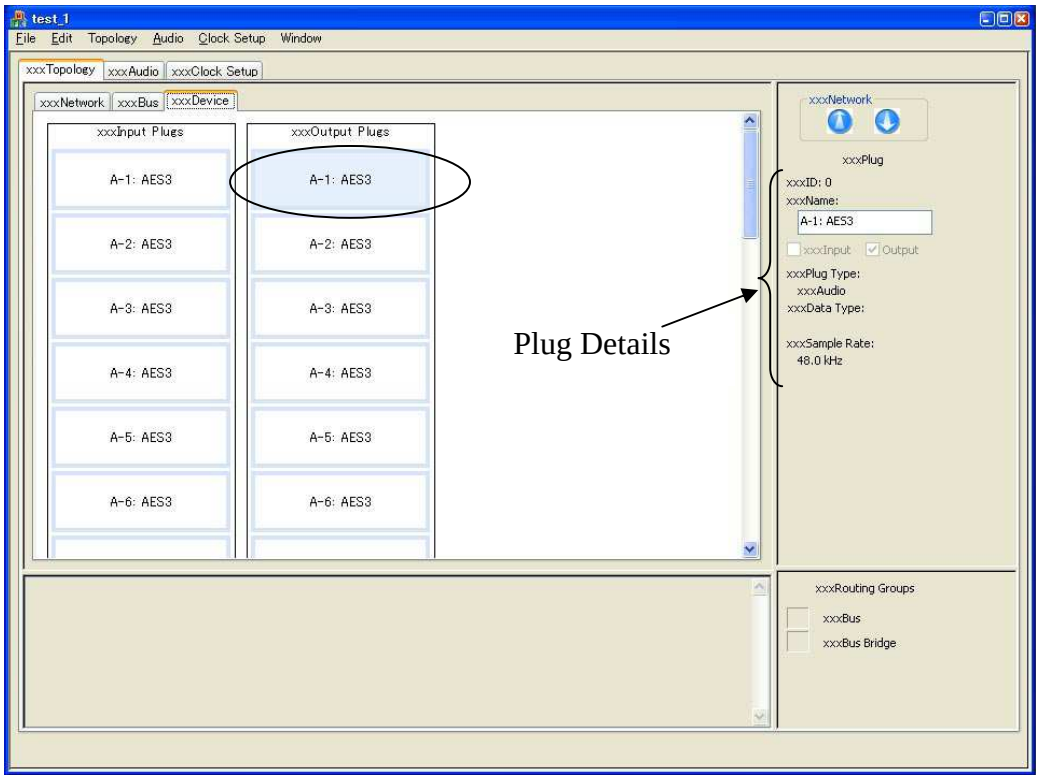

Figure 3.3: OTARI mLAN Control Software – Device Pane [OTARI, 2005]

If a plug is selected on the main display of the *Device* pane (i.e. *A-1: AES3* plug in the *Figure 3.3* --circled), its information is displayed on the right section of the pane (the *Plug Details*). *A-1: AES3* plug details displayed include:

- Its plug name – A-1: AES3.
- Its plug type – audio plug and is an output plug.
- Its sample rate – 48.0 kHz.

b) Audio Routing Matrix

The Audio Routing Matrix is used for setting audio routings. It displays IEEE 1394 devices and their associated plugs in two trees [*Figure 3.1*]. At the centre of the application is a grid-matrix that displays the connection status of IEEE 1394 device plugs, on which connection management is performed. The tree list on the left side of the matrix lists input plugs while the tree list shown above the matrix lists the output plugs. Plug cross-points with blue round dots represent live connections [*Figure 3.1*]. For example, inputs plug *C-1: AD* on device *mLAN ND-20B (57)* is connected to output plug *A-1: AES3* on device *mLAN ND-20B (54)*. The red "x" icons represent the cross-points on which connections cannot be made. For instance, red "x" icons are shown at the device level column of the outputs tree for the device *mLAN ND-20B (57)* (shown circled in *Figure 3.1*) and its open plugs on the input plugs tree because a plug cannot be connected to a device but only to another plug, and plugs of the same device cannot be connected to each other. The "–" icon denotes a point connecting different layers such as a bus to a unit, or a unit to each I/O plug [*Figure 3.1*].

Audio connections and disconnections are made by simply clicking on the cross-point of two plugs to be connected or disconnected. Connections and disconnections are applied by clicking the *Apply* button [*Figure 3.4*].

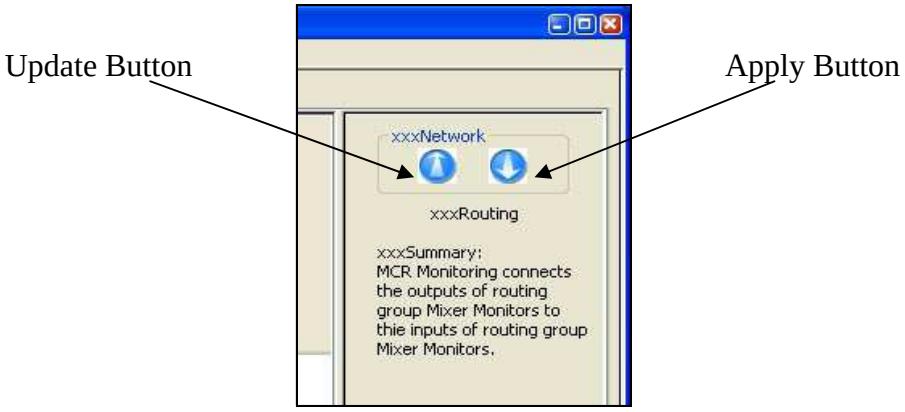

Figure 3.4: Update and the Apply Buttons
[OTARI, 2005]

The *Update* button is used to force a process that gets the latest information about the network devices, and plugs connection states. Although working with the Routing Matrix is easy once a user is oriented with it, it does not clearly show the user which tree list contains output nodes or which tree list contains input nodes. By using a combination of colours and shapes to represent different connection states of the device plugs, the Audio Routing Matrix makes it easy for the sound engineer to see which plugs are connected. Collapsible trees allow for the viewing of many devices at a time.

c) Clock Setup Pane

Figure 3.5 shows the *Clock Setup* pane in which word clock Master/Slave configurations are performed on the OTARI mLAN Control Software.

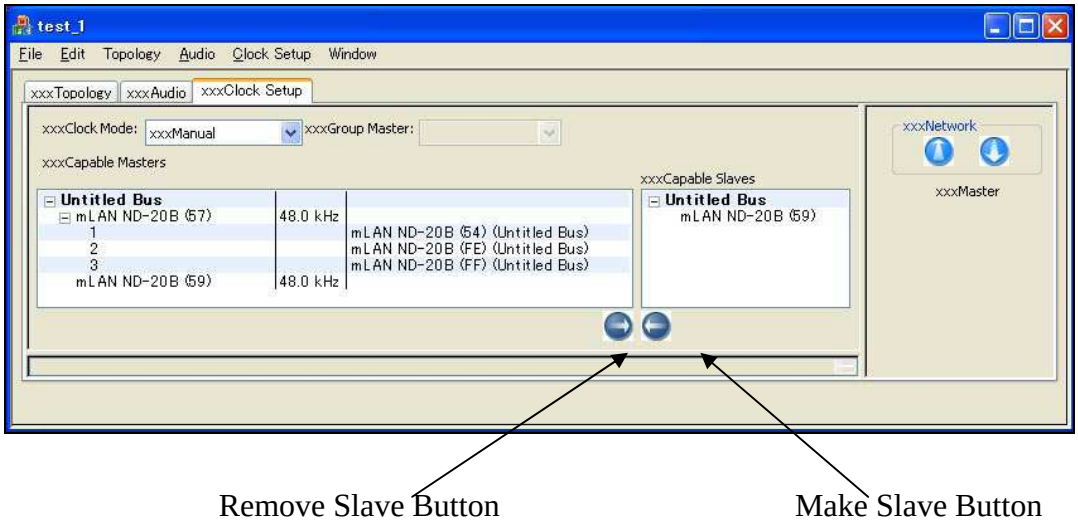

Figure 3.5: OTARI mLAN Control Software – Clock Setup Pane
[OTARI, 2005]

The *Word Clock* pane has two main sections [*Figure 3.5*], namely the “Capable Masters” section, and the “Capable Slaves” section. The “Capable Masters” section displays devices that can be configured as master devices and the “Capable Slaves” section displays devices that can only be made slaves. Devices that are not currently set as a slave or a master are displayed in the middle column between the “Capable Masters” column and the “Capable Slave” column. To set a word clock Master/Slave configuration, the sound engineer clicks and selects a device to be made a master on the “Capable Masters” field, and then clicks and selects as many slave devices as possible from the “Capable Slaves” field. Clicking the *Make Slave* button makes sure newly set slave device nodes appear under their master device on the “Capable Masters” field. If the sound engineer is satisfied, clicking the *Apply* button would apply the configuration to the network.

To clear a word clock Master/Slave configuration, the sound engineer simply clicks and selects the master device to be cleared and clicks the *Remove Slave* button. When the *Apply* button is clicked, the slave setting is cleared and applied to the network.

3.2.1.2 CobraNet™ Manager

The CobraNet™ Manager software, like the OTARI mLAN Control Software, displays the current status of the CobraNet™ network using a combination of two device tree lists and a grid-matrix. CobraNet™ is a standard developed by Peak Audio that is owned by Cirrus Logic Inc [Cirrus Logic, 2007]. It is based on the hardware and protocol-layer of traditional Ethernet networks [D and R Electronica B. V., 2007]. Benefits of using CobraNet™ include:

- Its transportation of high fidelity, uncompressed digital audio in real time over off-the-shelf, switched Ethernet networks.
- Its use of standard Ethernet hardware and equipment, which guarantees cost effective digital networking as no new infrastructure need to be installed to use it.
- Its use of SNMP (Simple Network Management Protocol) combined with numerous fault tolerant features provides reliability, powerful control and monitoring of the network and its devices.

Three packet types make up the CobraNet™ protocol, namely *beat, isochronous data* and *reservation packets* [Cirrus Logic, 2007]. The *beat* packet is directed at a multicast address and carries the network operating parameters, clock, and transmission permissions. Only one device on the network transmits beat packets, but all devices on the network are required to listen for these packets [Okai-Tettey, 2005]. The *isochronous* packet is the carrier of audio data and is subdivided into bundles. A bundle is the smallest network audio routing envelope, with a capacity to transmit up to 8 audio channels. *Table 3.1* summaries CobraNet™ capabilities.

Table 3.3: CobraNet™ Capabilities

Capability	Description
Latency	2.66 ms and 1.33 ms with low latency mode, 5.33 ms otherwise.
Number of channels	Up to 64 channels of 20-bit audio at 48 kHz sample rate.More channel count with 16-bit audio and less with 24-bit audio.
Network transport	CAT-5 cabling. Daisy chain and star
Control and monitoring	CobraNetTM Manager developed by D&R Electronica
Open standard/Proprietary based	Proprietary based, but uses a standards based transport

Figure 3.6 shows the CobraNet™ Manager software interface. Each device displayed on the CobraNet™ Manager software has 4 receiving sockets and 4 transmitting sockets, and a reference number [*Figure 3.6*]. These form the connection points for connection management on the matrix. CobraNet™ Manager Software makes audio connections by setting-up bundle numbers. An audio connection is made between a receiver and transmitter if they have the same bundle numbers set. Bundle numbers range between 1 and 65535. Different manufacturers/products use different ways of manipulating these bundle numbers [*Table 3.4*].

Table 3.4: Ways of Setting and Manipulating Bundle Numbers

Manufacturer/Product	Method for controlling bundle numbers
D&R Lyra	Use a webpage
Rane NM48/NM84	Use an encoder
Yamaha NHB32-C/ACU16-C	Use USB with an external computer
Crown PIP	Use their own IQ software
Others	Use DIP Switches

Bundle numbers can also be automatically assigned to devices when the sound engineer clicks on a cross-point of two sockets to be connected on the matrix. The cross-points are prepared using SNMP messages [D and R Electronica B. V., 2007]. *Figure 3.6* shows an example of this concept for two cross-points for sockets *Tx2* on device *USB Mixer* and *Rx2* on device *Lyra* and sockets *Tx2* on device *Yamaha NHB-32USB Mixer* and *Rx3* on device *Lyra*.

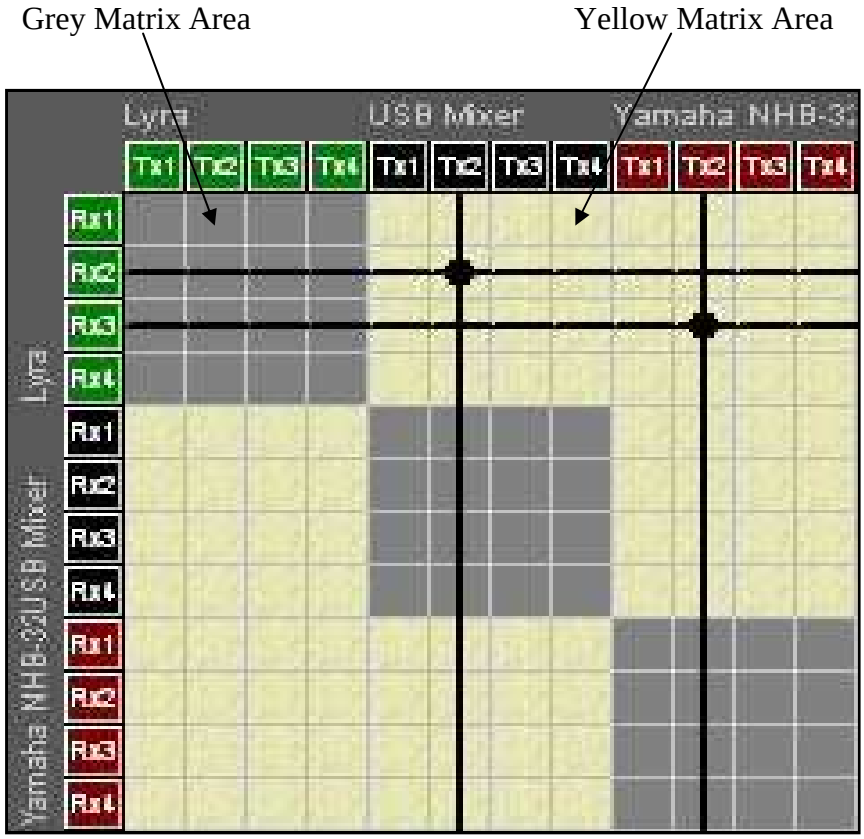

Figure 3.6: CobraNetTM Manager
[D and R Electronica B. V., 2007]

The *Gray Matrix Area* on the routing matrix depicts cross-points for sockets that cannot be connected because they are on the same device (internal audio routing is not allowed). The *Yellow Matrix Area* represents cross-points for sockets that can be connected. There are three types of connections that can be made on the CobraNet™ Manager software [D and R Electronica B. V., 2007]:

- Multicast connections - These involve a single transmitter transmitting to many receivers.

- Unicast connections - These involve a single transmitter transmitting to only one receiver using bandwidth on a dedicated link.
- Private connections - These involve a private receiver only receiving audio from a known transmitter.

The CobraNet™ Manager software uses a built-in scheduler to plan connections that are applied to the network based on the time they are scheduled to run. A user can configure a cross-point of two sockets to hold a multicast, private, unicast and secure connection events by simply right-clicking the required cross-point and selecting the correct menu item. CobraNet™ Manager software also allows the user to define custom plug-in functions and *macros* that can be triggered from the scheduler and/or from any plug-in. Each macro can hold one or more events. The sound engineer uses the Macro Event List to add, delete and change macro events. There are three types of events that can be added onto a Marco Event List:

- SNMP Events – These are events that can be triggered by SNMP messages.
- Cross-Point Events – These are events set on a cross-point of two sockets.
- Plug-in Functions – There are events triggered by external plug-in functions.

3.2.1.3 ESControl Management Software

The ESControl Management Software [*Figure 3.7*] is a Client/Server application for connection management in Digigram's EtherSound range of products that include EtherSound ES8in, ES8mic, ES881, ES1241, ES16161, LX6464ES and miXart 8 ES [Digigram SA, 2007a].

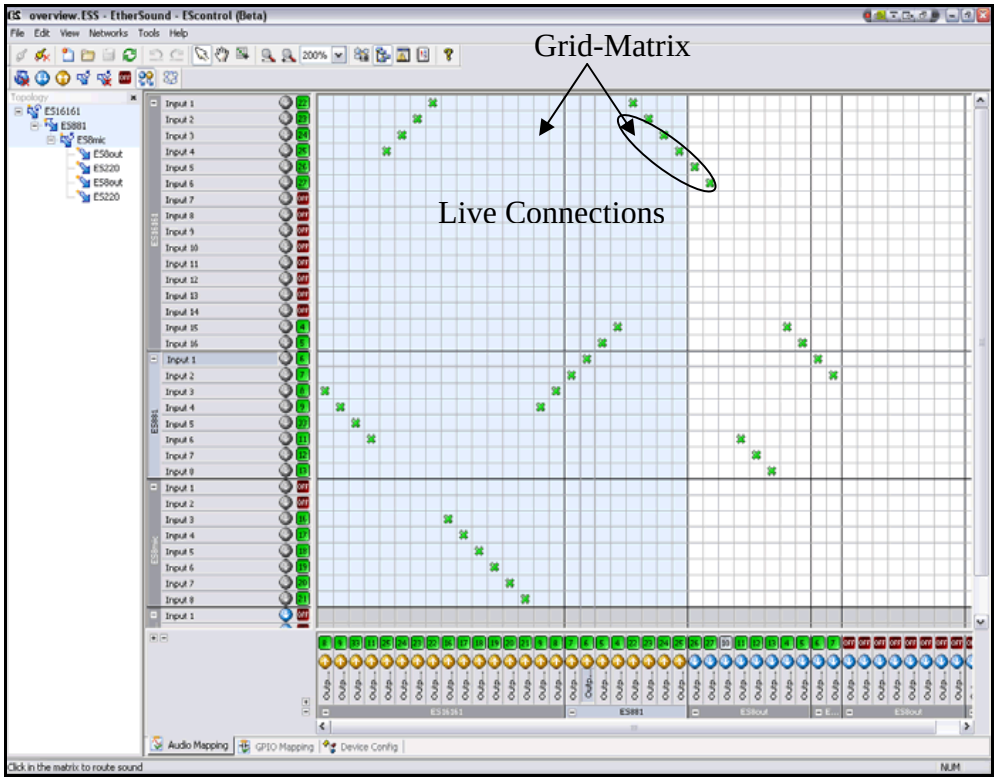

Figure 3.7: ESControl Management Software
[Digigram SA, 2007a]

Like the OTARI mLAN Control Software and the CobraNetTM Manager Software discussed in *sections 3.3.1.1* and *3.3.1.2* respectively, the ESControl Management Software combines two tree lists and a grid-matrix on which connection management is performed. The ESControl Management Software Server application uses a non-dedicated Ethernet to connect to the EtherSound network it controls. The ESControl Management Software also incorporates a multi-client application with the following features and capabilities [Digigram SA, 2007a]:

- A matrix interface [*Figure 3.7*].
- Automated network discovery capabilities.
- Save network-wide EtherSound configurations for instant reload.
- Supports remote system management and diagnosis using TCP/IP to connect to the server.

To establish audio connections, the sound engineer simply clicks the cross-point of two plugs to be connected [*Figure 3.7*]. The green "x" icons on the grid-matrix

represent a live connection. Audio connections can be broken by clicking a cross-point with a green “x” icon for the plugs to be disconnected.

3.2.2 List-Based Patchbays

List-based patchbays display devices and their associated plugs in a list structure. This means list-based patchbays display a large number of device nodes and plugs at a time, a feature that is required for large and complex audio networks. List-based patchbay examples include the Yamaha’s mLAN Patchbay and the PathfinderPC Router Control Software.

3.2.2.1 Yamaha mLAN Patchbay

Figure 3.8 shows the list-based Yamaha mLAN Patchbay for controlling audio routing between version 1 mLAN devices [Yamaha Corporation, 2004b]. It has three panels on which connection management tasks are performed, namely the *Audio* panel, the *MIDI* panel, and the *Word Clock* panel. The *Audio* tab, the *MIDI* tab, and the *Word Clock* tab are used to access these panels [Yamaha Corporation, 2004b, *Figure 3.8*]. The mLAN Patchbay *Audio* and *MIDI* panels display device plugs in two separate lists. The *From* list displays source plugs and the *To* list displays destination plugs. If a plug on the *From* list is connected, the connection information of the destination plug that is connected to it is shown directly opposite to it on the *To* list. If a plug on the *From* list is not connected, a “---” (hyphen) is displayed on the *To* list [*Figure 3.8*]. *Figure 3.8* shows that the output plug *AUX2* is connected to input plug *Input 1*, and is transmitting at 32 kHz.

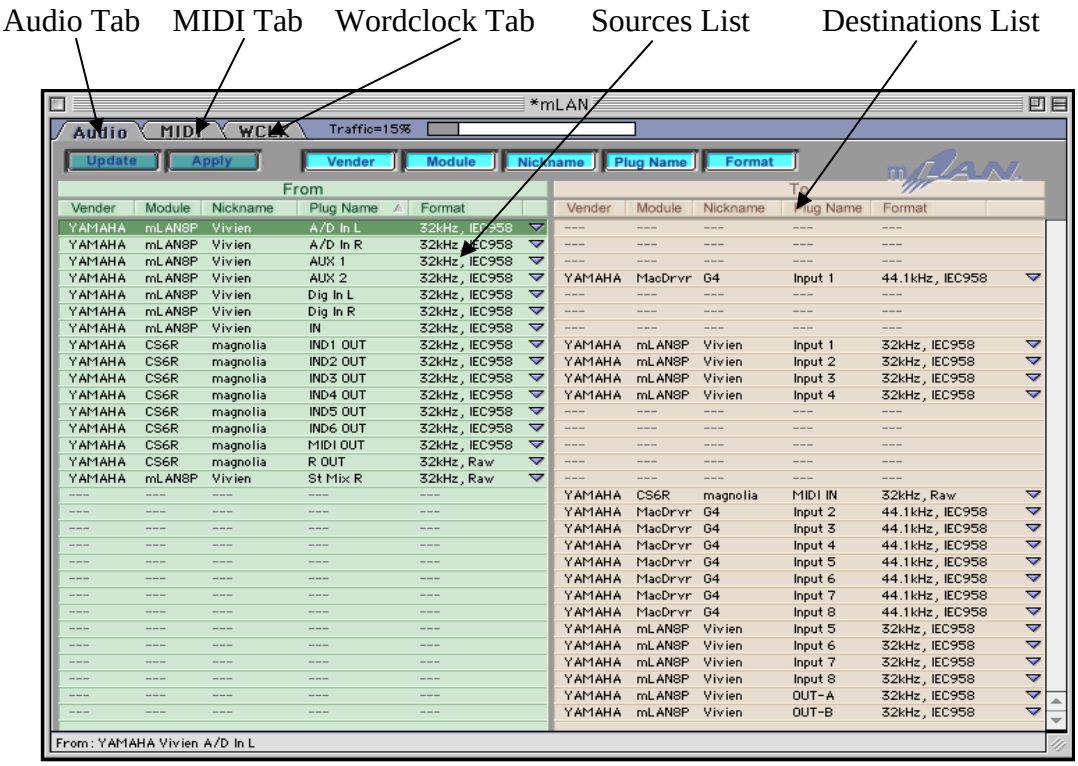

Figure 3.8: Yamaha mLAN Patchbay – Audio Page
[Yamaha Corporation, 2004b]

To establish audio connections on the Yamaha mLAN Patchbay, the sound engineer right-clicks the source plug on the *From* tree list. By traversing the submenus that appear, the engineer selects the desired destination plug. Output plugs that cannot be selected are disabled (shown greyed-out on the *Audio* panel [*Figure 3.8*]). Right-clicking areas that are displayed as "----" has no effect. If the connection is successful, the destination plug information immediately appears in the *To* column of the Yamaha mLAN Patchbay [*Figure 3.8*]

To break audio connection, the sound engineer right-clicks the output plug on the *From* section of the Yamaha mLAN Patchbay and selects the "Disconnect (T)" menu item on the submenu that appears. Word clock Master/Slave configurations are set on the *Word Clock* panel of the Yamaha mLAN Patchbay. To set the Master/Slave configuration, the sound engineer right-clicks the device to be made slave on the Slave section of the *Word Clock* panel [*Figure 3.9*]. On the submenus that appear, the user selects the proper master device for the selected slave. *Figure 3.9* shows this process for setting the master device *magnolia*.

Figure 3.9: mLAN Patchbay – Word clock Synchronisation [Yamaha Corporation, 2004b]

3.2.2.2 PathFinderPC Software

The PathFinderPC software is a Client/Server Router control software package. It is designed to provide facility-wide control over any number of audio, video and Machine Control routers including the Axia Livewire distributed routing system [Axia Audio, 2007]. The software consists of a server module called the PathFinderPC Server that communicates via a serial port or Ethernet with the routers, and a client application called the PathfinderPC Client that is a graphical user interface on which sound engineers perform connection management tasks [Axia Audio, 2004, 2007]. The client application connects to the server using the TCP/IP protocol over any established Local Area Network or even over the Internet. The server application makes the client requested-changes to any of the routers in the routing system and provides updates to the client application interface. The PathfinderPC package also includes a tool called the Panel Designer and an additional application called the PathfinderPC Mini. The Panel Designer tool allows a designer to create custom software routing panels. These panels are then available in the PathfinderPC Client application for the end-user. The PathfinderPC Mini application is used in a situation where the only controls the end user should access are those available through one of the custom designed panels [Axia Audio, 2004, 2007].

Details of the PathFinderPC Server are beyond the scope of this investigation. However, additional information can be acquired from the PathfinderPC manual [Axia Audio, 2004, 2007]. The PathfinderPC client application includes the following features and capabilities [Axia Audio, 2007]:

- It has three different routing views and methods for viewing and for making route changes.

- It has a simple interface for creating and activating entire scene changes to quickly change multiple routes in the system.
- It has the ability to lock routes so that they may not be changed by another user during a show.
- It has resource sorting capabilities.
- It has a *Virtual Router Creation* panel to create custom route lists for a particular room and/or to tie points together from different kinds of routers.
- It has an Event Programming Interface for creating standard events.

The PathfinderPC client is used to make routes, create and edit virtual routers, create, edit, and activate scene changes, as well as schedule events. It also has a search engine for finding route points and scenes throughout the entire routing system. At start-up, a *Server* details dialog box opens that allows the sound engineer to specify the IP address or a fully qualified name of the computer running the server application as well as the TCP/IP socket number of the server application (the default socket number is 5200). When the sound engineer clicks the *Chart View* under the view menu, a graphical grid view of the routing status is displayed [*Figure 3.10*]. Green dots represent cross-points with live connections which have streaming audio. Grey dots represent cross-points with live connections but with no streaming audio.

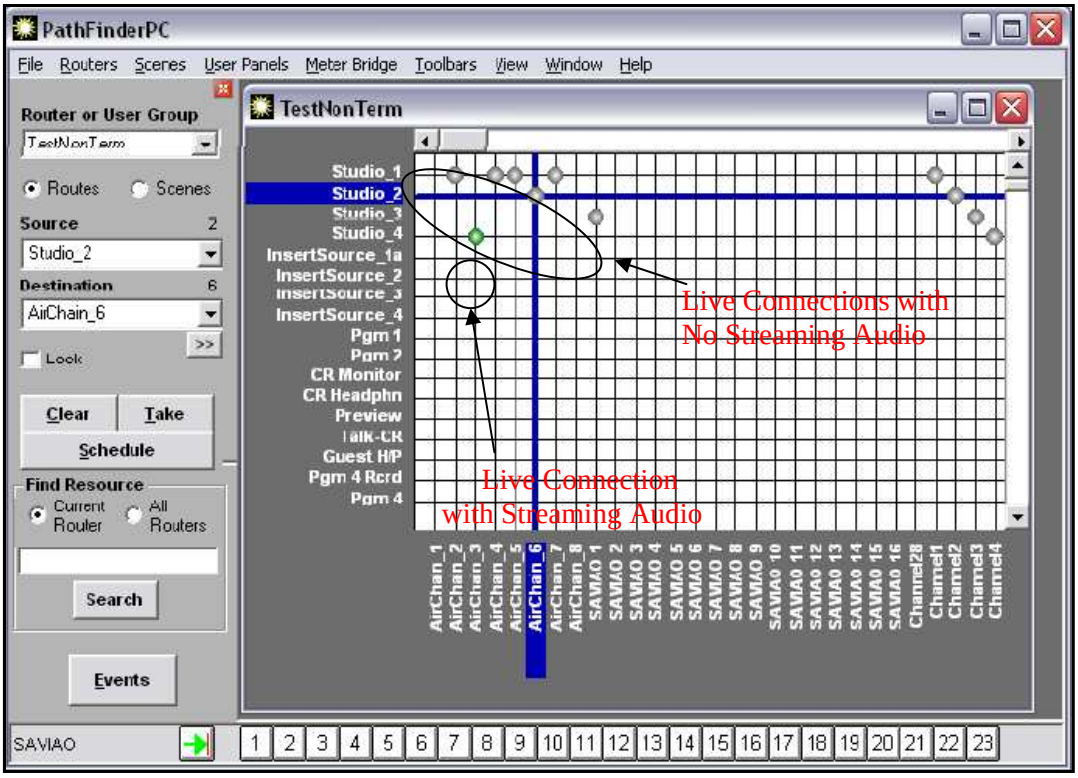

Figure 3.10: PathfinderPC Router Control Software
[Axia Audio / TLS Corporation, 2007]

3.2.3 Graphic-Based Patchbays

Graphic-based patchbays provide a pictorial representation of the devices and plugs on the audio network. The Yamaha's mLAN Graphic patchbay for version 2 devices is an example of graphic-based patchbays.

3.2.3.1 Yamaha mLAN Graphic Patchbay

The Yamaha mLAN Graphic patchbay enables the user to setup and manage connections between mLAN devices using a graphic interface to connect and disconnect virtual audio/MIDI connectors [Yamaha Corporation, 2004a]. C++ was used for developing the Yamaha mLAN Graphic Patchbay. *Figure 3.11* shows a screenshot of the Yamaha mLAN Graphic patchbay that displays three mLAN devices on the IEEE 1394 network:

- The mLAN Windows PC.
- The O1x.
- The MOTIF ES7.

Device graphics can be dragged around to specified locations within the Yamaha mLAN Graphic patchbay display. In *Figure 3.11*, the *O1x* device is dragged from the right-side of the application shown as a solid brown square (its original location) to the left-side of the application by simply clicking it, dragging and dropping it at one of the *Drop* locations designated by the brown empty squares.

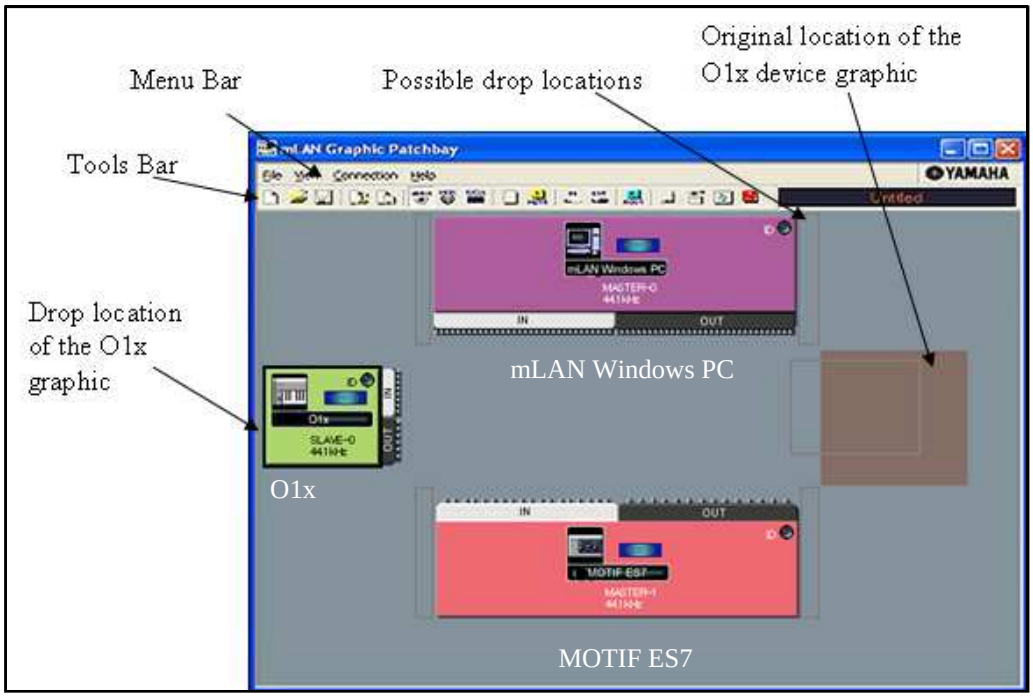

Figure 3.11: The Yamaha mLAN Graphic Patchbay
[Yamaha Corporation, 2004a]

The *Menu Bar* of the Yamaha mLAN Graphic Patchbay contains various editing and setup function menu items. *Menu Bar* items can be assessed by clicking the desired menu name to display the pull-down menu, and then choose the appropriate menu item to apply [Yamaha Corporation, 2004a]. *Figure 3.12* shows the *Tool bar* of the Yamaha mLAN Graphic Patchbay that contains icons, which allow the sound engineer to use the same functions and commands that can be accessed from the *Menu bar*.

Figure 3.12: mLAN Graphic Patchbay – Tool bar
[Yamaha Corporation, 2004b]

Tool bar icons enable access to capabilities such as:

- Opening saved routing settings (1).
- Saving routing settings (2).
- Viewing either MIDI or Audio plugs (3).
- Setting Master/Slave configurations (4).
- Refreshing the mLAN Graphic Patchbay (5).
- The help documentation (6).

To establish audio connections using the Yamaha mLAN Graphic Patchbay, the sound engineer opens the "Outputs Connector Window" of the source device and the "Inputs Connector Window" of the destination device by clicking their *maximise* buttons. These display the input and output plugs in detail [*Figure 3.13*]. *Figure 3.13* shows the *MOTIF ES7* device and the *mLAN Windows PC* device to be connected. To make the connection between plugs, for example between the plug *Out 1* of the *MOTIF ES7* device and the plug *In 1* of the *mLAN Windows PC* device, the sound engineer clicks the box next to the output and input labels of the two plugs on the open "Output" and "Input Connector Windows".

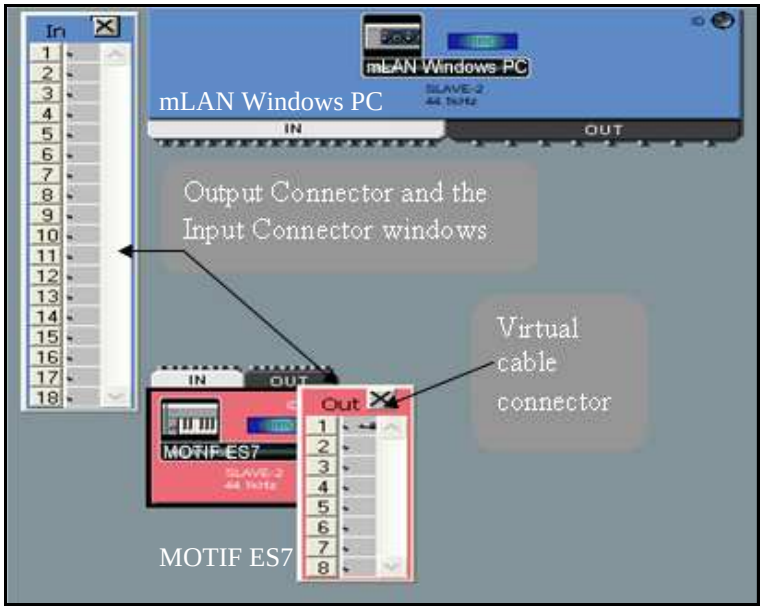

Figure 3.13: Yamaha mLAN Graphic Patchbay – Establishing Audio Connection [Yamaha Corporation, 2004b]

A virtual cable connector appears in the box [*Figure 3.13*] for the plug *Out 1* of the *MOTIF ES7* device. If the connection is successful, a coloured cable of the same colour as the output node is drawn to reflect the connection between the two plugs connected [*Figure 3.14*].

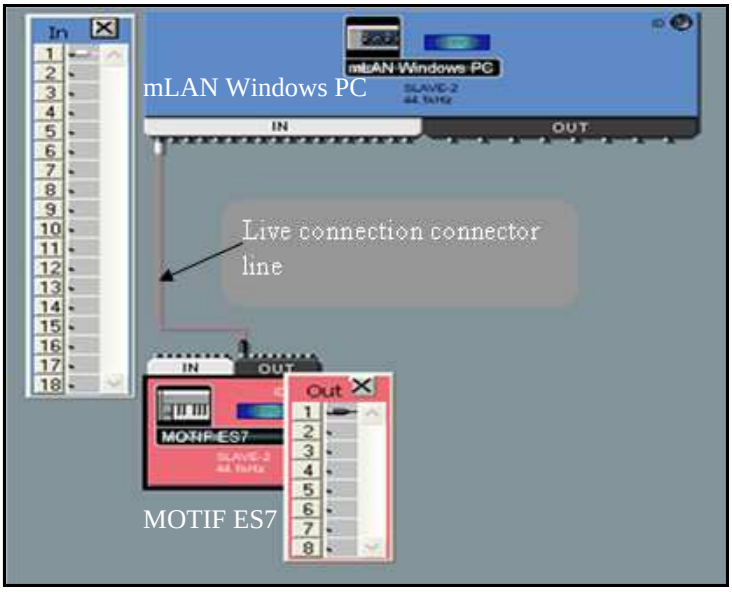

Figure 3.14: mLAN Graphic Patchbay – Successful Connection [Yamaha Corporation, 2004b]

Other connection management tasks that can be performed on the Yamaha mLAN Graphic Patchbay include:

- Breaking audio connections.
- Setting word clock Master/Slave configurations.

3.3 Current mLAN Client/Server Connection Management Applications

At the time of this investigation, only a Windows Explorer - style patchbay (known as the NAS Explorer patchbay – *Figure 3.15*) had been developed above the mLAN Client/Server architecture described in *section 2.2*. The NAS Explorer patchbay is a list-based connection management application that allows users to perform various audio patching tasks such as establishing audio connections or disconnections between output and input plugs of two or more transporter nodes, and setting of word

clock configurations [I/One Connects, 2007a]. The devices shown in the NAS Explorer patchbay screenshot [*Figure 3.15*] are two I/One audio breakout boxes, the *OGT - I/One Src* and the *OGT - I/One Dest*, and a Yamaha Corporation's *MAP4* board, as well as the host workstation, *mLAN Windows PC.* An I/One breakout box carries over a hundred audio channels with word clock and control data on one Firewire cable [I/One Connects, 2007b]. Each of these devices hosts an IEEE 1394 Node Controller that it uses to communicate with other devices over Firewire.

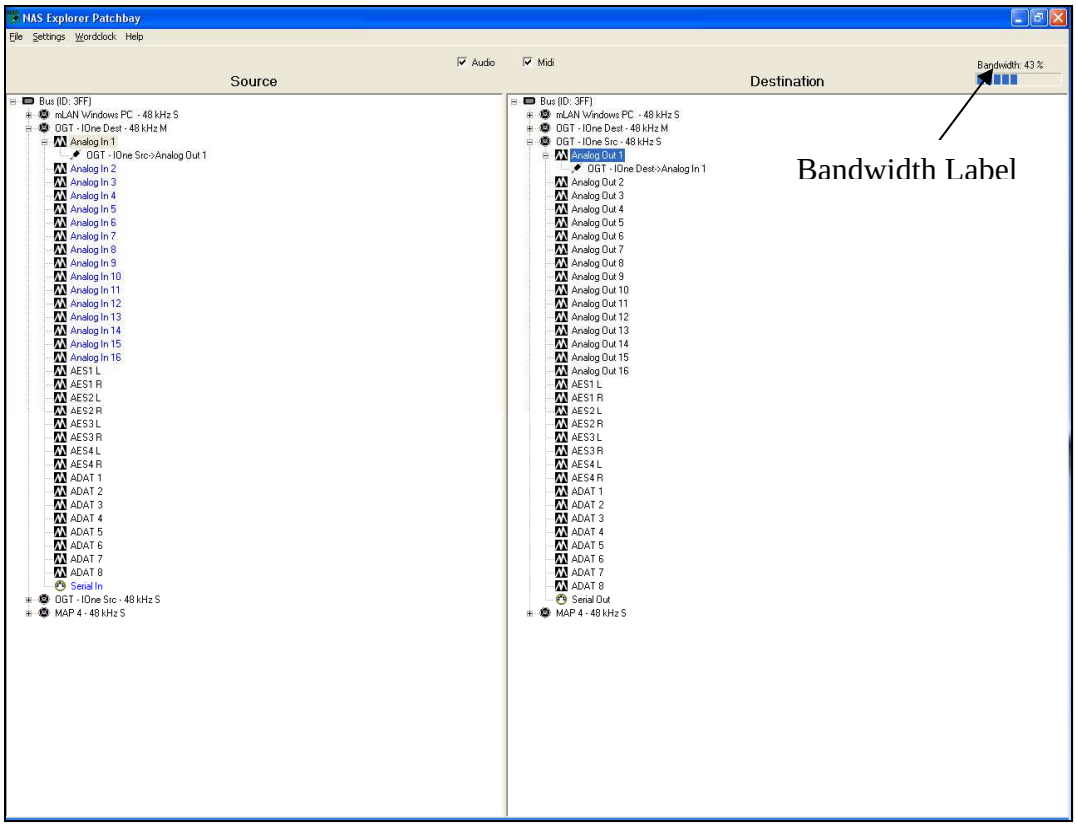

Figure 3.15: NAS Explorer Patchbay

The NAS Explorer patchbay uses the Client/Server architecture of the mLAN project and utilises XML messages as the communication protocol between it and the mLAN Connection Management Server (mCMS) server. In *Figure 3.15,* two separate sections of the NAS Explorer patchbay are visible. There is the *Source* section and *Destination* section. The *Source* section displays a tree list of output plugs of four nodes on the mLAN network, while the *Destination* section displays input plugs of the four nodes on the mLAN network. Using these two collapsible/expandable tree lists, the NAS Explorer enables sound engineers to view many devices in a simple and understandable way. Due to the NAS Explorer's simplicity, connection management

tasks are easily performed. For instance, to establish a connection between two plugs, the sound engineer clicks and expands the two device nodes whose plugs are to be connected on the *Source* and *Destination* trees (*OGT - I/One Src* and the *OGT - I/One Dest* devices - *Figure 3.15*). Once the trees are expanded, the sound engineer can select the plugs to be connected and right-click either of them.

In *Figure 3.16*, the sound engineer selected the output plug *Analog In 1* on the source device *OGT - I/One Src* and the input plug *Analog out 1* on the destination device *OGT - I/One Dest*. A *"Connect Analog In 1 --> Analog Out 1"* submenu will appear [*Figure 3.16*] that the sound engineer clicks to establish the connection between the two plugs. The request is implemented automatically without any further involvement of the sound engineer if there is enough bandwidth on the network otherwise an error message dialog box pops up notifying the user of the problem. If the connection is successful, audio should be routed from the output plug *Analog In 1* on the source device *OGT - I/One Src* to the input plug *Analog Out 1* on the destination device *OGT - I/One Dest*.

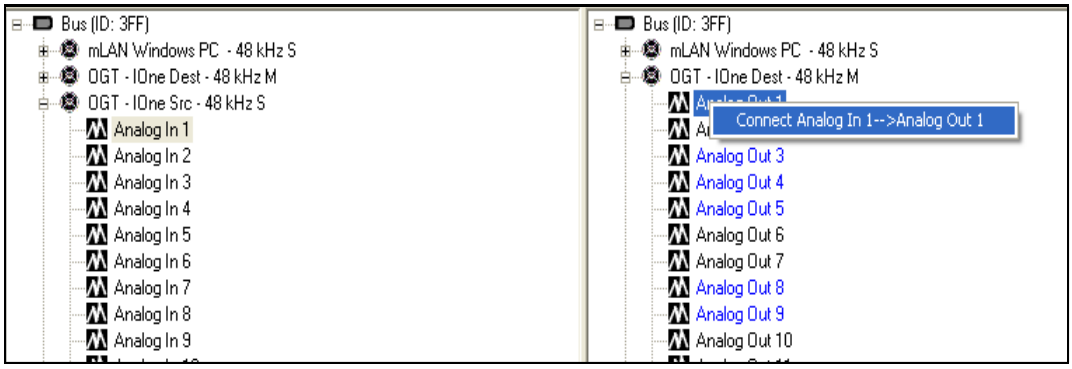

Figure 3.16: NAS Explorer Patchbay - Establishing Audio Connections

Breaking audio connections is performed by right-clicking one of the plugs to be disconnected, either on the *Sources* tree or *Destination* tree and clicking the "Disconnect OGT - I/One Src -> Analog Out 1" submenu that appears [*Figure 3.17*]. This command will break the audio connection between *Analog Out 1* on device *OGT - I/One Src* and *Analog In 1* of device *OGT - I/One Dest*.

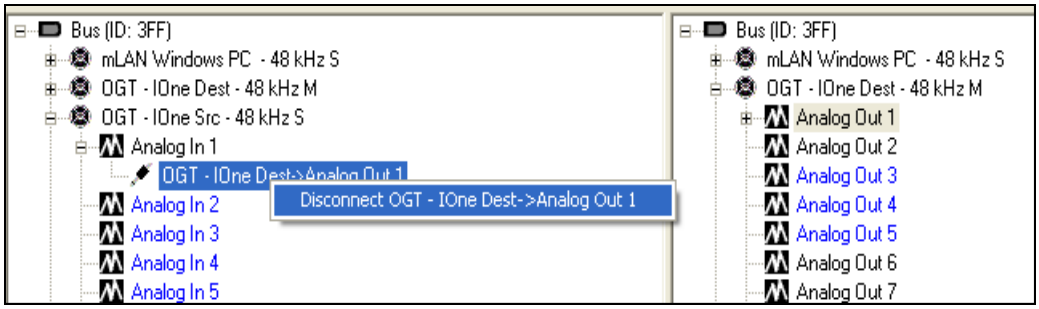

Figure 3.17: NAS Explorer Patchbay - Breaking Audio Connections

Dangling connections can be cleared by simply right-clicking the plug with the dangling connection. A "Clear Dangling Connection" submenu item will appear that the sound engineer clicks to clear the dangling connection [*Figure 3.18*].

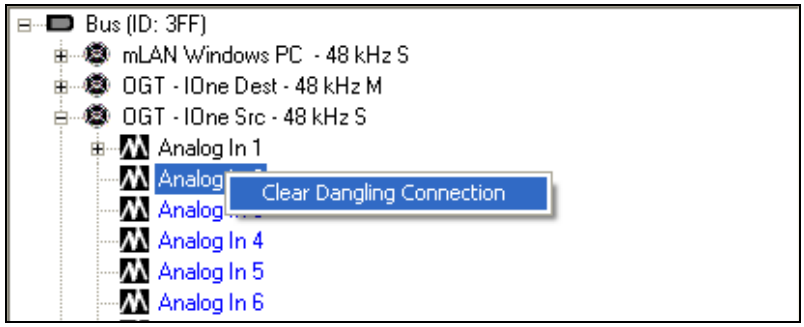

Figure 3.18: NAS Explorer Patchbay - Clearing Dangling Connections

A dangling connection is a destination-less routing left only at the sending or receiving side of two communicating devices. This may occur on the receiving device plug if the corresponding source device it is receiving from is removed from the network without performing a proper disconnection. The opposite is also true in which case the dangling connection would occur on the source plug.

Setting word clock Master/Slave configurations is done on the *Wordclock* panel [*Figure 3.19*]. To set a global master the sound engineer right-clicks the device to be made a global master on the *Available Wordclock Masters* section of the *Wordclock* panel. A submenu will appear with the menu item "Make Global Master" that the user clicks to commit the setting to the mLAN network. There is also the option of selecting the word clock source and the sample rate of the device to be a master.

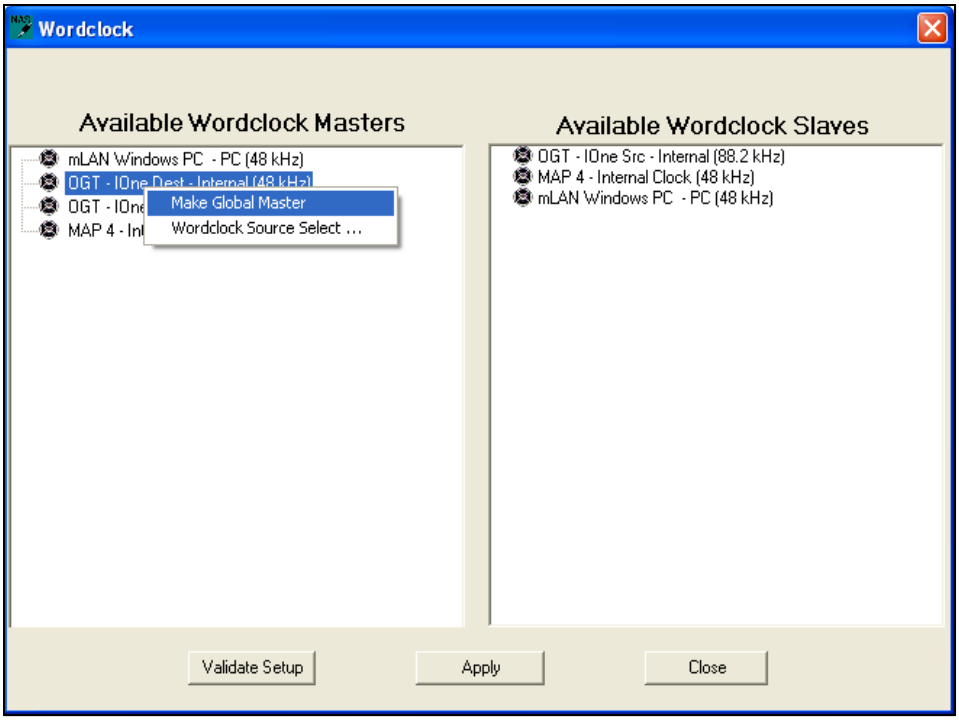

Figure 3.19: NAS Explorer Patchbay - Setting Master/Slave Configurations

The NAS Explorer patchbay also displays the amount of bandwidth available on the mLAN network. *Figure 3.15* shows that at the time this snapshot was taken there was 57% available bandwidth. The sound engineer operating the NAS Explorer application has an option of viewing either Audio or MIDI plugs individually or together by simply unselecting or selecting the appropriate check box on the top-middle section of the NAS Explorer application. Additional functionality that can be performed on the NAS Explorer patchbay includes:

- Renaming mLAN devices on the network.
- Modifying the server port number and its DNS name.
- Saving network routings status into a file.
- Opening saved settings.

Although the NAS Explorer patchbay can serve the needs of small sound installation networks like simple Project studios, it can not be used in application areas. For instance, b simply looking at the trees, one cannot tell the connection status of a plug or which plugs are connected, except by physically expanding the individual plugs. This may prove to be disadvantageous in large and complex networks like in

Broadcast networks. Other patchbay designs need to be implemented that utilise the mLAN Client/Server architecture to deal with the diverse requirements in complex networks.

3.4 Chapter Summary

This chapter listed general audio routing requirements for high-speed audio networks that need to be satisfied by any reliable connection management application. Some of the important audio requirements include:

- Establishing and breaking audio connections between device plugs.
- Setting and clearing word clock Master/Slave configurations.
- Saving/loading routing settings to/from a text file.
- Enabling the sound engineer to edit device properties such as device and plug names.
- Enabling the sound engineer to create/delete user accounts as well as manage resource distribution within the audio network.
- Enabling network users to book devices for their private use.

This investigation identified three sound installations, namely the Broadcast networks, Project studio networks and Hospitality/Convention Centre networks for which patchbays are to be developed that utilise the mLAN Client/Server architecture discussed in *section 2.2*. Broadcast networks are the most complex of the three networks since they span large areas and are distributed in nature, and use sophisticated software and hardware such as bridges and routers to connect one or more audio/video networks routing various kinds of data while Hospitality/Convention Centre networks are the simplest and are usually operated by musicians. They deal with fewer connections compared to Broadcast networks that require experienced engineers to operate and deal with hundreds of connections at a time. Project studio networks are in between the two extremes in terms of connection complexity.

A number of non-mLAN Client/Server connection management applications have been developed for these networks using third-generation languages such as C, C++

and C#. These applications can be grouped into three groups, namely the Grid-based patchbays, the List-based patchbays, and the Graphic-based patchbays. Only one mLAN Client/Server-based patchbay, the NAS Explorer, has been developed. The NAS Explorer patchbay is easy to use and understand but cannot be used in all sound installation networks.

The next chapter investigates the alternative development environment for mLAN Client/Server patchbays.

CHAPTER 4

4 An Alternative Development Environment for mLAN Client/Server Patchbays

Connection management application developers for mLAN Client/Server patchbays need to achieve a balance between producing a high quality, highly interactive application, and the amount of time and effort required to develop the application. mLAN Client/Server patchbays are connection management applications that utilise the mLAN Client/Server architecture discussed in *section 2.2*. As discussed in *chapter 3*, current non-mLAN and mLAN Client/Server patchbays (e.g. the NAS Explorer patchbay discussed in *section 3.3*) are predominantly developed using third-generation languages such as C, C++ and C#, which require enormous amounts of effort and time to develop a usable Graphic User Interface. This is because third-generation languages and their development environments do not provide essential tools and techniques for quickly creating and manipulating the behaviour of the graphic elements that form GUI applications. Furthermore, third-generation languages use complex syntax and structures that make it difficult for GUI developers to develop highly interactive GUI applications in the shortest possible time.

This investigation evaluates the possibility of using an IDE that supports scripting technologies for developing mLAN Client/Server patchbays. Scripting technologies were initially created for purposes of manipulating and gluing together components for distributed systems that are developed using third-generation languages such as C and C++. At the time of this investigation, scripting technologies had matured to incorporate powerful development tools, OOP concepts and support the development of highly interactive GUI applications for both the web and desktop standalone applications.

Prechelt (2002) compared scripting languages and third-generation languages for the following characteristics:

- Run time.

- Memory consumption.
- Source text length.
- Comment density.
- Program structure.
- Reliability.
- Amount of effort required to write programs.

Prechelt (2002) concluded that scripting languages are more effective and productive than third-generation languages like C and C++. Prechelt (2002) also found that third-generation language programs are typically two to three times as long as scripting language programs and they tend to contain a significantly higher density of comments. Moreover, his investigation found that scripting language programs take only one third of the time required by programmers for the third-generation to design, write, and test the program. This is mainly because of many improvements in scripting technologies that have evolved to include many third-generation language concepts such as OOP, inheritance, use of classes, objects, encapsulation, constructors and powerful graphics development tools. Examples of scripting languages and development environments available for application development today include JavaScript, HyperText Preprocessor (PHP), Perl, ActionScript, Sun Microsystems JavaFX, Adobe Systems Flash Professional 8, and Adobe Flex 2.

4.1 Possible Alternative Client Development Environments

This section discusses possible mLAN Client/Server clients development environments and scripting languages that can be used by interactive application developers. The development environments discussed include:

- The Microsoft Silverlight.
- The Sun Microsystems JavaFX.
- The Adobe Systems Adobe Flash Professional 8.
- The Adobe Systems Adobe Flex 2.
- The Laszlo Systems OpenLaszlo.

4.1.1 Microsoft Silverlight

Microsoft Silverlight is a cross-browser, cross-platform for building and delivering next generation .NET based media experiences and rich interactive applications for the Web [Microsoft Corporation, 2007a]. It supports Asynchronous JavaScript and XML (AJAX), Visual Basic.Net (VB.Net), C#, Python, and Ruby [Microsoft Corporation, 2007a]. It also uses vector-based graphics, media, text, animation, and overlays that enable seamless integration of graphics and effects into any existing Web application. Since Microsoft Silverlight is based on the Microsoft .NET Framework, it gives developers the opportunity to use existing skills and tools. Microsoft Silverlight presentation capability is based on the Microsoft .NET Framework 3.0 (the Windows programming infrastructure) and the eXtensible Application Markup Language (XAML) [Microsoft Corporation, 2007b]. *Figure 4.1* shows the Microsoft Silverlight authoring environment where developers create and modify media components during application development.

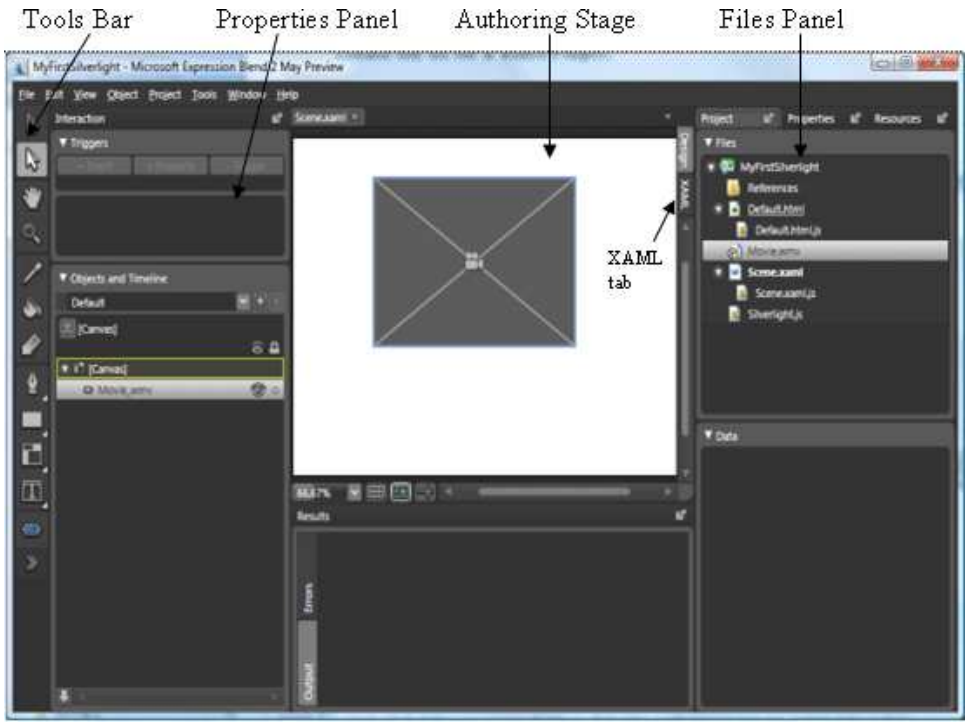

Figure 4.1: Microsoft Silverlight Application Authoring Environment [Microsoft Corporation, 2007b]

The *XAML* tab is used to access the XAML editor that allows the developer to enter XAML commands to control the application graphics, and add interactivity to the application under development. Microsoft Silverlight applications run on both Mac and Windows Operating Systems (OS). Its architecture comprises of three components; the presentation framework, the .Net Framework and the installer and updater modules [*Figure 4.2*].

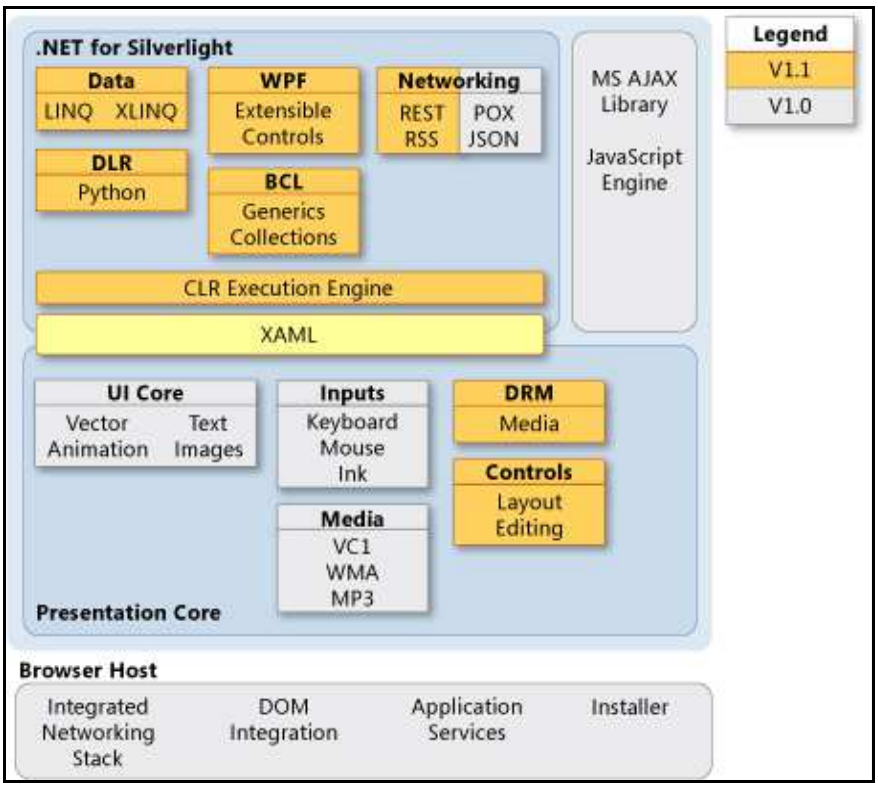

Figure 4.2: Microsoft Silverlight Architecture
[Microsoft Corporation, 2007b]

4.1.2 Sun Microsystems JavaFX

Sun Microsystems JavaFX is a group of products developed by Sun Systems that are based on Java technologies which enable developers to create and deploy interactive Rich Internet Applications (RIA) of high quality easily and in a very short space of time. Sun JavaFX applications can run on the desktop, on most popular web browsers, or on mobile devices that support Sun Microsystems JavaFX technologies. At the time of this investigation, the Sun Microsystems JavaFX releases included the JavaFX Script and the JavaFX Mobile [Sun Microsystems, 2007]:

- JavaFX Script[2] - Is a scripting language that is declarative and statically typed allowing developers to create rich, interactive Graphic User Interface applications with ease using the Java2D swing GUI components.
- JavaFX Mobile – Is a software system that provides a unified runtime environment, which allows for flexible development of interactive RIA for wireless carriers and mobile devices.

4.1.3 Adobe Systems Adobe Flash Professional 8

Adobe Flash Professional 8 (also known as Adobe Flash) is a powerful multimedia authoring and playback system from Adobe Systems that is bandwidth friendly because it uses small files and is browser independent [Adobe Macromedia, 2005]. Adobe Flash Professional 8 uses vector-graphic animation technology that allows developers to develop quality interactive RIA for a wide range of platforms. It uses ActionScript as its scripting language for application development [*section 4.2*].

4.1.4 Adobe Systems Adobe Flex 2

Adobe Flex[3] (Flex), like Adobe Flash Professional 8 is an ActionScript-based development system for developing Flash-based applications that was developed by Adobe Systems. It was initially introduced as a Java 2 Platform, Enterprise Edition (J2EE) application. Adobe Flex compiles ActionScript code and XML-based user interface descriptions (MXML) into binary Flash files (SWF files) like Adobe Flash Professional [Adobe Systems, 2007]. Armed with a wide range of user interface functions and tools, Flex is a powerful tool for creating rich client applications. It provides a modern, standards-based language and programming model that supports common design patterns and includes an Eclipse-based development environment, advanced data services, and a fast, enterprise-class client runtime based on the Adobe Flash Player software [Adobe Systems, 2007]. While Adobe Flash provides a powerful authoring tool for web developers, Adobe Flex enables more application developers to leverage the powerful Flash runtime to create data-driven RIA. Adobe Flash and Adobe Flex seem to be directed at doing the same thing but in actual fact,

[2] JavaFX Script Programming Language Reference - https://openjfx.dev.java.net/JavaFX_Programming_Language.html
[3] Adobe Flex 2 Language Reference - http://livedocs.adobe.com/flex/2/langref

they complement each other to produce high performance cross-platform applications for desktops and the web.

4.1.5 Laszlo Systems OpenLaszlo

OpenLaszlo[4] is an open-source platform for developing rich internet applications for the web. OpenLaszlo applications run on any popular browsers and platforms (Windows, Mac, Linux, Internet Explorer (IE), and Firefox). Its programs are written in XML and JavaScript, and then compiled into Flash. It also supports a rich graphics model with scalable vectors, bitmaps, movies, animation, transparency, fonts, audio, streaming media, reusable components, user interface widgets, control panels, property sheets, keyboard navigation, browser "back button" navigation, and graphical editing tools [Laszlo Systems, 2007]. *Figure 4.3* shows a typical OpenLaszlo application called Pandora, which is a music discovery service that uses OpenLaszlo to implement a slick, easy to use interface for listening to personalized internet radio stations via streaming MP3 audio.

Figure 4.3: Pandora Music Discovery Service
[Laszlo Systems, 2007]

[4] OpenLaszlo documentation - http://www.openlaszlo.org/documentation

4.2 The Alternative Development Environment and Scripting Language for mLAN Client/Server Clients

Of the 5 development environments discussed in the preceding section, Adobe Flash Professional 8 using ActionScript 2.0 was chosen as the development environment for mLAN Client/Server clients. Adobe Flash Professional 8 was chosen because at the time of the inception of this investigation there was Flash expertise in the Rhodes Computer Science department and the software was already available locally. No initial costs were incurred to use Adobe Flash and the developer had access to experienced Adobe Flash users, which accelerated his learning of the Adobe Flash toolkit. Adobe Systems uses the term *Adobe Flash*, or simply *Flash*, to refer to both the Adobe Flash Player and to the Adobe Flash Professional 8 multimedia authoring program [Adobe Macromedia, 2005]. For this investigation, Adobe Flash Professional version 8 was utilized.

4.2.1 Adobe Flash Professional 8 Description

Adobe Flash Professional 8 is an authoring tool that designers and developers use to create presentations, applications, and other content that enables user interaction [Adobe Macromedia, 2005]. It was originally designed to create animations for display on web pages in the 1980 s.This is because Adobe Flash files are smaller in size as they use vector graphics, which require significantly less memory and storage space than bitmap graphics[5]. At the time of this investigation, Adobe Flash Professional 8 provided a powerful interactive platform, with an object-oriented, type safe dynamic scripting engine that supports ActionScript 2.0, graphics effects filters**,** blend modes, bitmap rendering and advanced video and audio playback features [Adobe Macromedia, 2005]. Coupled with these features is its flexible design and animation authoring IDE that is composed of many programming interfaces, which include the code editor, the animation editor, the vector art editor, a compiler, a debugger and a very good help system, which were all essential in the development of mLAN Client/Server clients.

[5] Vector graphics are smaller because they are represented by mathematical formulas while bitmap graphics are larger because each individual pixel in the image requires a separate piece of data to represent it [Macromedia, 2005].

Blend modes allow the user to create composite images, in a process called compositing. Compositing is the process of varying the transparency or colour interaction of two or more overlapping objects. Blending allows the user to create unique effects by blending the colours in overlapping MovieClips[6] [Adobe Macromedia, 2005]. Elements that make-up a blend mode include [Adobe Macromedia, 2005]:

- Blend colour - Is the colour applied to the blend mode.
- Opacity - Is the degree of transparency applied to the blend mode.
- Base colour - Is the colour of pixels underneath the blend colour.
- Result colour - Is the result of the blend's effect on the base colour.

The *blend modes* feature was utilised for representing the changing states of the connection status for mLAN plugs on the Broadcast network patchbay (grid-matrix) when the user either performed a connection or disconnection.

Another feature of the Adobe Flash Professional 8 IDE that was useful in developing mLAN Client/Server clients is the *graphics effects filter*. Filters allow the user to add interesting visual effects to text, buttons, and MovieClips and are most often associated with applying drop shadows, blurs, glows, and bevels to graphic elements [Adobe Macromedia, 2005]. This feature was integral in developing the Hospitality/Convention Centre patchbay that required the use of shadows on the MovieClip objects, which represented individual rooms in a hospitality building. This feature will be explored further in *chapter 7* that discusses the development of a patchbay for Hospitality/Convention Centre networks.

Figure 4.4 shows the important components of the Adobe Flash Professional 8 design and animation authoring IDE that were utilised for creating patchbay graphics, animations and writing the scripts that control these graphics.

---

[6] A MovieClip symbol in Adobe Flash is a reusable piece of flash computer graphic - usually consisting of one or more graphic/button symbols. MovieClip behaviours are controllable easily using ActionScript. MovieClip properties that can be controlled using ActionScript include; MovieClip dimensions, position, colour, and alpha value to mention but a few. MovieClips can be deleted, and duplicated using the copy-paste functionality. MovieClips form the basic unit for an Adobe Flash application.

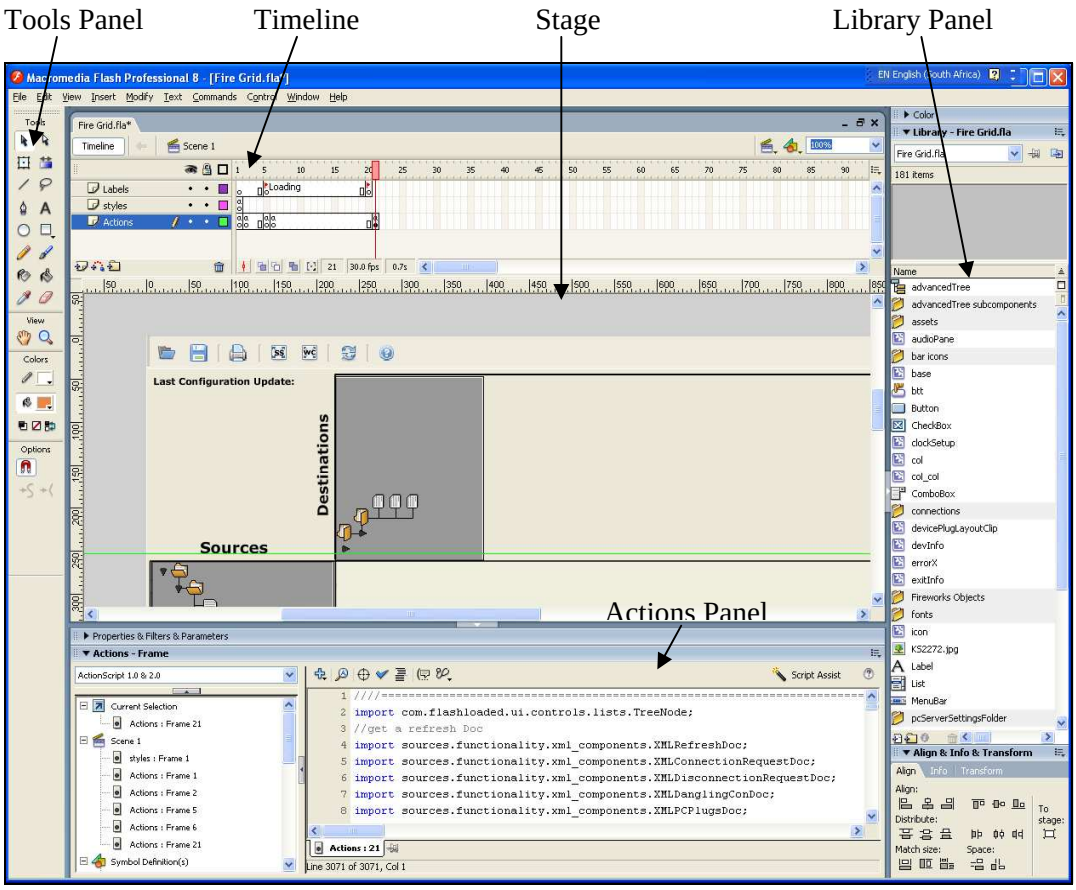

Figure 4.4: Adobe Flash Professional 8 Design and Animation Authoring IDE

When developers are creating Adobe Flash content using the Adobe Flash Professional 8 IDE they work in a Flash document file. These Flash documents have the file extension .fla (FLA). The components that form a Flash document include [Adobe Macromedia, 2005]:

- The *Stage* which is the rectangular area where the developer places graphic content, including vector art, text boxes, buttons, imported bitmap graphics or video clips, when creating Flash documents. The *Stage* has many features that aid graphics creation such as the *zoom-in* and *out* capabilities to change the view of the *Stage* as you work, the grid, the guides, and the rulers, which help the developer position content precisely on the *Stage* and the *Hand tool* that lets one move the *Stage* for example if because of *zooming- out*, one cannot see some parts of the Stage. The *Stage* area is also the rectangular space in Adobe Macromedia Flash Player or in a web browser window that appears when the Flash application is being played back.

- The *Timeline* organizes and controls a document's content over time in Layers and Frames. Flash documents divide lengths of time into frames. Layers are like multiple strips stacked on top of one another, each containing a different image that appears on the *Stage*. The major components of the *Timeline* are layers, frames, and the playhead. *Figure 4.5* shows the Adobe Flash Professional 8 IDE Timeline components.

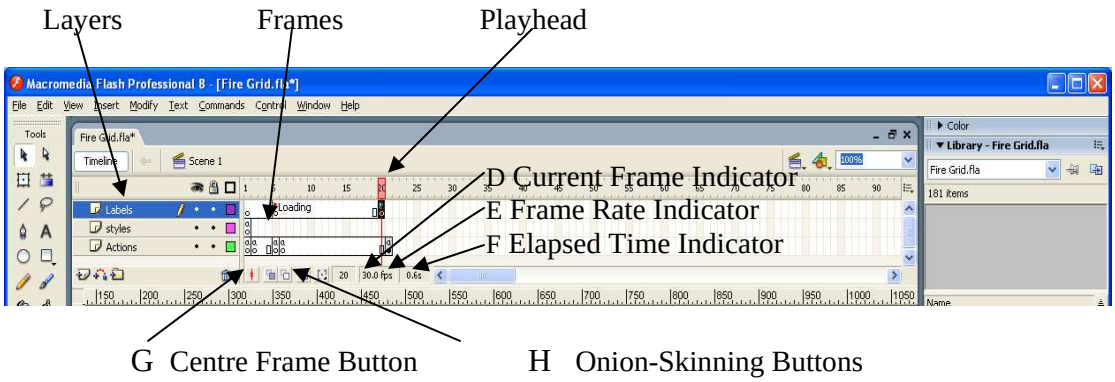

Figure 4.5: Adobe Flash Professional 8 IDE Timeline

Figure 4.5 shows three layers, the "Labels" layer, the "Styles" layer and the "Actions" layer. The "Labels" layer contains all the labels that the user sees when the application is loading like "Please wait, the application is loading components" in three patchbays. The "Styles" layer contains the styles that the applications use. These include all the skins that were used for Flash build-in components, the colours and the Text Font styles. The "Actions" layer contains the ActionScript 2.0 code that controls the behaviour of the graphics and MovieClips on the *Stage*. At playback time the "Playhead" moves through the timeline as a document plays to display the current frame displayed on the *Stage* and subsequently the *Stage* contents on that particular frame. The current frame number is indicated by the "Current Frame Indicator" while the rate at which frames are being viewed is indicated by the "Frame Rate Indicator", and is measured in frames per second (fps). The default frame rate for Adobe Flash applications is 12 fps. The "Elapsed Time Indicator" displays the time in seconds taken to reach a particular frame in the Timeline from the beginning of the playback at frame 1.

- The *Library panel* is where Adobe Flash Professional 8 stores and displays a list of the media elements in the Flash document. These are usually MovieClips, bitmap graphics, Fireworks artwork and any Flash built-in components that the developer adds to the Flash document.
- *ActionScript code* shown in the Actions panel in *Figure 4.4* allows the developer to add interactivity to the media elements in the Flash document's *Library panel*. This investigation utilised ActionScript 2.0. *ActionScript 2.0* is a scripting language based on ECMAScript scripting programming language that was standardized by ECMA International[7] and specified in the ECMA-262 specification[8].

These 4 components are fundamental in the creation of Adobe Flash applications components. The following section discusses how these components are utilised for creating a MovieClip Flash graphic symbol, which forms the basis for Flash applications.

- The first step is to open a new Flash Document [*Figure 4.6*]. The Flash Document exposes many drawing and programming interfaces that are used for creating the symbol. Some important interfaces that are shown in *Figure 4.6* include the *Tools* panel, the *Colour Mixer* Panel, the *Library Panel*, the *Alignment* Panel and the *Actions* Panel. The initial graphic is created using the *Rectangle* Tool that the user clicks and draws the shape graphic required on the *Stage* [*Figure 4.6*]. The *Stroke Colour* Tool is used to specify the border colour of the graphic to be drawn whilst the *Fill Colour* Tool is used to specify the fill colour.

[7] European Computer Manufacturers Association (ECMA) website - http://www.ecma-international.org

[8] Standard ECMA-262 (3r d Edition - December 1999) - http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-262.pdf

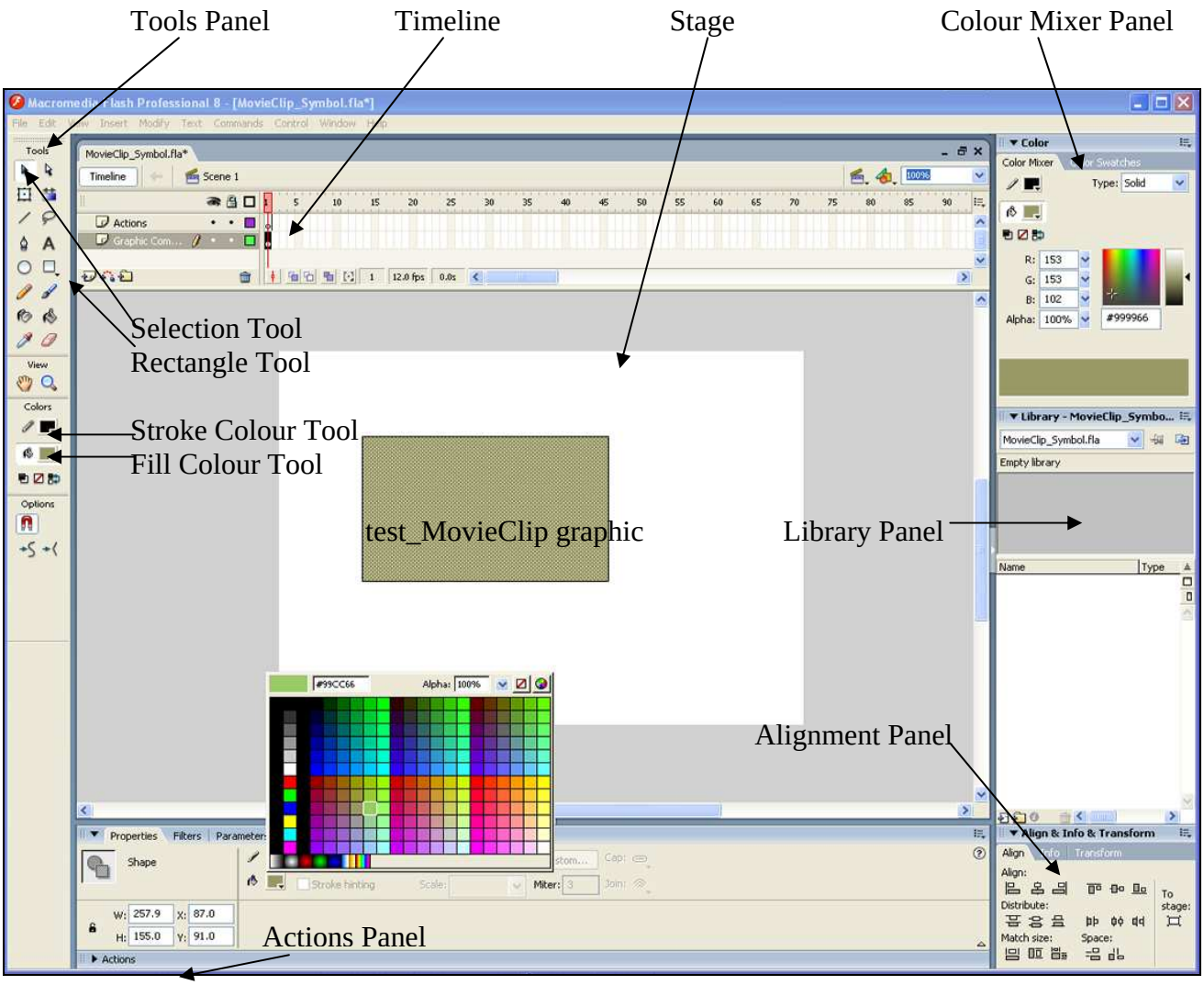

Figure 4.6: Creating a MovieClip Symbol – Flash Document

- When the basic graphic has been drawn on the *Stage*, it is selected using the *Selection* Tool [*Figure 4.6*].
- With the graphic selected, right-click it and choose the "Convert to Symbol…" menu item to convert the graphic into a symbol [*Figure 4.7*].

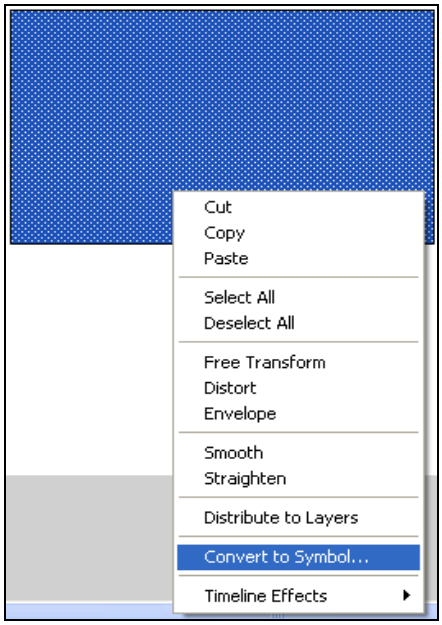

Figure 4.7: Converting a graphic drawing into a MovieClip Symbol

- A *Convert To Symbol* panel appears on which attributes of the new symbol are specified. Attributes that can be specified include [*Figure 4.8*]:

    a) The symbol *name* – Specifies the name of the MovieClip.
    b) The symbol *type* – Specifies the type of the graphic to be created (MovieClip, Button or Graphic). Note that in *Figure 4.8*, the M*ovie Clip* radio button is selected because a MovieClip symbol is being created.
    c) The symbol *registration* – Specifies the registration point of the MovieClip.
    d) The symbol *linkage identifier* – Specifies the ID name that will be used to reference the MovieClip in the *Library* Panel.
    e) The symbol *class* – Specifies the name of the class to be assigned to the MovieClip.

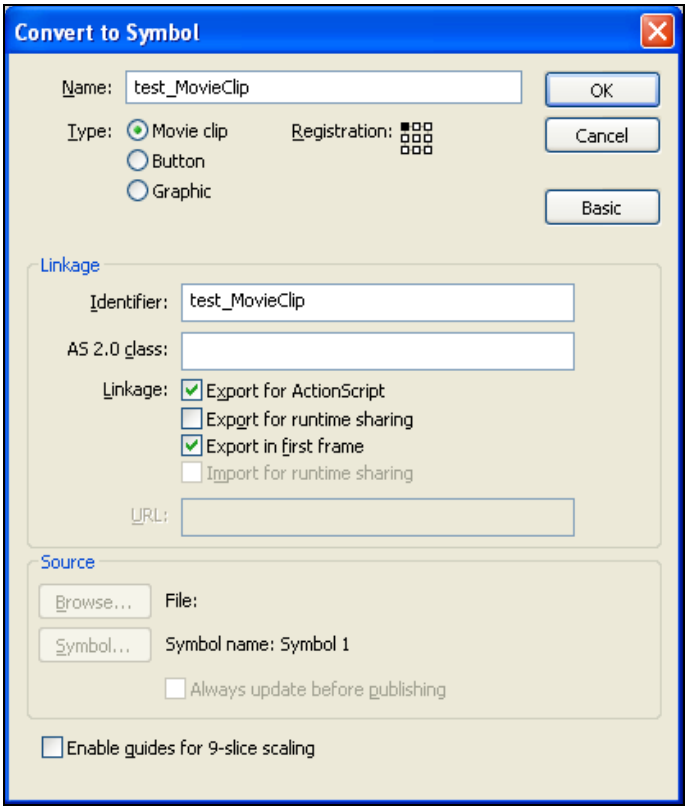

Figure 4.8: Specifying MovieClip Properties

- After specifying the new clip attributes click the *OK* button. A MovieClip with the specified name will appear in the *Library* Panel [*Figure 4.9*].

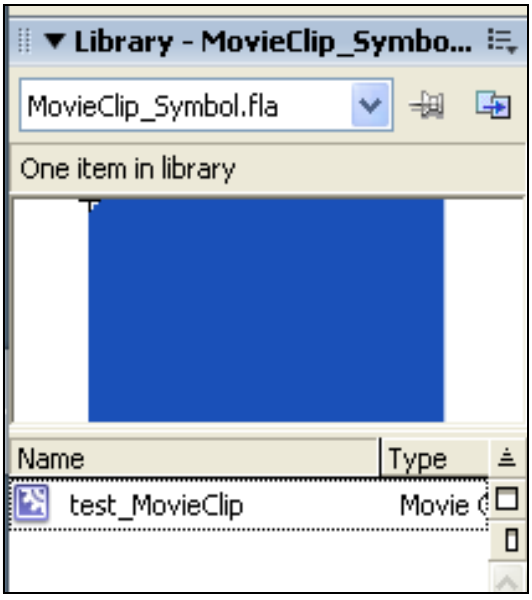

Figure 4.9: MovieClip in the Library Panel

Symbols in Adobe Flash can be drawn completely in ActionScript. The MovieClip symbol behaviour can be controlled easily using ActionScript, which can be in the

form of a class assigned to the clip in the *Convert To Symbol* panel or just a script written in the *Actions* Panel. The MovieClip attributes that can be controlled using ActionScript include its (this section lists only those properties that were utilised in developing the three patchbays):

- Dimensions – The developer can specify the height and width of a MovieClip symbol using the *_height* and *_width* variables.
- Position – The developer can specify the X and Y coordinates position of the MovieClip symbol using the _x and _y variables.
- _alpha value – This attribute allows the developer to set the alpha transparency value of a MovieClip symbol. Valid values are 0 (fully transparent) to 100 (fully opaque).
- Enabled – The attribute is a Boolean variable that indicates whether a MovieClip symbol is enabled. The default value of enabled is *true* but can be set to false using "[MovieClip name].enabled = false".
- Name – Specifies the name of the MovieClip symbol.
- Rotation – Specifies the rotation of a MovieClip symbol, in degrees, from its original orientation. Values from 0 to 180 degrees represent clockwise rotation; values from 0 to -180 degrees represent counter clockwise rotation.
- Visibility – A Boolean variable that indicates whether a MovieClip symbol is visible. MovieClip symbols that are not visible ([MovieClip name]._visible = false) are disabled.
- Mouse position – The developer can get the X and Y coordinates position of a mouse pointer over a MovieClip symbol using the _xmouse and _ymouse variables.

Each MovieClip symbol supports a wide range of Event Handlers, some of which include the:

- onPress () Event Handler – Invoked when the user clicks the mouse while the pointer is over a MovieClip symbol.
- onRelease () Event Handler – Invoked when a user releases the mouse button over a MovieClip symbol.

- onRollOver () Event Handler – Invoked when the user moves the mouse pointer over a MovieClip symbol area.
- onRollOut () Event Handler – Invoked when a user moves the pointer outside a MovieClip symbol area.

Other actions that can be performed on a MovieClip symbol include:

- Duplicating it using the "duplicateMovieClip(name:String, depth:Number, [initObject:Object])" method that returns a MovieClip object.
- Deleting it using the "removeMovieClip ()" or the "unloadMovie ()" methods.

Figure 4.10 shows the *Actions* Panel that depicts ActionScript Event Handler code that uses MovieClip properties and methods outlined in the preceding section.

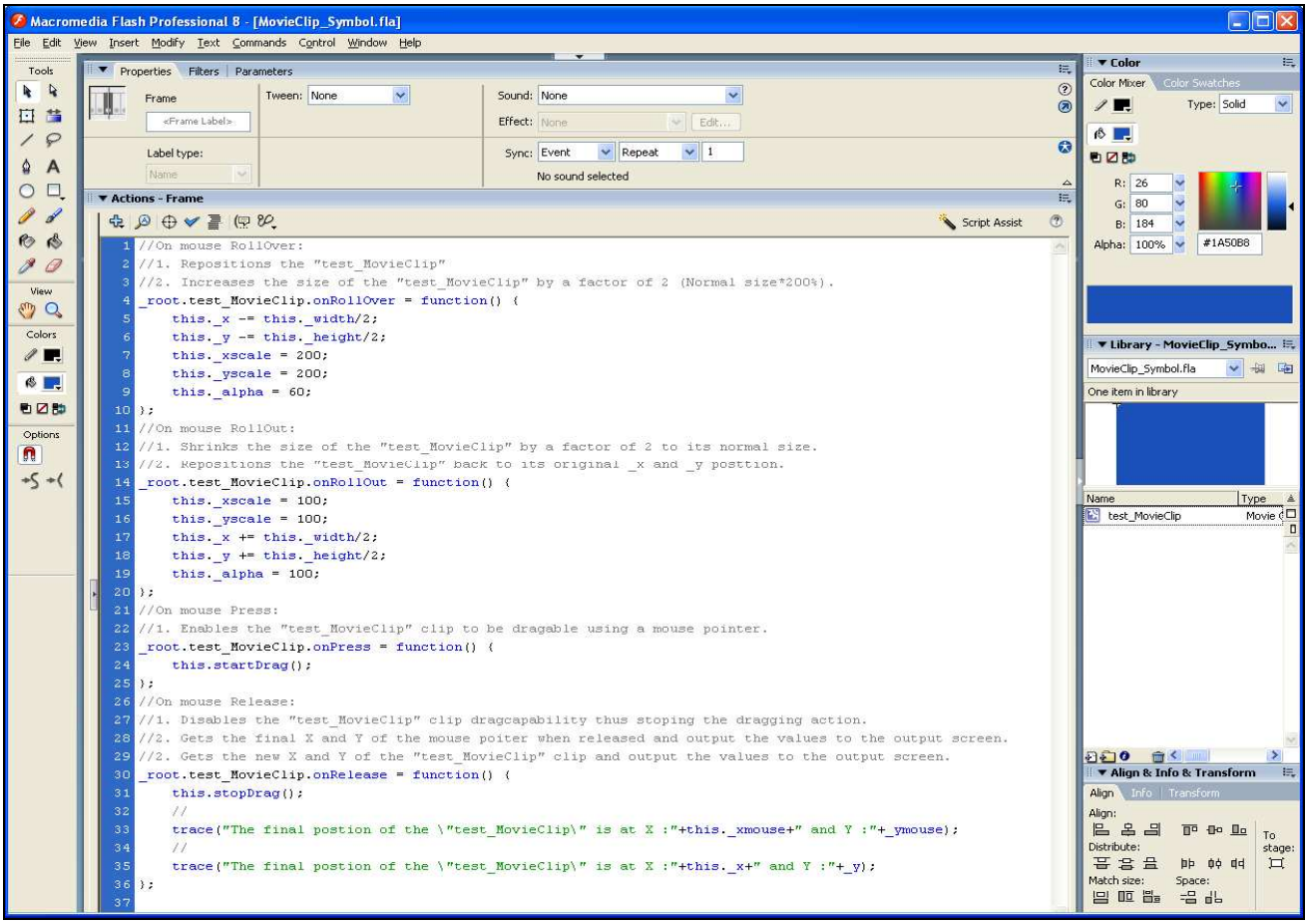

Figure 4.10: Specifying MovieClip Properties

Four MovieClip Event Handlers are shown in the *Actions* Panel displayed in *Figure 4.10*:

a) The onRollOver Event Handler

_root.test_MovieClip.**onRollOver** = function ( ) {

1. Repositions the "test_MovieClip" clip.
2. Increases the size of the "test_MovieClip" clip by a factor of 2 (Normal size*200%).

};

b) The onRollOut Event Handler

_root.test_MovieClip.**onRollOut** = function ( ) {

1. Shrinks the size of the "test_MovieClip" clip by a factor of 2 to its normal size.
2. Repositions the "test_MovieClip" back to its original _x and _y position.

};

c) The onPress Event Handler

_root.test_MovieClip.**onPress** = function ( ) {

1. Enables the "test_MovieClip" clip to be drag-able using a mouse pointer.

};

d) The onRelease Event Handler

_root.test_MovieClip.**onRelease** = function ( ) {

1. Disables the "test_MovieClip" clip drag capability thus stopping the drag action.
2. Gets the final X and Y coordinates of the mouse pointer when released over a MovieClip and outputs the values to the Adobe Flash output screen.
3. Gets the new X and Y coordinates of the "test_MovieClip" clip and outputs the values to the Adobe Flash output screen.

};

Once the developer creates the Adobe Flash application (or MovieClip symbol) on the Flash document, it is published by clicking the “File” menu on the Adobe Flash Professional 8 IDE and selecting “Publish”. A compressed version of the Flash document with all the media is created with the extension .swf (SWF). “SWF” stands

for “Small Web File”, and is a proprietary vector graphics file format produced by the Adobe Flash software. The Adobe Flash Player is used to play the SWF file in a web browser or as a standalone application. This investigation utilised the latest Flash player, Adobe Flash Player 9. According to Adobe Macromedia, the Adobe Flash Player is a high-performance, lightweight, highly expressive client runtime that delivers powerful and consistent user experiences across major operating systems, browsers, mobile phones, and devices [Adobe Macromedia, 2005]. Adobe Flash Player 9 consists of five important components that ensure effective rendering of flash applications [Adobe Macromedia, 2005]:

- A *Virtual Machine (AVM)* known as AVM2 that interprets and executes byte code. ActionScript code is compiled to this byte code. AVM2 supports full runtime error reporting, built-in debugging, and binary socket support so developers can extend the player to work with any binary protocol. Adobe Flash Player 9 also contains AVM1, which executes legacy ActionScript for maintaining backward compatibility with existing content.
- A hierarchical frame-advancing *Visual Object Model*. Adobe Flash maintains a hierarchy of MovieClip objects and graphics in a layered display list, where each object has its own frame-subdivided timeline, a depth number, and layers map onto a z-order.
- A set of *media decoders* that can decode and playback multiple streams of compressed audio and video simultaneously. It has audio decoders for Adaptive Differential Pulse Code Modulation (ADPCM), MP3, and NellyMoser audio streams. It also supports video codecs, including the Sorenson H.263 and Sorenson Spark codecs.
- A suit of advanced *rendering algorithms* for rendering vector graphics that include lines, gradients and filtered bitmap fills. Adobe Flash is a vector engine, but its support of bitmap fills means it is also a bitmap engine. When a bitmap is dragged onto the *Stage*, Adobe Flash actually creates a four-sided shape then attaches a bitmap fill of the graphic. Each shape fill can have a texture transform as well as simple shader capabilities for playing with the colour and alpha.

- A *framework library* that has basic string functions, arrays, sorting and mathematical functions. Adobe Flash also adds support for TCP communications, asynchronous loading, and a very advanced and easy-to-use XML interface that enables easy use of the communication protocol by the mLAN client and the mCMS server.

4.3 Chapter Summary

This chapter discussed an alternative development IDE and the scripting language chosen for developing mLAN Client/Server clients. Following from the survey of current patchbay applications in *chapter 3*, it was shown that current mLAN and non-mLAN patchbay applications are developed in C, C++ and C#. However, Prechelt in his comparison of scripting languages and third-generation languages using the criteria of run time, memory consumption, source text length, comment density, program structure, reliability and amount of effort required to write programs found that scripting languages are more effective and productive than third-generation languages like C and C++. This is because these languages use complex syntax and structures which require far more time and effort to write a quality GUI application than scripting languages. For instance, he proved that third-generation language programs are typically two to three times as long as scripting language programs and they tend to contain a significantly higher density of comments. As a result, this chapter evaluated an alternative development environment that supports scripting technologies. Five possible alternative mLAN Client/Server client development environments were considered that is Microsoft Silverlight, Sun Microsystems JavaFX, Macromedia Adobe Flash Professional 8, Adobe Flex 2, and Laszlo Systems OpenLaszlo. Of the five alternative development environments discussed, Macromedia Adobe Flash Professional 8 using ActionScript 2.0 was chosen as the development environment because it was already a known environment at the Rhodes Computer Science Department.

The next chapter explains in detail the development process of the Broadcast network patchbay.

CHAPTER 5

5 Broadcast Patchbay Design and Development

Broadcast hardware studio networks are more complex than Project studio and Hospitality/Convention Centre networks. Broadcast studio requirements, and the nature of the patchbays for these networks are well defined. Therefore, it was decided that the Broadcast patchbay be developed before the Project studio and Hospitality/Convention Centre patchbays, as it had the potential of testing Adobe Flash capabilities as an alternative for developing mLAN connection management applications. The assumption was that if Adobe Flash allowed the effective implementation of a complex patchbay such as the Broadcast patchbay, it would be easily utilised to develop simpler patchbays for the other networks. Grid-based patchbays are commonly deployed within Broadcast networks because they allow for a realistic viewing of the input and output plugs of all devices on the audio network, using two device trees and a connections grid-matrix [*Chapter 3: section 3.2.1*]. The grid-matrix displays the state of the connections (connected or disconnected) for each pair of plugs displayed on the visible tree nodes. The tree-like structure coupled with the grid-matrix allows the display of as many devices, which is a requirement for Broadcast networks that deal with hundreds of connections at a time. In contrast to non-mLAN grid-based patchbays discussed in *section 3.2.1*, the patchbay developed for Broadcast networks in this investigation utilised the mLAN Client/Server architecture discussed in *section 2.3*.

Figure 5.1 displays eight phases of the Iterative and Incremental Process (Rational Unified Process - RUP) that was followed in the development of the Broadcast patchbay. RUP is a software engineering process that provides for the framework of best software development practices. Following this process reduces risks involved in the development of software, and ensures that the delivered software successfully fulfils end user requirements within a predictable schedule and budget [Rational SDC, 1998, Kruchten, 2000]. This chapter discusses in detail how this process (RUP) was adopted in the planning, designing and implementation of a grid-based patchbay for Broadcast networks.

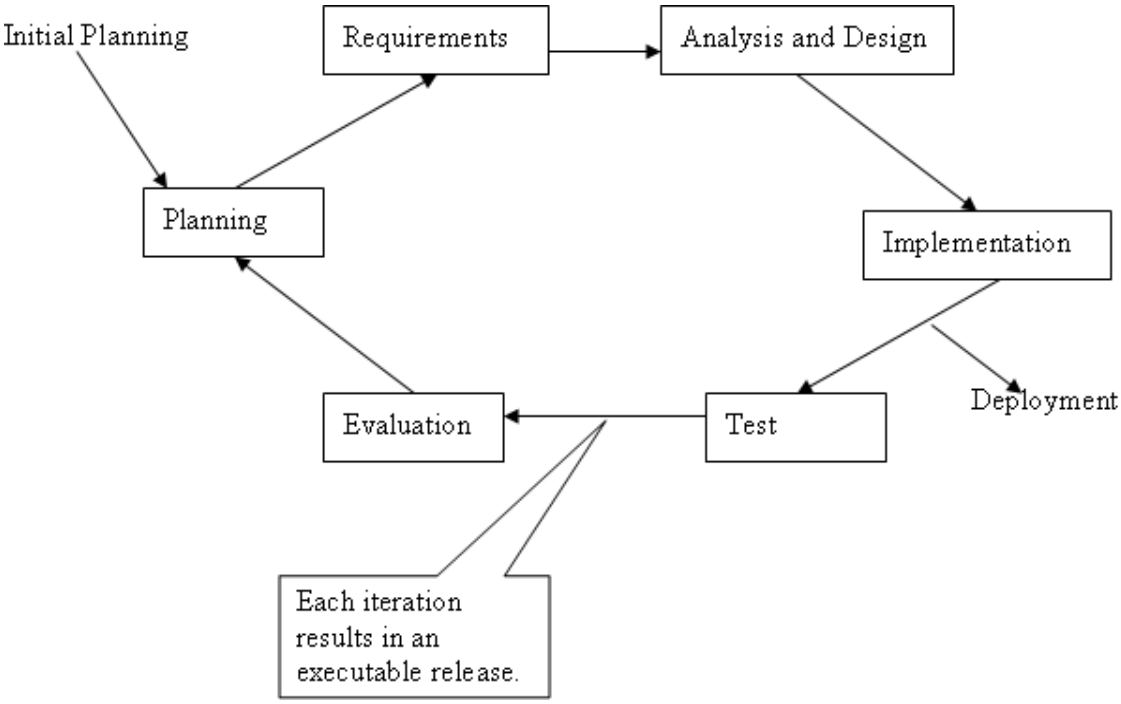

Figure 5.1: An Iterative and Incremental Process (RUP) [Kruchten, 2000]

5.1 Broadcast Patchbay Requirements Analysis

As already mentioned, many grid-based patchbays have been developed to control audio routing over Broadcast networks. As a result, software requirements for the Broadcast patchbay were acquired from already existing grid-based patchbay manuals such as the Otari ND-20 mLAN Control Software manual [OTARI, 2005]. Broadcast patchbay requirements were listed from analysing the functionality provided by the Otari ND-20 mLAN Control Software. Also added were mLAN network specific requirements such as the ability to view only audio plugs, or only MIDI plugs, or both together, and changing device Plug Layouts. Textual scenarios (Stimulus/Response sequences) were used to describe each feature incorporated into the Broadcast patchbay. The IEEE Recommended Practice for Software Requirements Specifications [IEEE Inc, 1998] was utilised as the standard guideline for documenting these requirements. *Appendix A-A1* provides a complete Software Requirements Specification document for the Broadcast patchbay.

5.2 Broadcast Patchbay Description

The Broadcast patchbay developed was named the “FireGrid”, which reflects that it was developed for Firewire networks and would use a grid-matrix for connection management. The Broadcast patchbay comprises three main panels, namely the *Control Window*, the *Wordclock* panel, and the *Settings* panel.

The *Control Window* [*Figure 5.2*] is the main control panel that appears when the Broadcast patchbay is started. It is on this panel that important audio routing and connection management tasks are performed. *Figure 5.2* displays a screenshot of the Broadcast patchbay *Control Window* that shows a typical mLAN network with three mLAN compatible devices, namely the *mLAN Windows PC* device, the *IOne Connects* device, and the *MAP4* device, together with their associated plugs. The *Control Window* has two main sections; the “device trees” section (that comprise of the *Sources tree* and the *Destinations tree*) and a “grid-matrix” section (that is named the *Control Grid-Matrix* in *Figure 5.2*). The *Sources tree* exposes the output device nodes and their plugs, whilst the *Destinations tree* displays the input device nodes and their plugs.

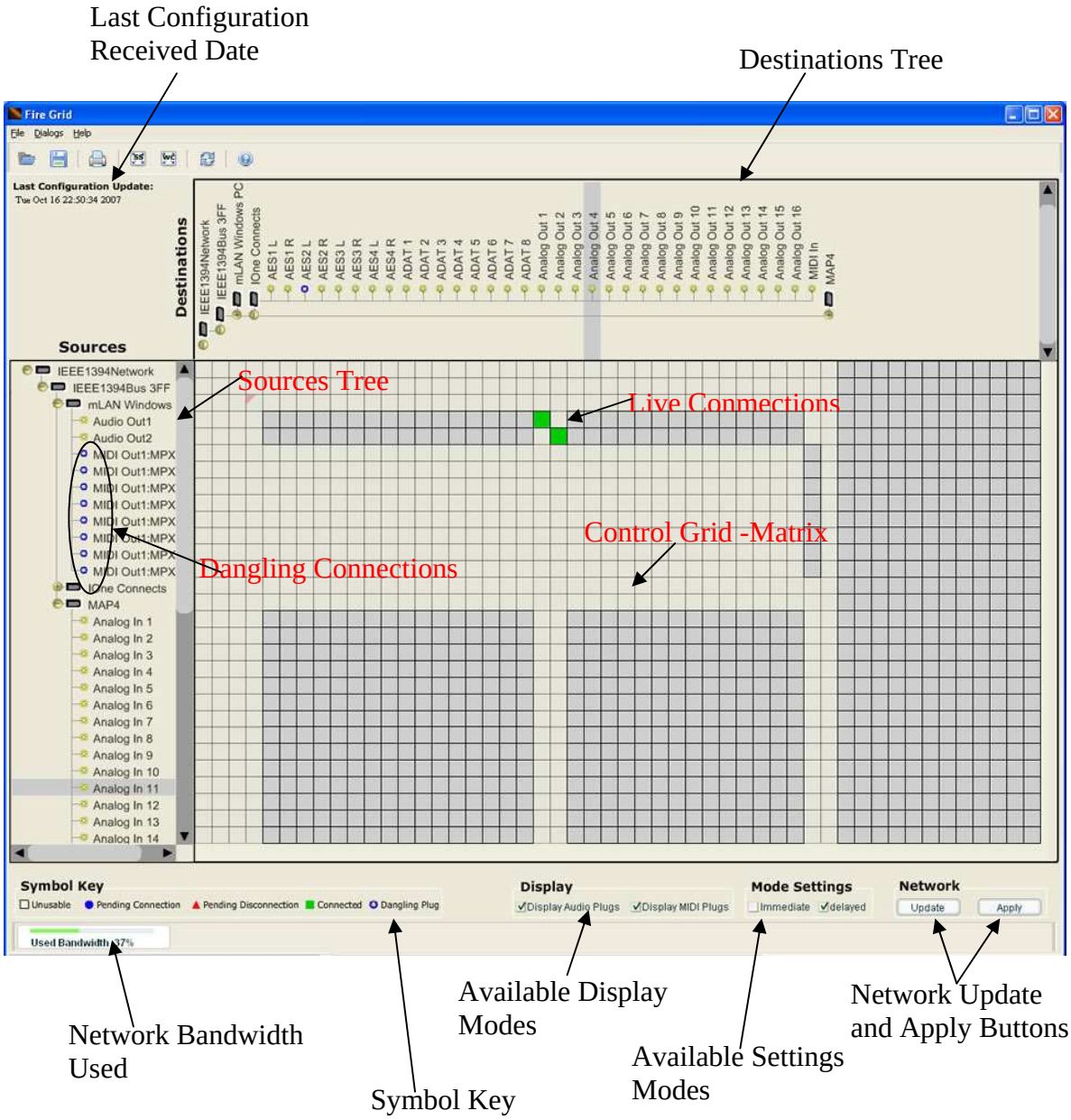

Figure 5.2: Broadcast Patchbay Control Window

The *Control Grid-Matrix* displays the connection status of two corresponding plugs (output and input plugs) visible on the *Sources tree* and *Destinations tree*. In *Figure 5.2*, the *Control Grid-Matrix* shows two live connections between the *mLAN Windows PC* and the *IOne Connects* devices. The source plugs, *Audio Out 1* and *2* of the *mLAN Windows PC* device on the *Sources tree*, are connected to destination plugs, *Analog Out 1* and *2* of the *IOne Connects* device on the *Destinations tree*. These live connections are represented by green square graphic boxes on the *Control Grid-Matrix*. Also shown are dangling connections, which are represented by blue round icons on the actual input and output plug nodes on the "device trees". Examples of

dangling connections displayed include the dangling connection on the destination plug *AES2L* of the *IOne Connects* device on the *Destinations tree*, and on the source plug *MIDI OUT1: MPX1* of the *mLAN Windows PC* device on the *Sources tree*. Connection management tasks that can be performed on the Broadcast patchbay *Control Window* include:

- Establishing audio connections.
- Breaking audio connections.
- Renaming mLAN network devices.
- Clearing all device connections for a particular mLAN network device.
- Viewing detailed device information for a particular mLAN network device.
- Requesting the display of only MIDI plugs, only Audio plugs or both together.
- Saving / Opening routing settings into / from a text file.
- Clearing dangling connections.
- Updating the patchbay with latest mLAN network information from the mCMS server.
- Identifying a particular mLAN network device.
- Changing Plug Layout type for a particular mLAN network device.
- Changing the patchbay Mode Settings (either Immediate or Delayed).

The *Wordclock* panel [*Figure 5.3*] allows the user to set/clear word clock Master/Slave configurations. It displays the mLAN network devices in two sections, namely the “Master Capable Devices” section, and the “Slave Capable Devices” section. The “Master Capable Devices” section displays a tree of device nodes that can be configured as a master device on the mLAN network, and the “Slave Capable Devices” section displays a tree of device nodes that can be configured as slaves on mLAN network. *Figure 5.3* displays a screenshot of the *Wordclock* panel that displays three mLAN devices on the network, namely the *mLAN Windows PC* device, the *IOne Connects* device, and the *MAP4* device. The screenshot shows the configured global master, the *IOne Connects* device, with two slaves, the *mLAN Windows PC* device and the *MAP4* device, below its tree node in the “Master Capable Devices” section. A global master device is a master device for all mLAN network

devices. Only one global master device can exist in an mLAN network at a time. The *IOne Connects* device is transmitting and receiving at a sample rate of 48 kHz, and therefore, all its slave devices are synchronised to transmit and receive at this sample rate.

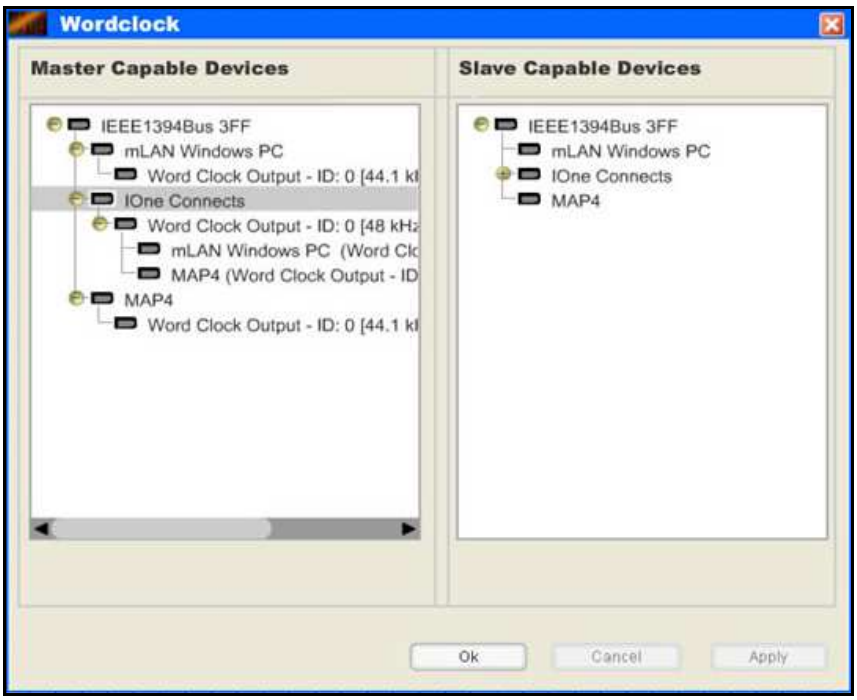

Figure 5.3: Broadcast Patchbay Wordclock Settings Panel

Connection management tasks that can be performed on the *Wordclock* panel include:

- Setting a global master unit for the whole mLAN network.
- Enslaving a single device to a particular master device.
- Releasing all slave devices of a particular master device.
- Removing a single slave device from its master device.
- Changing a master device word clock sample rate and its word clock source.

The *Settings* panel has two sections, namely the "Server Settings" section and the "PC Plugs" section. The "Server Settings" section [*Figure 5.4*] allows the user to configure the DNS name and the port number of the mCMS server to be utilised by the Broadcast patchbay. The current port number is "52941" and the DNS name is "Localhost". The mCMS server can reside on any workstation anywhere, as long as

the Broadcast patchbay can make a TCP/IP connection to it and pass XML messages using the specified port number.

Figure 5.4: Broadcast Patchbay Server Settings Panel – Server Settings

The "PC Plugs" section of the *Settings* panel [*Figure 5.5*] allows the user to specify the number of both MIDI and Audio source plugs for the *mLAN Windows PC* device. *Figure 5.5* shows that at this time, there are two audio source plugs and eight MIDI source plugs. The audio source plugs and MIDI source plugs combo boxes are used to configure the number of each plug type. At the time of this investigation, the mLAN firmware did not allow for the configuration of audio and MIDI destination plugs for the *mLAN Windows PC* device.

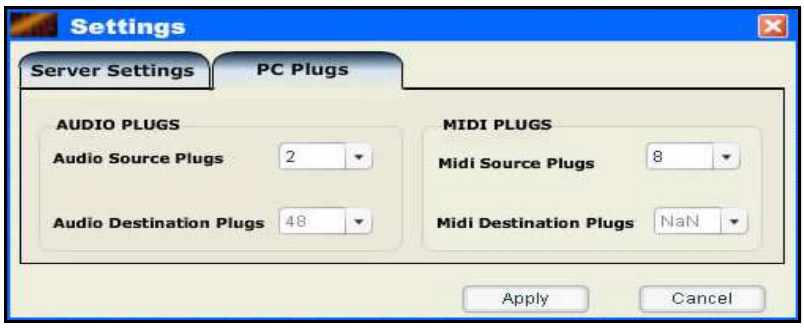

Figure 5.5: Broadcast Patchbay Server Settings panel – PC Plugs

5.3 Broadcast Patchbay Design and Implementation

5.3.1 Modelling Broadcast Software Requirements

The requirements elicitation phase discussed in *section 5.1* led to the creation of two central, high-level models, namely the Use Case Model and the Object Model, which further reinforced the understanding of the system requirements and the patchbay

users. The Unified Modelling Language (UML) was used for graphically specifying, visualising, constructing and clearly documenting the Broadcast patchbay requirements before the development process started. Graphically modelling the patchbay requirements has many benefits that include:

- Providing a structured approach to solving the problem.
- Providing an easy way of dealing with the complexity of software applications such as the Broadcast patchbay, by breaking its components into smaller manageable bits.
- Providing a way of managing and reducing the risk of mistakes once the development process starts [Maksimchuk and Naiburg, 2004].

The **Use Case Model** models the scope of the Broadcast patchbay. It is easily understood by both developers and application users. This provides a good way of communicating requirements between the developer and the user.

The **Object Model** (Class Diagram) describes a static view of the system in terms of classes and their relationships [Booch, Rumbaugh and Jacobson, 2005]. It paves the way for more concrete and complex models such as Activity diagrams and Sequence diagrams, which define the system before implementation starts. For purposes of this investigation, only sequence diagrams were utilised because they model the flow of logic within the system in a visual manner, enabling the documenting and validating of system requirements, and can be used for both analysis and design purposes.

5.3.1.1 Use-Case Driven Analysis

5.3.1.1.1 Use Case Diagram

Figure 5.6 displays the Broadcast patchbay Use Case diagram, which displays the high-level functions of the Broadcast patchbay and identifies two entities (Actors) that interact with the patchbay, namely the sound engineer and the mCMS Server.

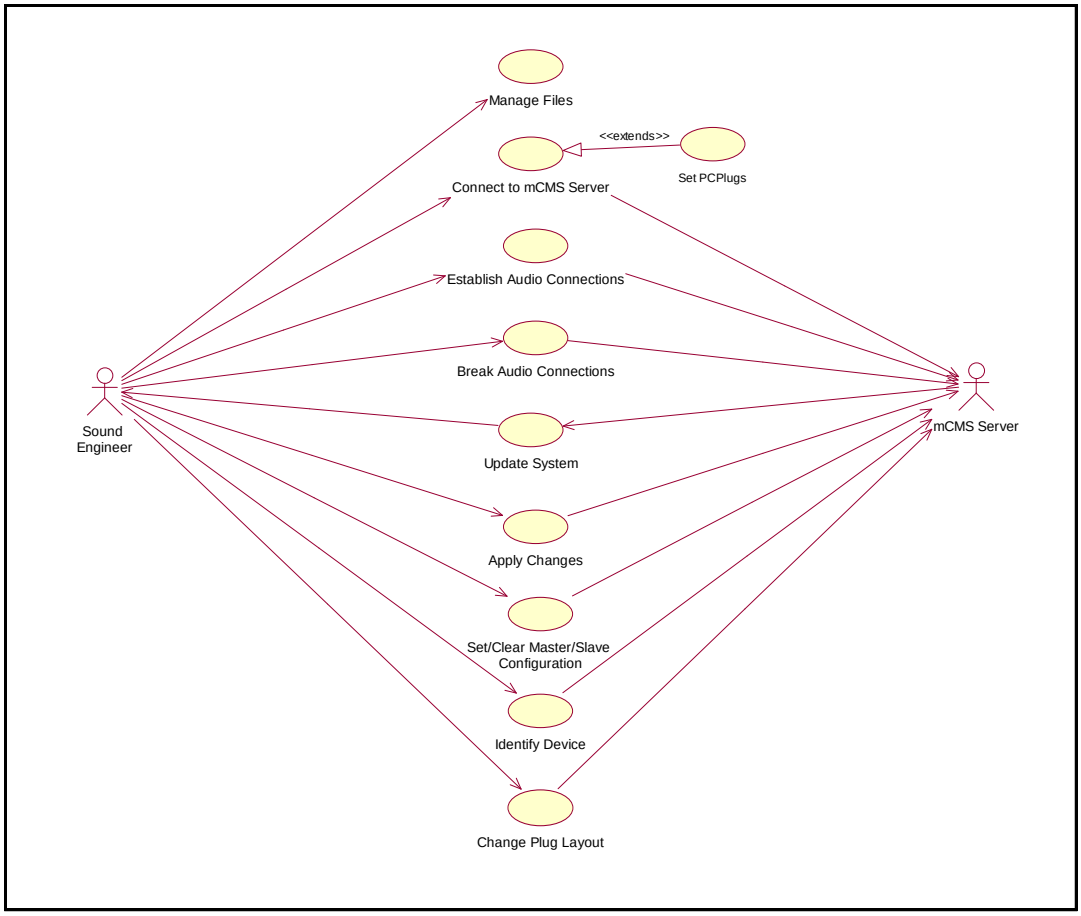

Figure 5.6: Broadcast Patchbay Use Case Diagram

5.3.1.1.2 Description of Actors

a) Sound engineer

The sound engineer is the operator of the Broadcast patchbay. In Broadcast networks, this is usually an experienced user who understands the complexity of the network. At the time of the investigations, the Broadcast patchbay had one user level, but it was desirable to have multiple user levels with different rights.

b) mCMS Server

The mCMS Server services requests from the Broadcast patchbay [*section 2.2* and *2.3*]. It resided on the same workstation as the Broadcast patchbay, but can be installed on a different workstation that the Broadcast patchbay can access remotely over any medium (wireless or dedicated Ethernet cables), and communicate with using XML messages.

5.3.1.1.3 Description of Use Cases

a) "Manage Files" Use Case

This Use Case describes a feature that enables the sound engineer to save audio routing settings into a text file on the host workstation. Saved audio routing settings files can also be opened on the Broadcast patchbay using the *Open* File menu item.

b) "Connect to mCMS Server" Use Case

At start-up, this feature allows the Broadcast patchbay to connect to the mCMS server using the server name and port number specified by the sound engineer.

c) "Establish Audio Connections" Use Case

This functionality enables the sound engineer to establish audio connections between two plugs (input and output) of the same type (audio or MIDI), on different devices on the grid-matrix of the Broadcast patchbay depending on the Mode set (either Immediate or Delayed).

d) "Break Audio Connections" Use Case

This feature allows the sound engineer to break audio connections on the network, and is based on the Mode set (either Immediate or Delayed).

e) "Update System" Use Case

The sound engineer uses this feature to force system updates and use latest information from the mCMS Server to update the patchbay display.

f) "Apply Changes" Use Case

This feature allows the sound engineer to apply changes made on the patchbay to the actual physical mLAN network. Examples include applying changed Master/Slave configuration settings, and new audio connections and disconnections.

g) "Set/Clear Master/Slave Configuration" Use Case

This feature enables the sound engineer to set and clear a word clock Master/Slave configuration on the patchbay before performing any connection management tasks, such as establishing and breaking audio connections.

h) "Identify Device" Use Case

This feature enables the sound engineer to identify a particular device on the network to work with.

i) "Change Plug Layout" Use Case

This feature allows the sound engineer to change the *Plug Layout* for any device on the mLAN network. Device *Plug Layouts* are changed to optimise bandwidth usage on mLAN networks.

5.3.1.2 Class-Driven Analysis

5.3.1.2.1 Object Model

Figure 5.7 displays the Object Model that depicts the structural relationships between objects written for the Broadcast patchbay. It is evident from the Object Model that the patchbay developed was of sufficient complexity to test the capabilities of Adobe Flash as a serious connection management application development tool.

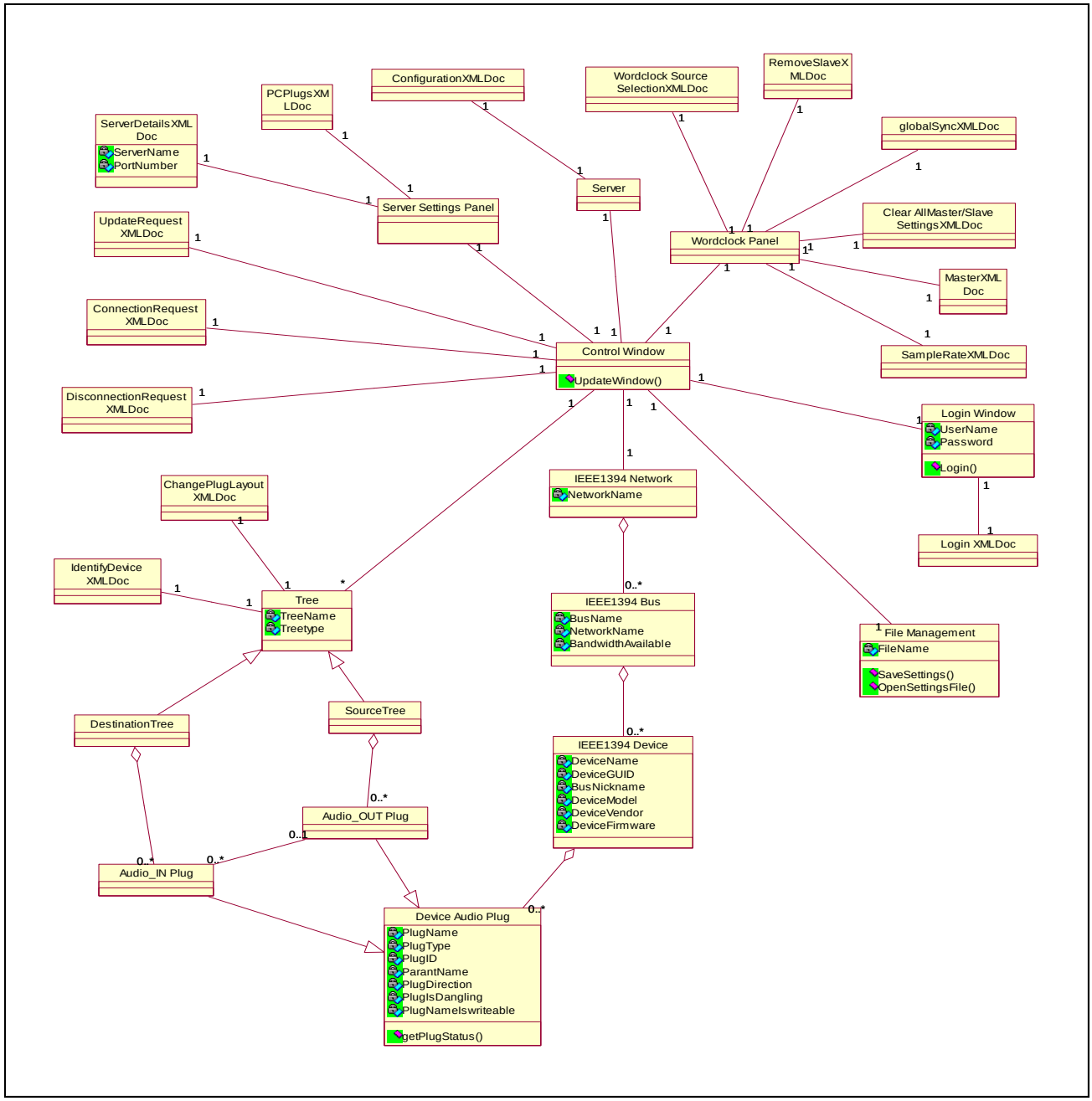

Figure 5.7: Broadcast Patchbay Object Model

The model displays the objects that model the Broadcast patchbay human interaction panels and the mLAN network. The *IEEE1394 Network* object is shown to contain an aggregation of *IEEE1394 Bus* objects on the mLAN network. Each *IEEE1394 Bus* object contains an aggregation of *IEEE1394 Device* objects, which in turn each contain an aggregation of *Device Audio Plug* objects. The *IEEE1394 Device* and the *Device Audio Plug* objects are created and instantiated by the Tree object. The model also shows the interrelationships between the various Broadcast patchbay panel objects (the *Settings* panel, the *Workclock* panel, *Login* panel, and the *Control*

Window) and their associated XML document objects. The XML document objects are classes that represent all XML messages that are used by the Broadcast patchbay to communicate with the mCMS Server.

5.3.2 Broadcast Patchbay Sequence Diagrams and Implementation

This section uses sequence diagrams to describe how groups of the Broadcast patchbay objects collaborate, and how messages are passed between them in order to fulfil the use cases of the Broadcast patchbay discussed in *section 5.3.1.1*. Each sequence diagram typically captures the behaviour of a single scenario within a use case [Maksimchuk et al, 2004].

5.3.2.1 "Connect to mCMS server" Use Case

At start-up, a TCP/IP connection is established between the Broadcast patchbay and the mCMS server. Once the connection is made, a pop-up *Settings* dialog box is displayed [*Figure 5.4*]. The user utilises this dialog box to specify the server DNS name and the port number if the patchbay is being run for the first time, and this information is not already in the registry of the host workstation. If this is successful, a *Login* pop-up dialog box appears that allows the sound engineer to specify the username and the password information to authenticate with the mCMS server. Once the authentication process finishes successfully, the connection information (the server name, the port number, the username, and the password) is stored in the registry of the host workstation. This prevents the *Settings* and *Login* dialog boxes from popping up each time after the initial setup. *Figure 5.8* is a sequence diagram that describes this start-up process, and shows the relationships between the objects involved in this process. It shows that at start-up, the Broadcast patchbay creates a *Server* and *mLANSocket* objects. The *Server* object contains the following methods:

- *getServerNameAddress ( )* – This method gets the server DNS name from the *Server Settings* dialog box.
- *getPortNumber ( )* – This method gets the server port number from the *Server Setting*s dialog box.
- *connectToServer (serverName : String, serverPort : Number)* – This method uses the *mLANSocket* object to connection to the mCMS server. It takes two

arguments, the server name and port number, that it gets from the *getServerNameAddress ( )* and *getPortNumber ( ) methods*.

- *getUserName ( )* – This method gets the sound engineer's username from the *Login* dialog box.
- *getPassword ( )* – This method gets the sound engineer's login password from the *Login* dialog box.
- *login (userName : String, userPassword : String)* – This method takes two arguments, the sound engineer's username and password and creates an XML "login" message that is sent to the mCMS server, and carries the sound engineer's username and password for authentication.

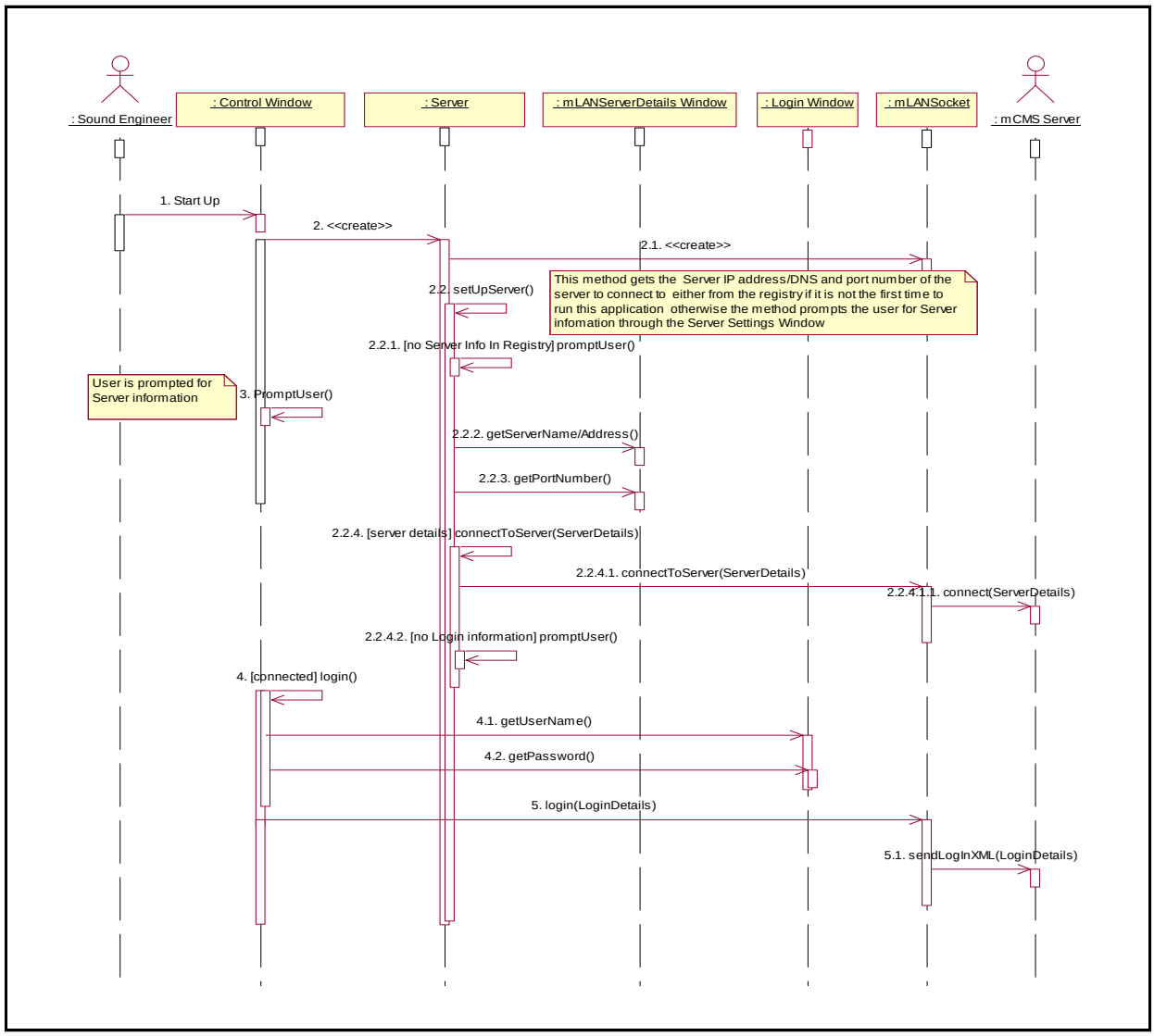

Figure 5.8: Broadcast Patchbay Start-Up Sequence Diagram

Listing 5.1 shows the XML "login" document that is sent to the mCMS server to authenticate the user. It shows that the username is "admin" and the password is "mlan".

```
- <mLANCommand version="1.0">
  - <object name="useradmin" namespace="">
    - <method name="logon">
        <parameter name="username" value="admin" />
        <parameter name="password" value="mlan" />
      </method>
    </object>
  </mLANCommand>
```

Listing 5.1: XML "login" Request Document

The sound engineer has the option of reconfiguring this connection information stored in the host workstation's registry by using the same *Settings* and *Login* dialog boxes that can be accessed via the "Dialogs" menu item, or by clicking the appropriate icon on the Tool bar of the Broadcast patchbay [*Figure 5.9*].

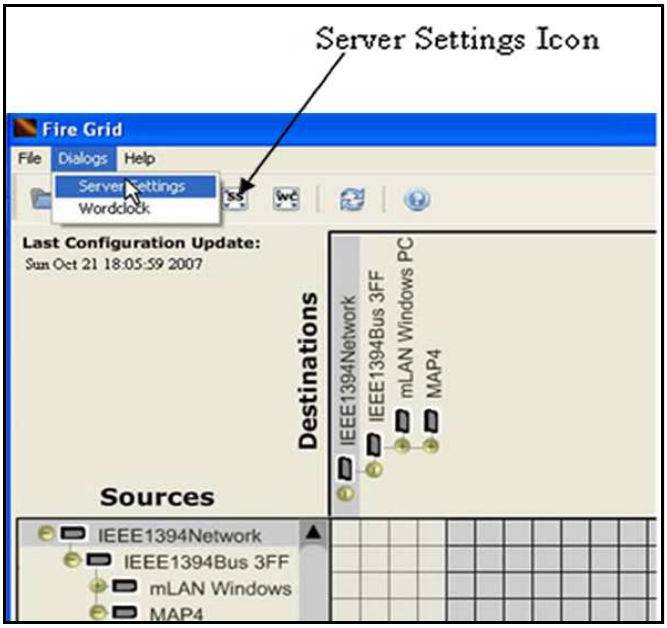

Figure 5.9: Accessing the Server Setting Dialog Box

The Use Case diagram also shows that the "Connect to mCMS Server" Use Case has an optional feature of setting up PC plugs using the *Settings* dialog box [*Figure 5.5*]. There are a total of 4915 Bandwidth Units on the mLAN network. These Bandwidth

Units are shared by all Transporter nodes on the network that wish to stream audio or MIDI on the Firewire bus. The number of source PC plugs determines how many Bandwidth Units are allocated to the PC. If not all PC plugs are needed, reducing their number to a minimum of two reduces the amount of bandwidth that is in use. This means that other devices have more bandwidth at their disposal.

Once the patchbay connects and authenticates with the mCMS server, the mCMS server invokes the Enabler module that gets the current network information. The mCMS server encapsulates this information into an XML “configuration” document [*Chapter 2: Listing 2.2*] that is sent to the Broadcast patchbay.

On receiving the XML “configuration” document, the Broadcast patchbay extracts the network information and uses it to reconstruct the mLAN network object hierarchy that represents the actual mLAN network. *Figure 5.10* shows a sequence diagram that describes this network reconstruction process.

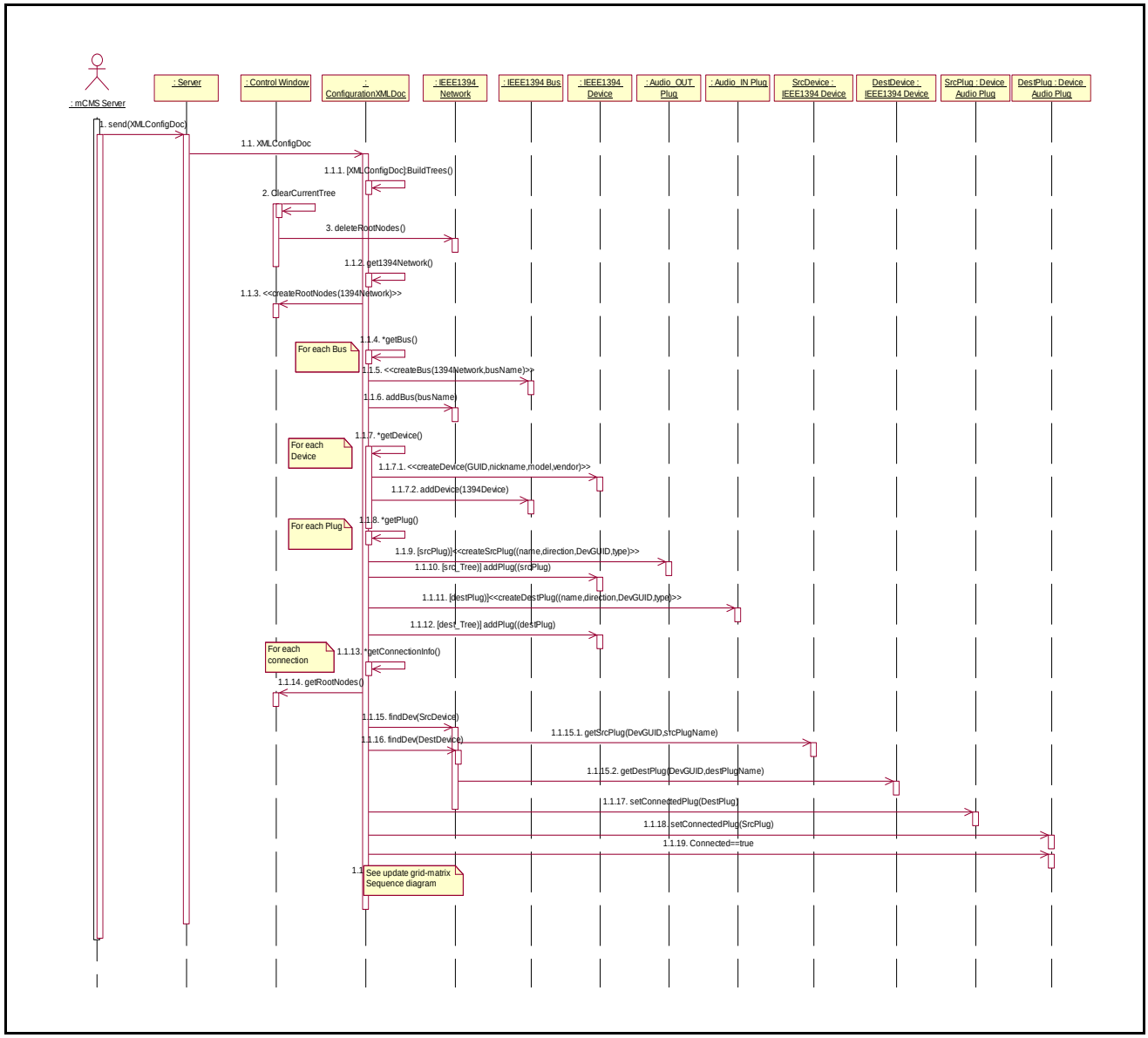

Figure 5.10: Receiving the Configuration XML Document Sequence Diagram

Figure 5.10 shows that, to reconstruct the mLAN network, the Broadcast patchbay loops through the XML "configuration" document elements as follows:

- It checks if the *Sources tree* and *Destinations tree* have nodes, and if not, it creates an *IEEE1394 Network* node on both trees.
- Loops through all bus elements, and creates a bus node on both trees for each XML bus element found.
- For each bus element, loops through all device elements and creates mLAN device nodes on both trees.

- For each device element, loops through its plug elements, and creates plug nodes on both trees, depending on whether they are inputs or outputs.

The following paragraph describes how the grid-matrix of the Broadcast is updated to reflect the latest connection information from the mCMS server.

Figure 5.11 is a sequence diagram that describes the relationships and interactions between the Broadcast patchbay objects when the grid-matrix is updated, either when the sound engineer scrolls the *Sources* tree or the *Destinations* tree, or whenever the patchbay receives an XML "configuration" document from the mCMS server:

- The patchbay loops through all visible nodes on the *Sources* tree, and checks the value of the Boolean variable "Connection_Ability" for each node.
- If the Boolean variable value is "false", it means that node is not a plug, therefore it marks the row cross-points corresponding to this node on the grid-matrix as unusable (unusable cross-points are shown greyed-out on the grid-matrix [*Figure 5.11*]).
- If the Boolean variable value is "true", that node is a source plug node. For this plug, it checks its Boolean variable "Connected". If its value is "false", it does nothing, otherwise it loops through all visible nodes on the *Destinations* tree.
- For each visible node on the *Destinations* tree, it checks its Boolean variable "Connection_Ability" value. If its value is "false", it marks the corresponding cross-point as unusable, otherwise it checks if its "plug ID" attribute is the same as that held by the variable "connectedDestination" of the source plug node on the *Sources* tree and that the GUID of both plugs' parent nodes are different. If that is "true", a live connection exists between the two plugs, and a green graphic box is drawn at their cross-point on the grid-matrix [*Figure 5.11*].

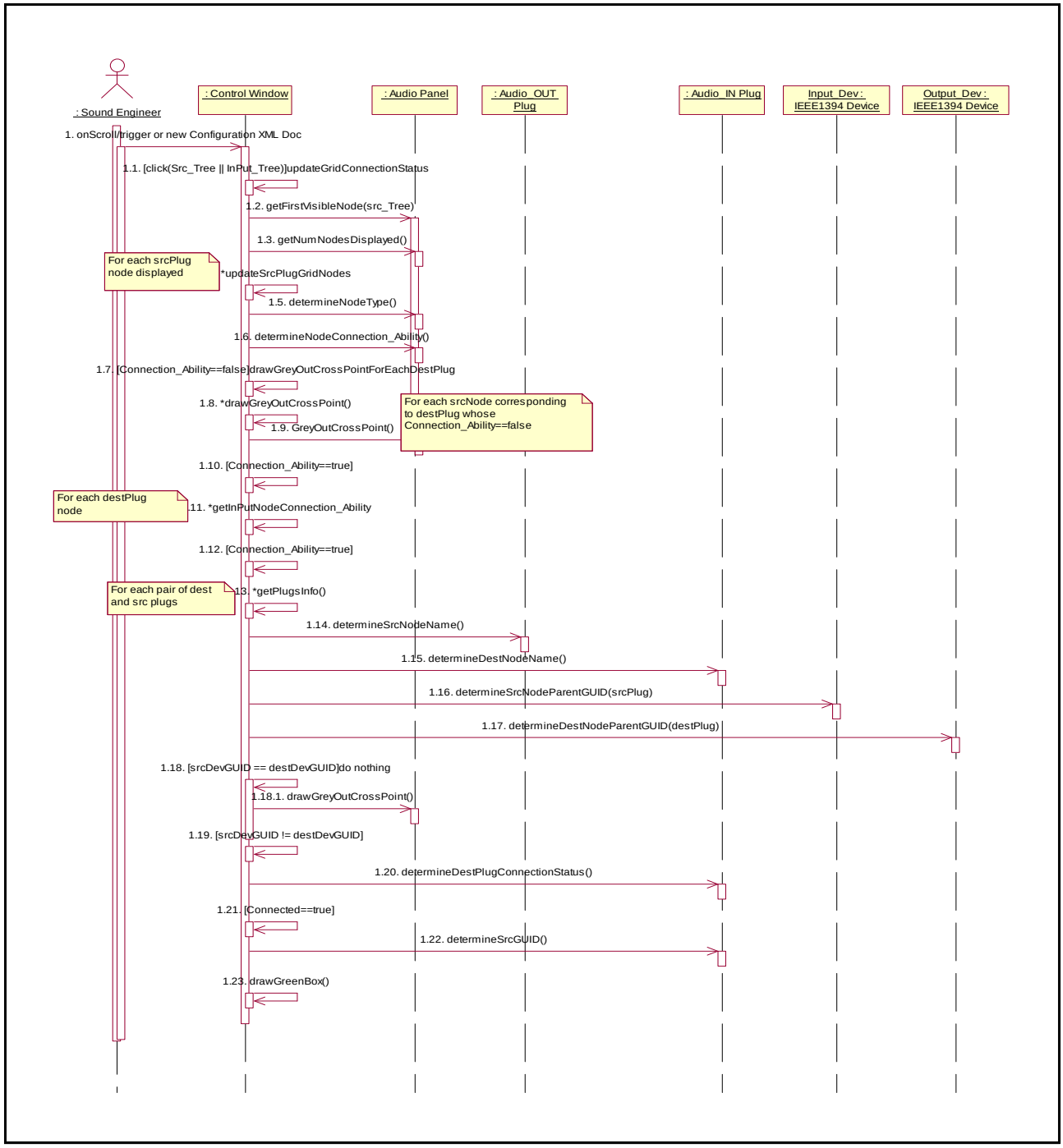

Figure 5.11: Updating the Grid-Matrix

5.3.2.2 "Establishing Audio Connections" Use Case

Audio connections can be established on the Broadcast patchbay in two ways, depending on the current mode of the patchbay, whether it is in "Delayed Mode" or in "Immediate Mode". Audio connections are made between mLAN plugs of the same type that are on different mLAN devices. Plugs of different types cannot be connected (an audio plug can not be connected to a MIDI plug).

a) Establishing Audio Connections in "Delayed Mode"

If the patchbay is in “Delayed Mode”, the sound engineer makes audio connections by clicking all plug cross-points on the grid-matrix, of plug pairs, on different devices to be connected. This creates blue graphic circles on the clicked cross-points, which represent pending connections [*Figure 5.12*]. A pending connection is a connection that has been made on the patchbay but that has not yet been applied to the actual physical mLAN network and has no streaming audio.

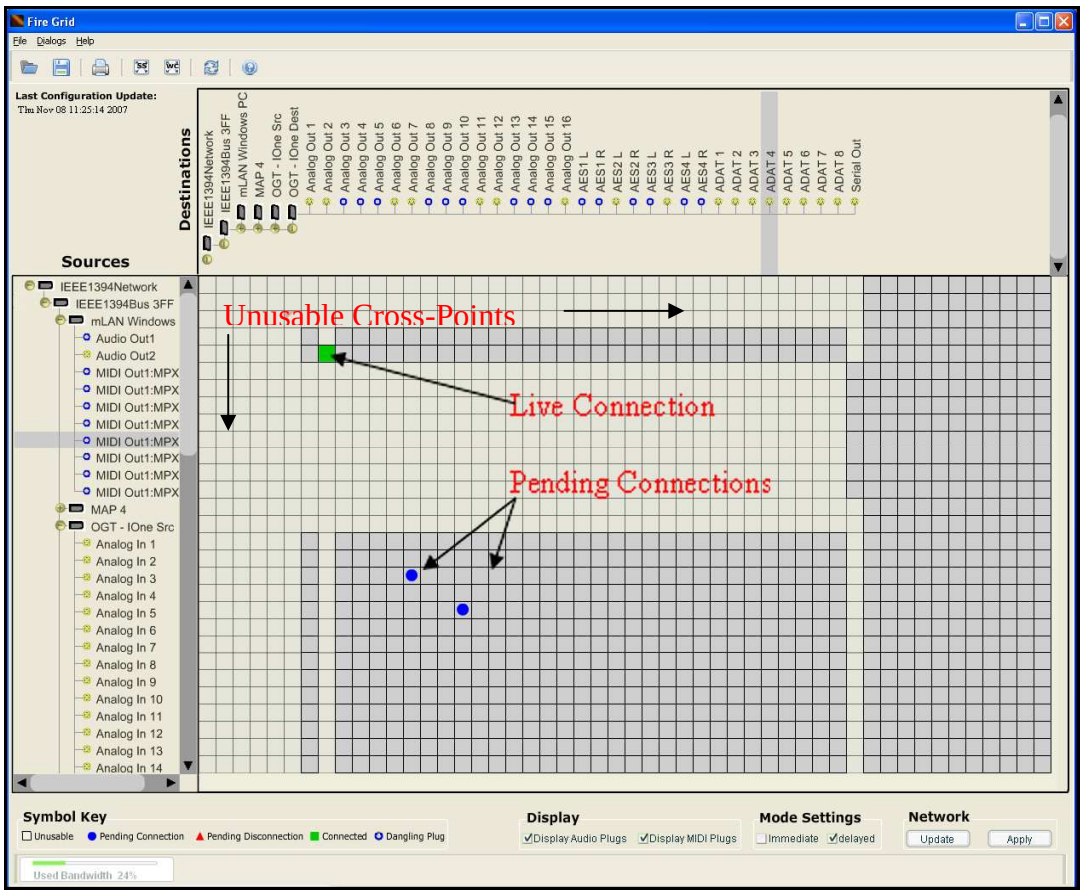

Figure 5.12: Pending and Live Connections on the Grid-Matrix

In *Figure 5.12*, the sound engineer is connecting output plugs *Analog In 3* and *Analog In 5* on the *OTG-IOne Src* device (displayed on the *Sources tree*) to input plugs *Analog Out 7* and *Analog Out 10* on the *OTG-IOne Dest* device (displayed in the *Destinations tree*).

When the user clicks the *Apply* button, the patchbay loops through all pending connection cross-points, each time creating an XML “connection request” document

[*Listing 5.2*]. The XML "connection request" documents are sent to the mCMS server which implements the connections through the Enabler module.

```
<?xml version="1.0" encoding="utf-16" ?>
- <mLANServerCommand version="1.0">
  - <object name="patch">
    - <method name="connect">
        <parameter name="sourceGUID" value="0013f00400400011" />
        <parameter name="sourcePlugType" value="audio" />
        <parameter name="sourcePlugID" value="1" />
        <parameter name="destinationGUID" value="0013f00400000014" />
        <parameter name="destinationPlugType" value="audio" />
        <parameter name="destinationPlugID" value="33" />
      </method>
    </object>
  </mLANServerCommand>
```

Listing 5:2: XML "connection" Request Document

Listing 5:2 shows an XML "connection request" document that requests a connection between the output plug with sourcePlugID "1" (on a device with GUID "0013f00400011" and of type "audio") to a destination plug with destinationPlugID "33" (on a device with GUID "0013f00400000014", and also of type "audio"). If the connect request is implemented, the blue pending connection graphic changes to a green box that reflects a live connection, and audio is routed between the two newly connected plugs.

Figure 5.13 shows a sequence diagram that describes the process of establishing audio connections in "Delayed Mode" as discussed in the preceding section. As the user moves the mouse-pointer on the grid-matrix, the patchbay selects the nodes on the *Sources tree* and the *Destinations tree*. Assuming the user wants to establish an audio connection, and clicks a cross-point of two selected plug nodes:

- The input plug ID and its parent node GUID values are assigned to the corresponding source plug node variables "connectedDestination" and "connectedGUID" respectively. This occurs whether the application is in "Delayed Mode" or "Immediate Mode" [*Figure 5.12*].
- Since the patchbay is in "Delayed Mode", the source plug node Boolean variable "n_Connected" value is set to "true".

- When the *Apply* button is clicked, the patchbay loops through all visible *Sources* tree nodes.
- For each source plug node found, the patchbay checks its "n_Connected" variable. If its value is "true", an XML "connection request" document is created and sent to the mCMS server with the destination plug node ID and its parent device node GUID, and a green live connection graphic box is drawn at the appropriate cross-point.

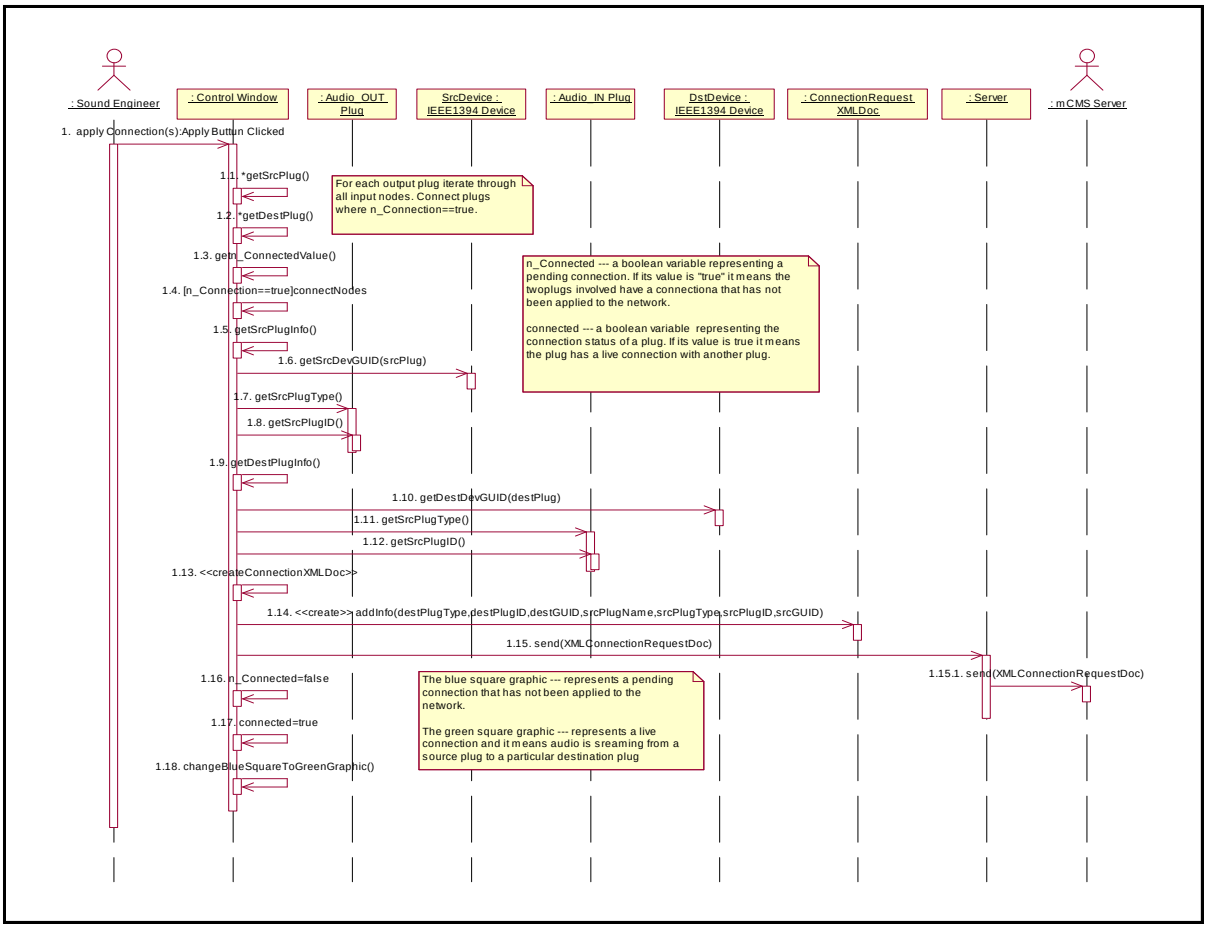

Figure 5.13: Establishing Audio Connections in "Delayed Mode"

b) Establishing Audio Connections in "Immediate Mode"

If the patchbay is in "Immediate Mode", when the sound engineer clicks a cross-point of two plugs to be connected, an XML "connection request" document is automatically created and sent to the mCMS server without any further involvement of the sound engineer. The green graphic box is drawn on the clicked grid box that represents a live connection between the newly connected plugs. Establishing audio connections functionality is implemented as follows:

- When a cross-point of two plugs (on different devices and of the same type) is clicked on the grid-matrix, the patchbay captures the input and output plug IDs, their parent device GUID values, and their type (MIDI or audio). The input plug ID and its parent GUID are assigned to the output plug device variables “connectedDestination” and “connectedGUID” respectively.
- An XML “connection request” document [*Listing 5.2*] is created, which carries the input and output plug IDs, their parent device GUID values and their type, and sent to the mCMS server that implements the request.
- The patchbay creates a green graphic box, which represents the newly created connection.

Only one connection can be made at a time in “Immediate Mode”, in contrast to many connections that can be made at a time when the patchbay is in “Delayed Mode”.

5.3.2.3 “Breaking Audio Connections” Use Case

Audio connections can be broken as easily as they can be made. Breaking audio connections can be done in two ways, depending on whether the patchbay is in “Delayed Mode” or in “Immediate Mode”.

a) Breaking Audio Connections in “Delayed Mode”

If the patchbay is in “Delayed Mode”, the sound engineer breaks audio connections by clicking all live connections (depicted by green graphic boxes on the grid-matrix). When a live connection graphic box is clicked in “Delayed Mode”, it changes to a red rectangle graphic, which represents a pending disconnection. *Figure 5.14* reflects that a pending disconnection exists between the source plug *Audio Out 2* on the *mLAN Windows PC* device (on the *Sources tree*) and the destination plug *Analog Out 2* on the *OGT-IOne Dest* device (on the *Destinations tree*).

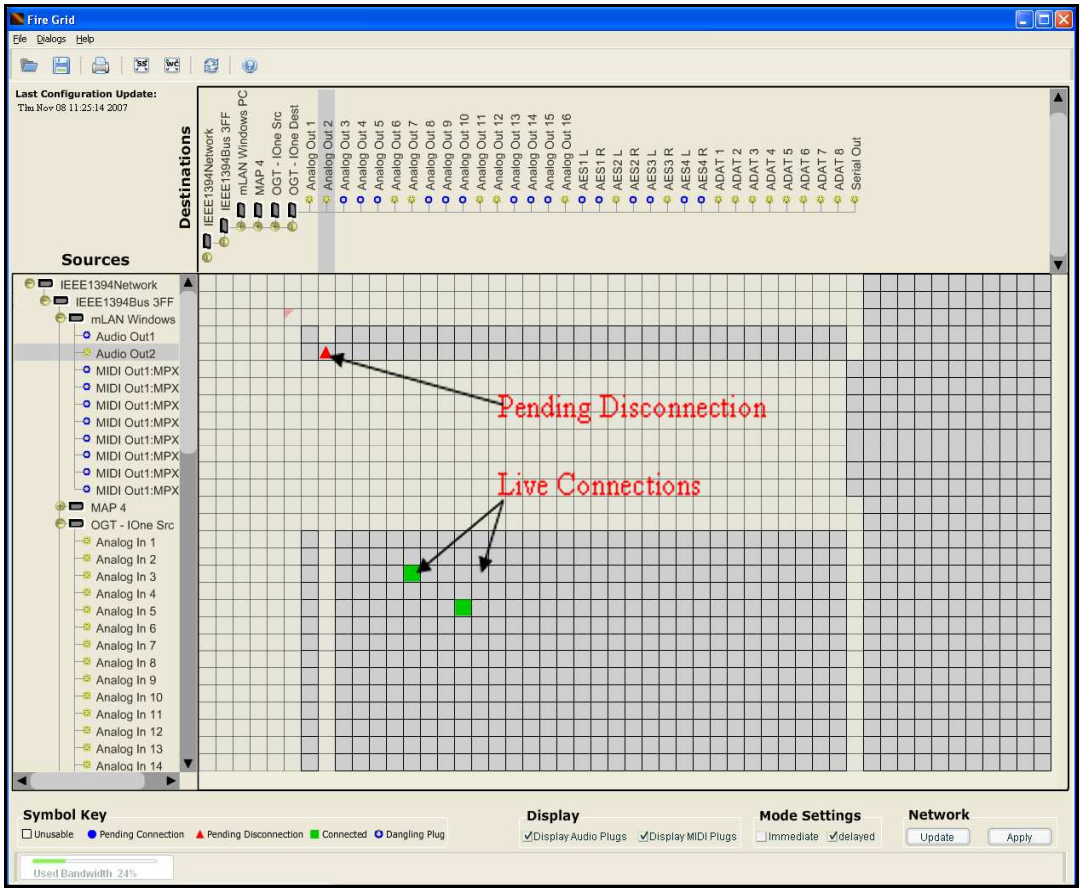

Figure 5.14: Pending and Live Connections on the Grid-Matrix

To commit the disconnections to the mLAN network, the sound engineer clicks the *Apply* button. This action clears all pending disconnection triangles and sends XML "disconnection request" messages to the mCMS server for each disconnected pair of plugs. *Listing 5.3* shows a typical XML "disconnection request" message sent to the mCMS server that carries information about the destination plug to be disconnected. The XML "disconnection request" message requests a disconnection of the audio input plug with ID "32" on a device with GUID "0013f00400000014" [*Listing 5.3*]. If the disconnection is performed successfully, the patchbay grid-matrix is updated appropriately.

```
<?xml version="1.0" encoding="utf-16" ?>
- <mLANServerCommand version="1.0">
  - <object name="patch">
    - <method name="disconnect">
        <parameter name="destinationGUID" value="0013f00400000014" />
        <parameter name="destinationPlugType" value="audio" />
        <parameter name="destinationPlugID" value="32" />
      </method>
    </object>
  </mLANServerCommand>
```

Listing 5.3: XML "disconnection" Request Document

Figure 5.15 shows the relationships and interaction between the Broadcast patchbay objects when disconnecting plugs in "Delayed Mode". It shows that:

- When the sound engineer clicks a particular cross-point of two connected plugs, the source plug node ID and its parent node GUID are assigned to the source plug node variables "n_Destination" and "n_disconnectedGUID" respectively.
- If the patchbay is in "Delayed Mode", the source plug node Boolean variable "n_Disconnected" is set to "true", which represents a new disconnection that has not been applied to the mLAN network.
- When the *Apply* button is clicked, the patchbay loops through all *Sources* tree nodes. For each source plug node found, the patchbay checks its "n_Disconnected" attribute. If its value is "true", an XML "disconnection request" document is created and sent to the server with the destination plug node ID and its parent node GUID. The red graphic triangle is cleared at the cross-point.

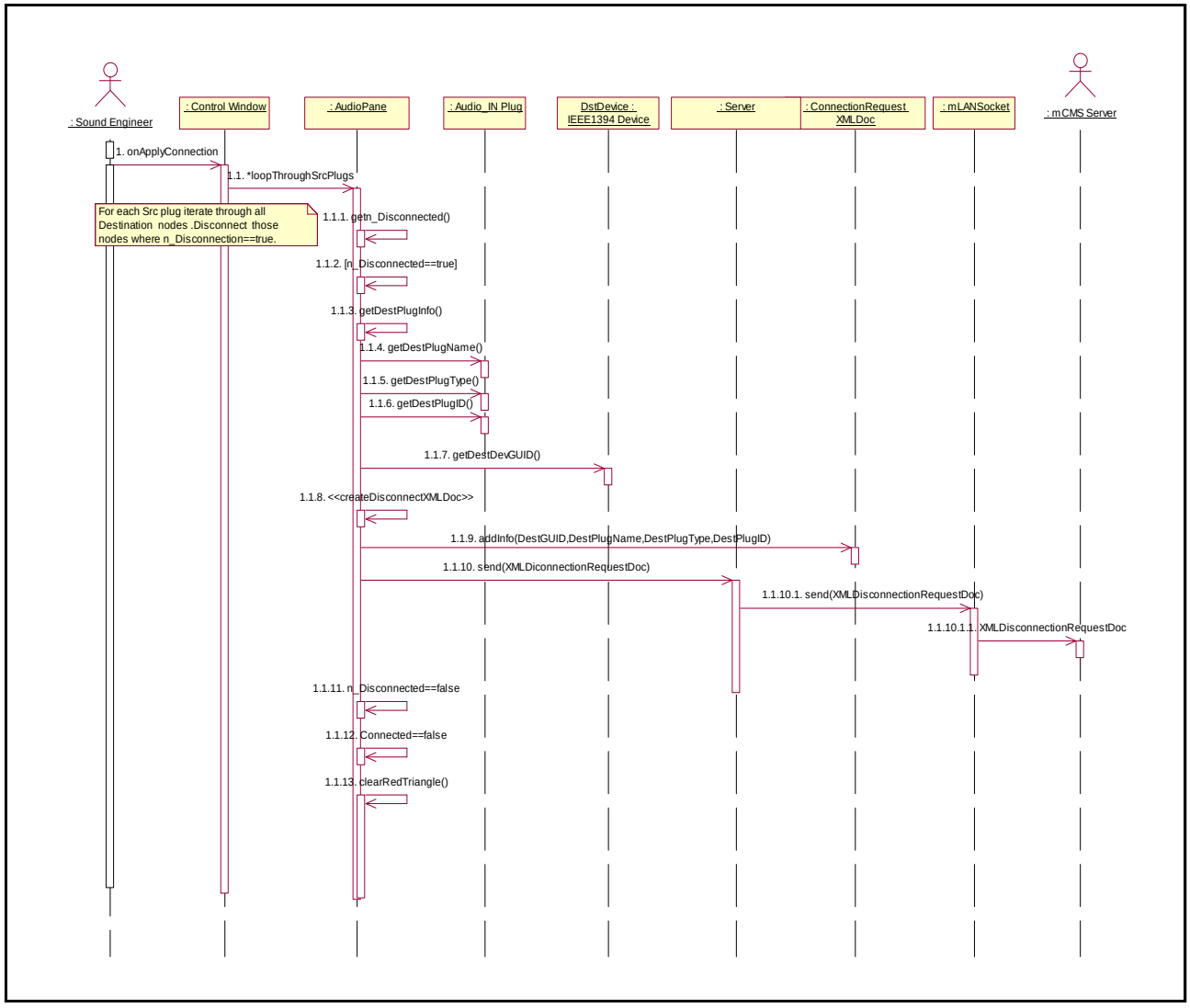

Figure 5.15: Breaking Audio Connections in "Delayed Mode"

b) Breaking Audio Connections in "Immediate Mode"

If the patchbay is in "Immediate Mode", the sound engineer breaks audio connections by clicking each live connection (depicted by green graphic boxes on the grid-matrix) to be disconnected. Once a live connection is clicked, its connection is immediately disconnected without the need for the user to click the *Apply* button. The green live connection graphic box is automatically cleared and the patchbay updated appropriately. Breaking audio connections functionality is implemented as follows:

- When the sound engineer clicks a cross-point of two connected plugs, the input plug ID and its parent node GUID are assigned to the outplug plug node variables "n_Destination" and "n_disconnectedGUID" respectively.

- An XML “disconnection request” document [*Listing 5.3*] is created and sent to the server with the input plug ID, its type, and its parent device GUID.
- The green graphic box is cleared at the cross-point to reflect the disconnection.

In “Immediate Mode”, only one disconnection can be disconnected at a time whilst in “Delayed Mode” many disconnections can be performed at a time.

5.3.2.4 “Setting/Clearing Master/Slave Configurations” Use Case

For any two mLAN devices on the mLAN network to transmit or receive audio packets from each other, they need to be synchronised to transmit or receive at the same sample rate. The Broadcast patchbay uses the *Wordclock* panel to set the following word clock Master/Slave configurations:

- Change a master device word clock sample rate and its word clock source.
- Set a global master device for the whole mLAN network.
- Enslave a device to a master device.
- Release all slave devices for a particular master device.
- Remove a single slave device from its master device.

The *Wordclock* panel can be accessed using the *Wordclock* panel icon on the main *Control Window* or through the “Dialogs” menu item [*Figure 5.9*]. To set a word clock Master/Slave configuration, the sound engineer has to specify the word clock source and the sample rate for the device to be configured as the master device. *Figure 5.16* and *5.17* show two panels that are used to set the word clock source and the sample rate for a device called *MAP4*. To access the *Wordclock Source* panel the user right-clicks the device on the “Master Capable Devices” section of the *Wordclock* panel [*Figure 5.15*]. On the submenu that appears, the user selects the “Wordclock Source Selection” menu item. A sub-panel [*Figure 5.16*] will be displayed on which the user can select the desired word clock source and sample rate.

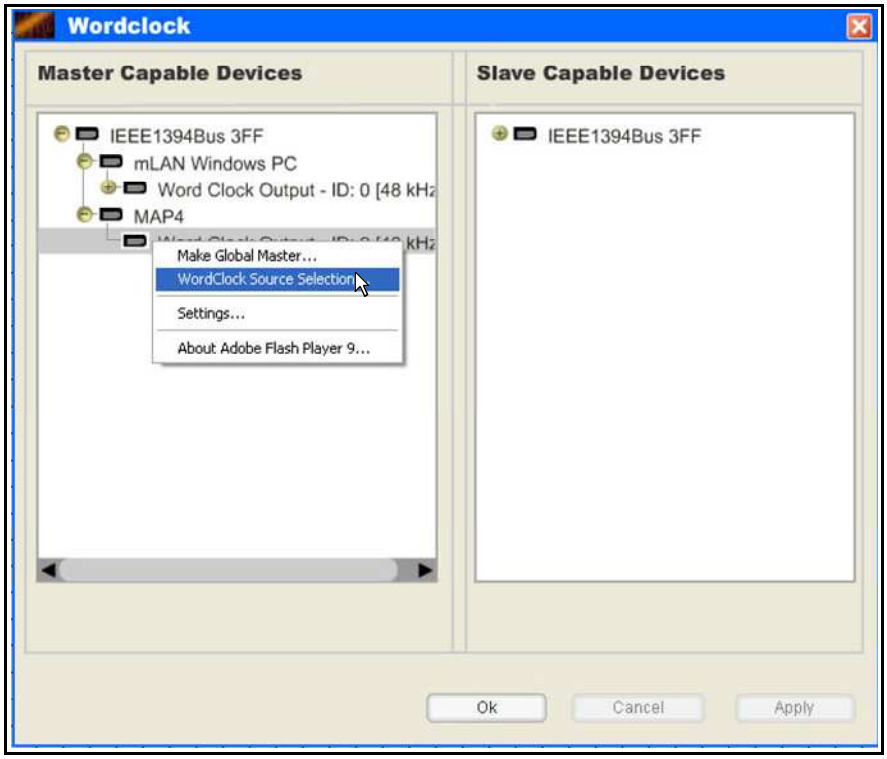

Figure 5.16: Accessing the Wordclock Source Panel

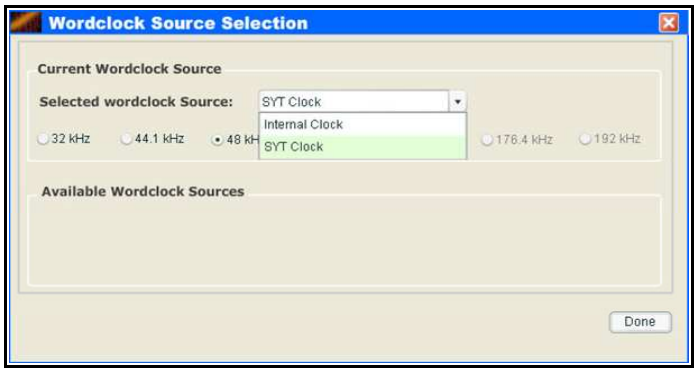

Figure 5.17: Setting the Word Clock Source and Sample Rate

To select a *word clock source*, the sound engineer uses the "Selected Wordclock Source" combo box that lists all possible word clock sources for the selected device. *Figure 5.17* shows two possible word clock sources for the *MAP4* device, namely the "Internal Clock" and the "SYT Clock" word clock sources. The panel also displays possible sample rates for the *MAP4* device. The current sample rate for the *MAP4* device is 48 kHz shown by a selected radio button [*Figure 5.17*]. Supported sample rates for the *MAP4* device are 32 kHz, 44.1 kHz, 48 kHz, 88.2 kHz and 96 kHz. Sample rates that are not supported are greyed-out (176.4 kHz and 192 kHz sample rates) [*Figure 5.17*]. To change the sample rate, the user simply selects the

appropriate radio button and clicks the *Done* button to apply the changes to the network. An XML "wordclock source" document is created and sent to the mCMS server.

Once the word clock source and the sample rate have been selected and applied to the device to be made a master, the global master can now be set for the whole mLAN network. To set a global master unit, the sound engineer right-clicks the device on the "Master Capable Devices" section of the *Wordclock* panel and selects the "Make Global Master" submenu item [*Figure 5.18*].

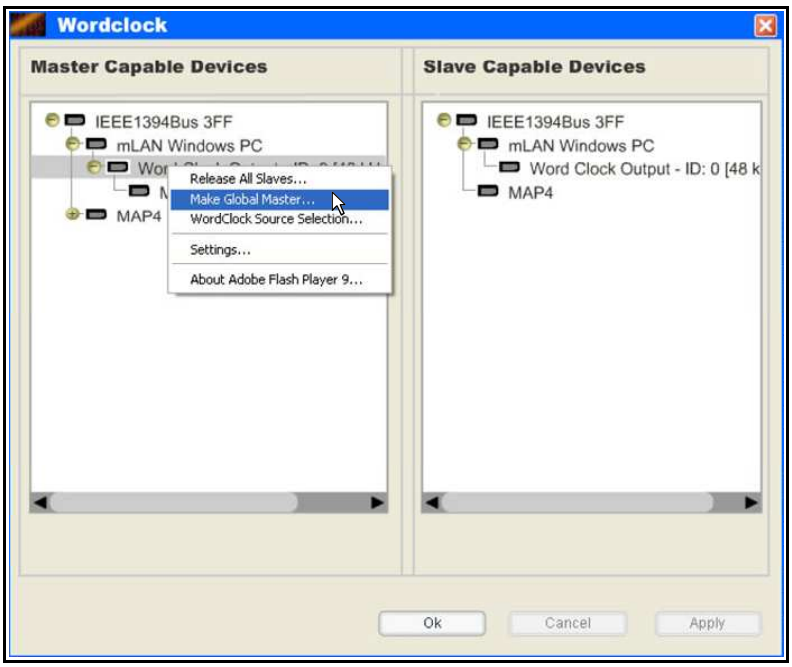

Figure 5.18: Setting the Word clock Source and Sample Rate

If the configuration is legal and there is enough bandwidth on the network, an XML "global master" request document [*Listing 5.4*] is sent to the mCMS server, and the configuration committed to the network. The *Wordclock* panel is updated appropriately, and all devices, except the device being made a global master, are deleted from the "Slave Capable Devices" section and attached to the newly set global master device node.

```
<?xml version="1.0" encoding="utf-16" ?>
- <mLANServerCommand version="1.0">
  - <object name="samplerate">
    - <method name="syncsetup">
        <parameter name="masterGUID" value="0013f00400400011" />
        <parameter name="masterWordClockOutputID" value="0" />
        <parameter name="masterCurrentSyncSourceID" value="1" />
        <parameter name="masterSampleRate" value="0000BB80" />
        <parameter name="slave" value="NODE_GUID='0013f00400000014',WORD_CLOCK_ID='0'" />
      </method>
    </object>
  </mLANServerCommand>
```

Listing 5:4: XML "global master" Request Document

Figure 5.19 describes the process of setting a global master when the operator right-clicks and selects the "Make Global Master" menu item for a particular device on the mLAN network:

- The patchbay gets the master device details (its GUID, its sample rate, its wordClockOutputID, and its currentSyncSourceID).
- It loops through all its slave devices and gets their GUID values (making sure that their "wordClockOutputID" variable value is the same as that of the master device).
- It creates and sends an XML "global master" document to the mCMS server carrying the master device and slave devices information.
- The "Capable Masters Device" and "Slave Capable Device" tree are updated appropriately, and all slave device nodes are attached to the global master device.

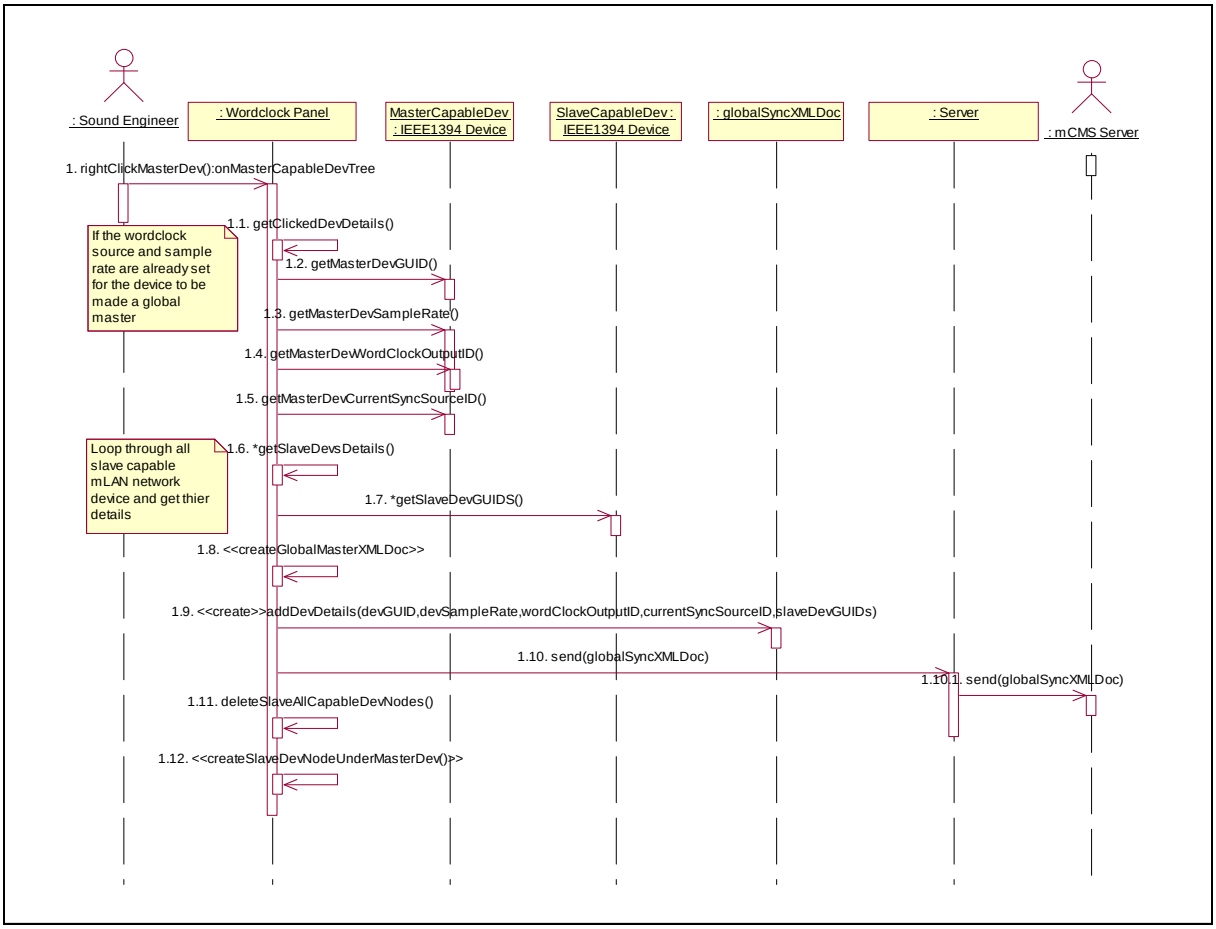

Figure 5.19: Setting a Global Master Device

It is possible to have two or more master devices on one mLAN network. In that case, slave devices are attached to their master devices individually [*Figure 5.20*]. To enslave a single device to its master device, the sound engineer selects and highlights the master device on the "Master Capable Device" section of the *Wordclock* panel. This is shown in *Figure 5.20* for the *MAP4* device. On the "Slave Capable Devices" section, the sound engineer right-clicks the device to be made a slave of the selected master, and selects the "Make A Slave" submenu item. An XML "syncsetup" message is automatically sent to the mCMS server for implementation. Assuming the sample rate and word clock source has been set:

- When the sound engineer selects and highlights a node on the "Master Capable Device" section of the *Wordclock* panel to be a master, its "new_Master" variable is set to "true", its GUID and sample rate are assigned to the global variables "masterGUID" and "masterSampleRate" respectively.

- When the sound engineer selects a slave node on the “Slave Capable Devices” section of the *Wordclock* panel, it is deleted and attached to the master node on the “Master Capable Device” tree.
- An XML “syncsetup” document [*Appendix B: Listing 17*] is created that carries the sample rate and GUID values for the master device, and the array of all its slave device GUID values including the newly attached slave devices. The XML document is sent to the mCMS server to implement the setting.

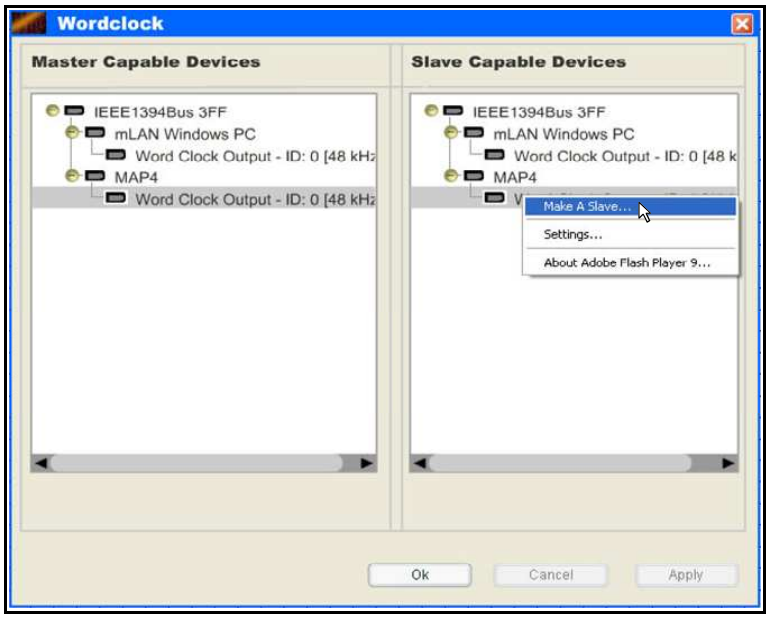

Figure 5.20: Setting individual Slave Devices

All slave devices for a particular master device can also be easily released by right-clicking the master device and selecting the “Release all Slaves” submenu item [*Figure 5.21*].

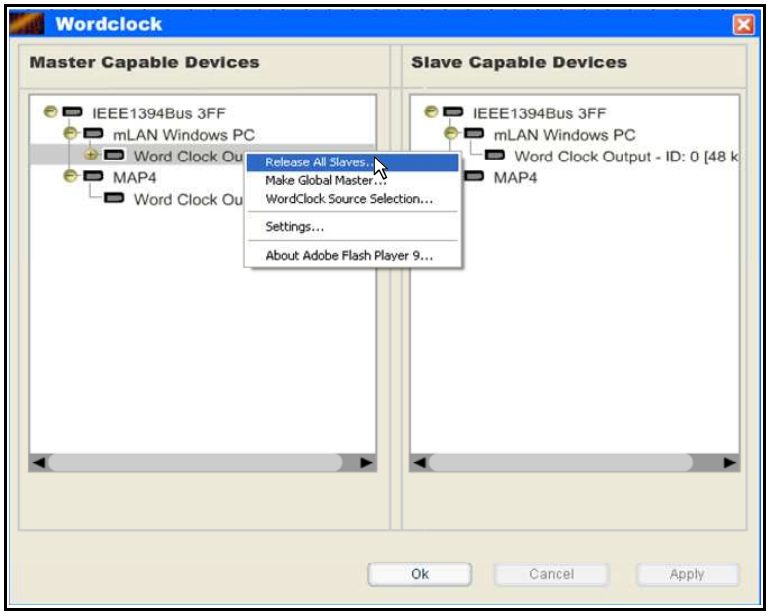

Figure 5.21: Release all Master Device Slave Devices

An XML "syncsetup" message is sent to the mCMS server, which automatically implements the request and the *Wordclock* panel is updated appropriately by removing the slave device nodes from the cleared master device, and attaching them to their respective slave device nodes on the "Slave Capable Devices" section. When the sound engineer selects the "Release all Slaves" submenu item:

- All its slave device nodes are deleted from the "Capable Master Device" tree and attached to their individual slave node on the "Capable Slave Device" tree.
- An XML "syncsetup" document is sent to the mCMS server that contains the sample rate and GUID values of the master device for which slaves have been cleared.
- Slave device nodes are deleted from their master device node on the "Capable Master Device" tree and attached to their individual slave nodes on the "Capable Slave Device" tree.

It is possible to remove only one slave device from a master device with more than one slave device. This can be done by simply right-clicking the slave device to be removed from its master on the "Master Capable Devices" section, and selecting the "Remove Slave" submenu item [*Figure 5.22*].

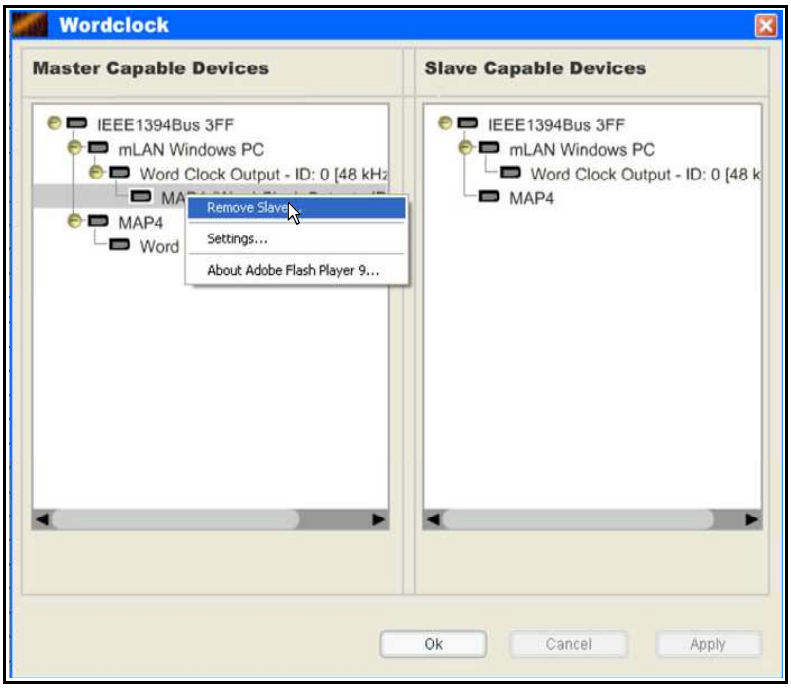

Figure 5.22: Removing a Particular Slave Device

An XML "syncsetup" message is sent to the mCMS server, and the slave device node is automatically removed from its master device and attached to its slave device node on the "Slave Capable Devices". When the sound engineer selects the "Remove Slave" submenu item on a particular slave node:

- Its slave node is deleted from the "Capable Master Device" tree and attached to its slave node on the "Capable Slave Device" tree.
- An XML "syncsetup" document is sent to the mCMS server that contains the sample rate, GUID value of the master device and the GUIDs of its remaining slave devices.

5.3.2.5 "Identify Device" Use Case

In a large mLAN network, it is often necessary for the sound engineer to identify a particular device to work with on the network. The sound engineer can do this on the Broadcast patchbay by right-clicking the device node on the *Sources tree* and selecting the "Identify Device" submenu item [*Figure 5.23*].

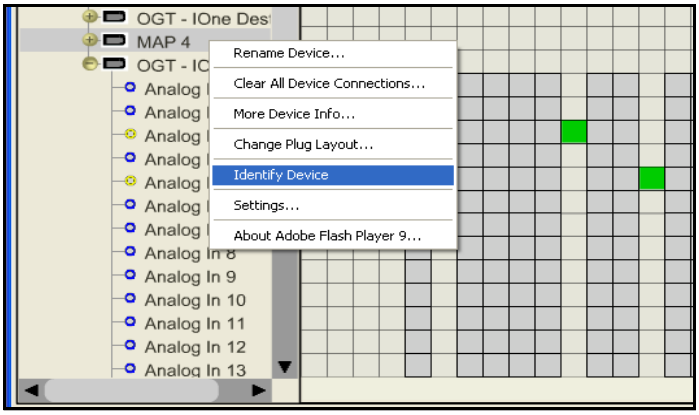

Figure 5.23: Identifying a Device

The patchbay automatically creates an XML "identify device" request document [*Listing 5.5*] that is sent to the mCMS server, which contains the GUID of the device to be identified. If the request is implemented successfully, the device will respond by flashing one or more of its LEDs.

```
<?xml version="1.0" encoding="UTF-8" ?>
<mLANServerCommand version="1.0">
- <object name="patch">
  - <method name="identify">
      <parameter name="GUID" value="0013f00400400011" />
    </method>
  </object>
</mLANServerCommand>
```

Listing 5.5: XML "identify device" Request Document

5.3.2.6 "Change Plug Layout" Use Case

To change the Plug Layout of a device, the sound engineer right-clicks the device node on the *Sources tree* and selects the "Change Plug Layout" submenu item [*Figure 5.14*]. A *Plug Layout* panel opens [*Figure 5.25*]. It displays the current Plug Layout of the selected device. The sound engineer can now use the "Select Plug Layout" combo box to select a new Plug Layout to apply to this device.

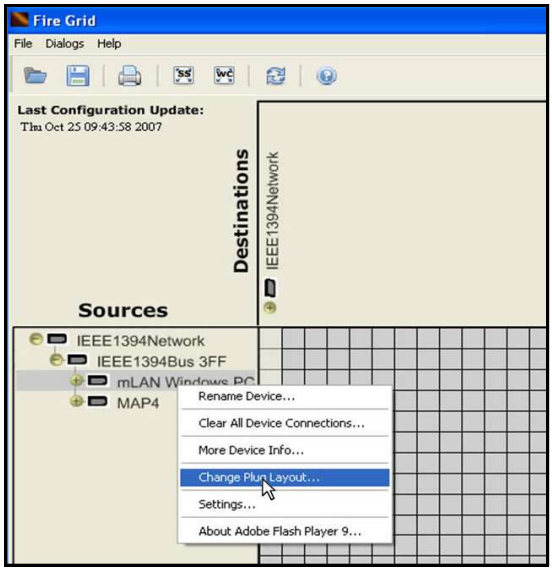

Figure 5.24: Change Plug Layout

When the user has chosen the appropriate Plug Layout on the *Plug Layout* panel [*Figure 5:25*], the setting is applied by clicking the *Apply* Button.

Figure 5.25: Select Plug Layout Panel

Listing 5.6 shows a typical XML "Plug Layout" message that is sent to the mCMS server when the sound engineer applies the Plug Layout setting by clicking the *Apply* Button. It is requesting the Enabler module to apply a Plug Layout with ID "1" to a device with GUID "0013f00400000014". If the Plug Layout is successfully applied, the mCMS server sends an XML "configuration" document to the patchbay with updated network information that the patchbay in turn uses to updates its display. The effect of changing a device Plug Layout setting is reducing or increasing the number of plugs for the device depending on the chosen Plug Layout.

```
<?xml version="1.0" encoding="utf-16" ?>
- <mLANServerCommand version="1.0">
  - <object name="patch">
    - <method name="setCurrentPlugLayout">
        <parameter name="GUID" value="0013f00400000014" />
        <parameter name="plugLayoutID" value="1" />
      </method>
    </object>
  </mLANServerCommand>
```

Listing 5.6: XML "plug layout" Request Document

5.3.2.7 "Clear Dangling Connections" Use Case

Dangling connections can be cleared in two ways, either by connecting a plug with a dangling connection, or by clearing the dangling connection. When the sound engineer attempts to make a connection between two mLAN plugs, one of which has a dangling connection, the dangling connection is first cleared automatically and then the connection made [*section 5.3.2.2.1*]. When a dangling connection is cleared, an XML "clear dangling connection" message is sent to the mCMS server that carries the information about the plug with a dangling connection. The information carried by the XML "clear dangling connection" message includes the GUID of the parent device of the plug with a dangling connection, and the plug specific information that includes the plug transmitting direction (in or out), the plug ID, and the plug type (audio or MIDI) [*Listing 5.7*].

```
<?xml version="1.0" encoding="utf-16" ?>
- <mLANServerCommand version="1.0">
  - <object name="patch">
    - <method name="clearDanglingConnection">
        <parameter name="GUID" value="0013f00400400011" />
        <parameter name="direction" value="out" />
        <parameter name="plugID" value="15" />
        <parameter name="plugType" value="audio" />
      </method>
    </object>
  </mLANServerCommand>
```

Listing 5:7: XML "clear dangling connection" Request Document

Alternatively, dangling connections can be cleared directly by left-clicking the blue round icon of the plug with a dangling connection on the device trees. This automatically clears the dangling connection by changing the blue dangling connection icon of the clicked plug, and sending an XML "clear Dangling

connection" message [*Listing 5.7*] to the mCMS server for implementation [*Figure 5:2*]. This method of clearing dangling connections is described by the following sequence diagram [*Figure 5.26*]

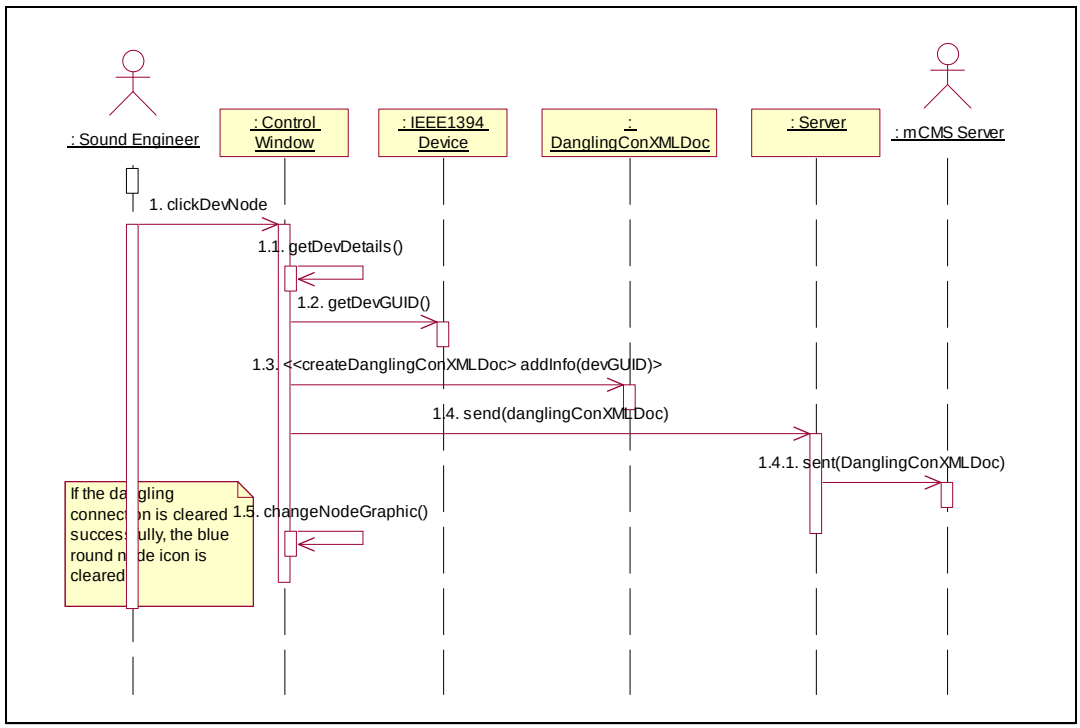

Figure 5.26: Clearing Dangling Connections

The patchbay gets the GUID value of the clicked node on the tree that has a dangling connection, creates and sends an XML "clear dangling connection" document to the mCMS server that carries the GUID of the device to be cleared. If the Enabler successfully clears the dangling connection, the blue round icon on the tree node (newly cleared node) is changed to the normal yellow node icon that depicts a tree node representing a plug with no dangling connection [*Figure 5.26*].

5.3.2.8 "Change Device Name" Use Case

At the time of this investigation, the mLAN device firmware allowed the renaming of devices only, plug names could not be changed. To change the name of a device, the sound engineer would right-click the device on the *Sources tree* and selects the "Rename Device" submenu item. A *Rename Device* panel [*Figure 5.27*] appears on which the sound engineer can specify the new name of the device.

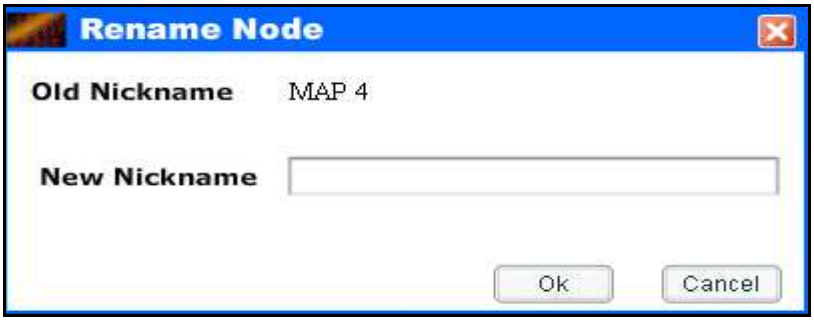

Figure 5.27: Rename Device Panel

When the sound engineer clicks the *Ok* button, an *XML* "rename device" message [*Appendix B: Listing 11*] is created that contains the GUID of the device to be renamed and the new name for that device. If the renaming process succeeds, the label name of the device is automatically changed to the new name on both device trees on the interface [*Figure 5.28*].

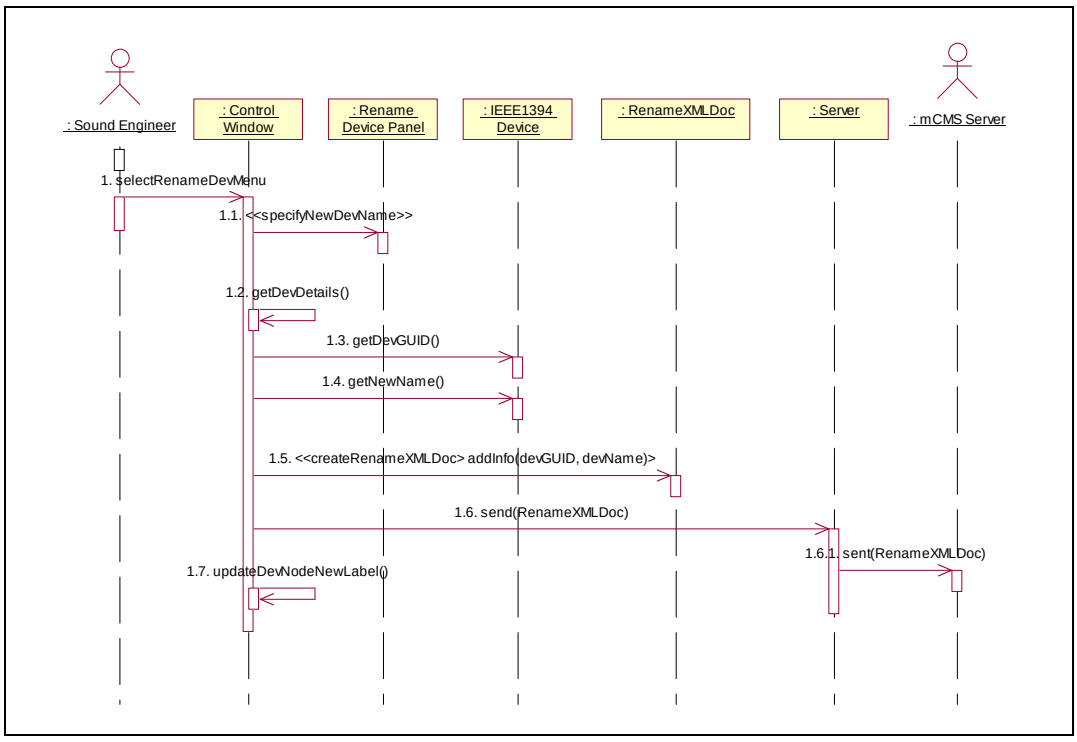

Figure 5.28: Renaming a device

5.3.2.9 "Managing Files" Use Case

The Broadcast patchbay allows the sound engineer to save routing settings into a text file as well as load saved routing settings into the patchbay. To do this, the sound engineer uses the "File" menu on which the "Save" menu item is used to save the

setting and the “Open” menu item is used to open saved settings. When the sound engineer selects the “Save” menu item:

- The patchbay gets the *Sources tree* and converts its node information into an XML object using the [*Sources tree*].toXML ( ) method.
- The *Sources tree* XML object is then saved into a text file within a specified location within the workstation.

To open saved routing settings in a text file within a specified location in the workstation, the sound engineer selected the “Open” menu item on the “File menu”. When the sound engineer selects the “Open” menu item:

- The patchbay opens the “Open” file dialog box, where the user navigates to a specific file location and selects the file to be opened and clicks the *Open* button of the “Open” file dialog box.
- The file data is loaded into the patchbay, and converted into an XML object that is parsed as XML.
- The patchbay loops through all source device nodes displayed on the *Sources tree* of the patchbay and compares them to their corresponding XML elements in the loaded XML data. If a device node exists in the XML data but not on the *Sources tree*, it is ignored, otherwise, the patchbay loops through each of the device plugs.
- For each device plug found on the *Sources tree*, its “connected” variable is compared to that of the same plug in the loaded XML data. If its value is “true”, then that plug has a live connection. This value is compared to that of the XML plug variable, if the plug XML “connected” variable is also “true” the plug is ignore, otherwise the live connection is disconnected and the green box icon cleared on the grid-matrix. If the *Sources tree* plug variable “connected” value is “false” and that of the same plug in the XML data is “true”, the patchbay gets the destination plug ID and it parent GUID of the connected plug. This information is used to locate the destination plug on the *Destinations tree*. If the plug’s parent device is on the network, its child plug nodes are searched for the plug with the ID from the XML plug data. If the

plug is found, the two plugs are connected, a green graphic box drawn on the grid-matrix at their cross-point. An XML "connection request" is created that contains the plug IDs, plug types and names of both plugs that is sent to the server to implement a connection request.

5.2 Broadcast Patchbay Usability Testing

This section discusses the usability testing phase of the Broadcast patchbay. Note that this usability testing was not meant to be an in-depth study as it was not the aim of this investigation but was done to ensure that all Broadcast patchbay requirements captured in the beginning of this chapter were adequately implemented.

The RUP software development process encourages consistent usability testing of the software by potential users, to ensure that the software performs the tasks it was designed to support satisfactorily. Various usability testing methods were employed during and after the development of the Broadcast patchbay. The international standard on usability testing, the ISO 9241-11 (1998), provided guidance during the testing of the Broadcast patchbay. It defines usability as the "extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use". Another international usability standard that was found to be useful is the ISO/IEC FDIS 9126-1 (2000), which defines usability as the "capability of the software product to be understood, learned, used, and be attractive to the user, when used under specified conditions". Therefore, it is clear that usability is a quality attribute that assesses how easily user interfaces can be used. According to Nielsen (2004), usability is defined by six quality factors:

- Fit for use (or functionality) – Determines if the developed system supports the tasks it was designed to support.
- Ease of learning – Determines how easily the system is to learn for new users.
- Task efficiency – Determines how quickly occasional users can perform their tasks on the system.
- Ease of remembering – Determines how easily users can re-establish proficiency with the system if they have not been using it for a long period.

- Subjective satisfaction – Determines how pleasant and satisfying the system's interface is for the user.
- Understandability - Determines how easy is it to understand what the system is doing, for example in situations of system failure.

At different stages in the development of the Broadcast patchbay, different usability testing techniques were utilised that generated a set of proposals for redesigning and modifying the patchbay in regard to these six usability factors.

5.2.1 Usability Testing User Profiles

Broadcast networks are complex in nature, and are usually controlled by well trained sound engineers who understand how audio is routed within the mLAN network. The Broadcast patchbay usability testing was carried out by users who fulfilled the following requirements:

- Had worked within Broadcast network studios for at least 3 years.
- Between the age of 18 and 60 years.
- Have experience working with grid-based patchbays.

Appendix C-C1 shows a detailed *User Profile Form* that was utilised for gathering usability testing users' information.

5.2.2 Usability Methodologies and Findings

Two different usability testing methods, namely the heuristic evaluation and the user testing, were selected to test the usability of the Broadcast patchbay. A preliminary heuristic evaluation was the first technique applied. It detected minor usability problems and missing functionality in the software that were fixed as much as possible before the user testing of the software by prospective users. Nielsen (2004) defines the heuristic evaluation process as a "usability engineering method for finding the usability problems in a user interface design so that they can be attended to as part of an iterative design process". The heuristic evaluation "System Checklist" used was developed by Deniese Pierotti from Xerox Corporation (1994) and is shown in *Appendix C-C3*. The usability principles (the "heuristics") evaluated by this heuristic

evaluation checklist include the following, which comply with Nielsen's (2005) ten usability heuristics:

- Visibility of System Status.
- Match Between the System and the Real World.
- User Control and Freedom.
- Consistency and Standards.
- Error Prevention.
- Recognition Rather than Recall.
- Flexibility and Minimalist Design.
- Aesthetic and Minimalist Design.
- Help Users Recognize, Diagnose, and Recover From Errors.
- Help and Documentation.

The Broadcast patchbay was updated to incorporate missing functionality before it was sent for usability testing. The usability questionnaire shown in *Appendix C-C2* was used for the Broadcast patchbay usability testing, and was derived from the Software Usability Measurement Inventory (SUMI) tool [Kirakowski, 1994], the Purdue Usability Testing Questionnaire (PUTQ) tool [Lin, Choong and Salvendy, 1997], and the Questionnaire for User Interface Satisfaction (QUIS) tool [Chin, Diehl and Norman, 1988]. It had 45 questions with each question grading the user's response on a Likert scale from 0 to 9, where 0 represented *strongly disagree* and 9 represented *strongly agree* [Siegle, 2004]. The questions utilized the following criteria that were relevant to the three patchbays developed:

a) Consistency

The consistency section of the usability questionnaire evaluated the consistency of the following aspects:

- The font size and type used in naming and labelling different panels of the patchbay.
- The wording used within the patchbay interface panels and that it is consistent with the user guidance provided.

- The grouping and ordering of menu options for different panels of the patchbay.

b) Learnability

This section tested how easy it was to learn to use the patchbay, and how much load (names and commands to remember when performing connection management tasks) the user was required to remember. This section also evaluated the following aspects:

- How straightforward was performing tasks on the patchbay.
- How steep was the learning curve for the user.
- How meaningful and useful were the patchbay commands.

c) Terminology, User Guidance and System Information

This section evaluated the usefulness of the system terminology and user guidance material provided. The questionnaire evaluated:

- If the terminology used in the guidance material was related to tasks.
- System feedback - How helpful are error messages.
- If the patchbay provided the CANCEL option.
- If error messages are disruptive or informative.

d) Screen Positioning

This section of the usability questionnaire evaluated the patchbay to see if:

- Reading information on the screen is easy.
- Positioning of messages on the screen captures the attention of the user.

e) Flexibility

This section evaluated how much control the patchbay gave the user. Specifically, the questions for this section evaluated the patchbay with respect to:

- The customisability of its panel components.
- Its zooming capabilities for clarity of the displayed interface components.

- Its menu options dependability in the context of the panel and their purposes.
- Its zooming capabilities for display expansion.

f) Minimal Action

This section evaluated the patchbay to see if it provided default values, function keys and options for accessing frequently used features.

g) Perceptual Limitation

This section of the usability questionnaire evaluated the patchbay to see if:

- It uses blending colours for different panels.
- It demarcates groups of information.
- Active windows and components are highlighted.
- Its screen density is reasonable.

h) System Capabilities

This section evaluated the efficiency and reliability of the patchbay when the user performs connection management tasks with respect to speed, reliability, security and error handling.

The Broadcast patchbay users were two sound engineers, one from Germany and the other from Canada, who fulfilled test user requirements stated in *section 5.4.1*. The developer could not get local usability test users who fulfilled the requirements in *section 5.4.1*. As a result, the developer could not have face-to-face usability testing sessions with the users, and instead users were asked to perform common connection management tasks on the patchbay according to their discretion and complete the usability testing questionnaire, which was sent them through email.

5.2.2.1 Broadcast Patchbay Usability Testing Results

This section discusses results of the Broadcast patchbay usability testing process. *Figure 5.29* shows the Broadcast patchbay version 1 interface that was sent to usability users for the first testing iteration.

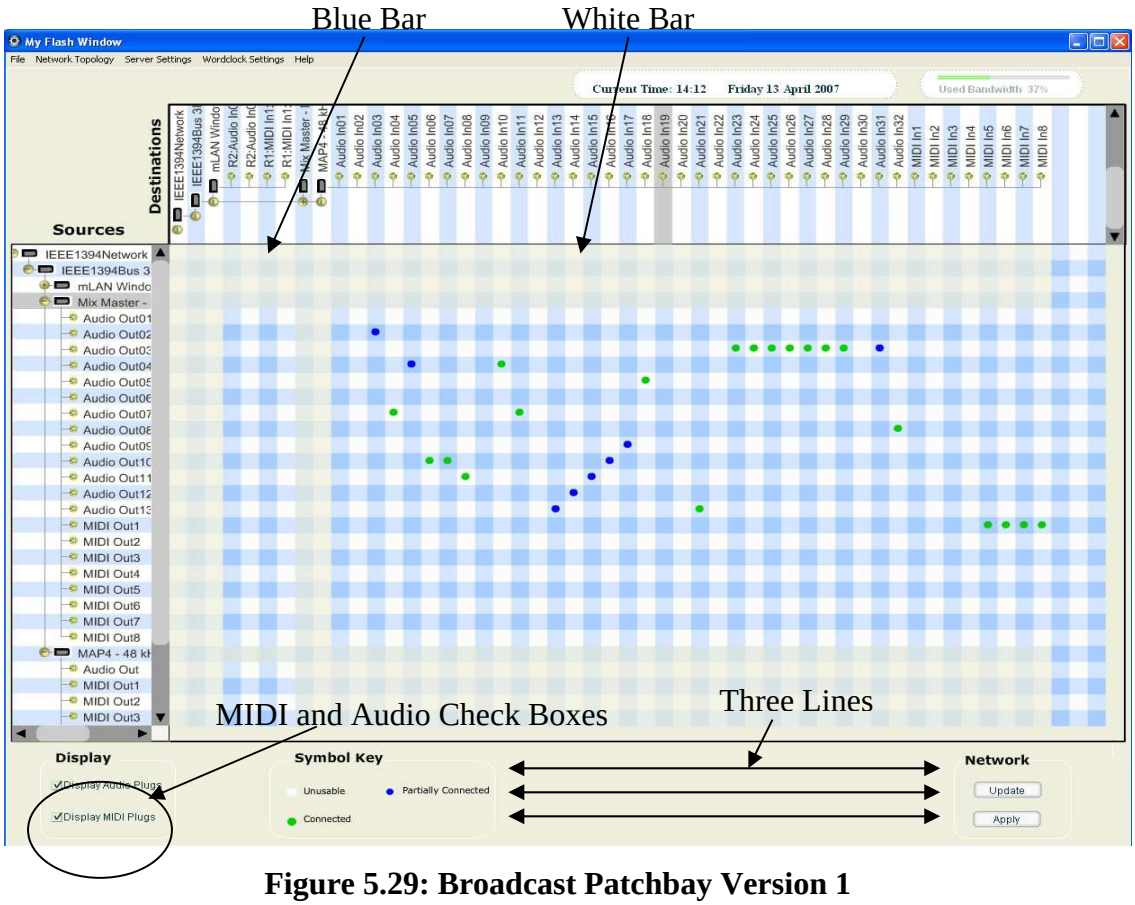

Figure 5.29: Broadcast Patchbay Version 1

The following sections present usability testing results for each of the eight aspects described in the preceding section.

a) Consistency

One of the users identified consistency issues, that the *Wordclock* panel title font type and size did not match that of the main *Control Window* and other panels. Both users *strongly agreed* that the labels were located at consistent location on all screens of the Broadcast patchbay interface panels. Both users *strongly agreed* that the grouping and ordering of menu options for different panels of the Broadcast patchbay was logical.

b) Learnability

This section evaluated how easy the patchbay was to learn to use. The following results were noted:

- Both users *strongly agreed* that learning to use the Broadcast patchbay was easy.
- Both users *strongly agreed* that remembering names and use of commands within the Broadcast patchbay was not challenging.
- Both users *strongly agreed* that performing tasks was straightforward.
- Both users *strongly disagreed* that the Broadcast patchbay required a steep learning curve.

c) Terminology, User Guidance and System Information

This section of the usability questionnaire evaluated the terminology and error messages used by the system. Both users *strongly agreed* that the Broadcast patchbay terminology used was related to tasks and the error messages were not disruptive but informative. However, one of the users *strongly disagreed* that the Broadcast patchbay provided the CANCEL option. The system did not provide the user with the option of rolling back, for instance, when one makes a mistake while performing a connection management task.

d) Screen Positioning

This section looked at the positioning of different graphic components of the Broadcast patchbay interface. Both users *agreed* that the Broadcast patchbay's organization of information was logical and standard. However, one the users also thought the "Date and Time" information at the top-right corner was redundant information, which the workstation (running the patchbay) displays by default. Therefore, this information wasted space. The arrangement of the legend elements in three lines also wasted valuable space [*Figure 5.29*]. Both users *strongly agreed* that error messages dialog boxes were positioned at convenient locations to attract the user's attention.

e) Flexibility

This section evaluated how much control the patchbay gave the user:

- Both users *agreed* that the Broadcast patchbay provided the user with direct manipulation capabilities such as being able to directly manipulate the application grid-matrix box when establishing and breaking audio connections.
- Both users *disagreed* that the Broadcast patchbay allowed the users to display elements according to their needs, for instance, the Broadcast patchbay did not provide the user ways of customising tree list nodes or of changing its colours. The system provide only one way of doing thing, the user was allowed to customise aspects of its components.

f) Minimal Action

Both users *strongly disagreed* that the Broadcast patchbay provided default values and function keys for frequent control entries. The system did not provide shortcut keys and icons for fast access to frequently used panels and features.

g) Perceptual Limitation

This section evaluated the aesthetic aspects of the Broadcast patchbay interface components. Both users had problems with the "table cloth" effect introduced by the alternating *Blue* and *White* bars [*Figure 5.22*], which they thought were hard on the eyes and gave the interface a sluggish feel. In addition, the small green and blue graphic circles on the grid-matrix made it hard for the user to see and work easily with the cross-points in a case where there were many connections displayed on the grid. However, both users *strongly agreed* that the system screen density, that is the spacing between patchbay interface components, was reasonable.

h) System Capabilities (Speed and Reliability)

This section evaluated the efficiency and reliability of the Broadcast patchbay when the user performs connection management tasks with respect to speed, reliability, security and error handling. Both users *strongly agreed* that tasks such as making and breaking audio connections, which were performed on the *Control Window* grid-matrix executed efficiently. One of the users *disagreed* that the system reliability was good enough. Reliability issues were found on the *Wordclock* panel that include:

- On the *Wordclock* panel, if one changed the sample rate of a master device that has slave device nodes, the master device sample rate label changed but the sample rate label of the slave device nodes remained the unchanged.
- On the *Wordclock* panel, one could make a word clock Master/Slave configuration between two devices with different sample rates.

5.2.2.2 Redesigning of the Broadcast Patchbay

The usability testing process revealed problems with the patchbay, notably the improper functioning of the *Wordclock* panel, the sluggishness of the grid-matrix, the wasted space on the main *Control Window*, and consistency issues on the *Wordclock* panel. Only the critical features were incorporated into the second version of the Broadcast patchbay. *Figure 5.30* shows the redesigned and final Broadcast patchbay. Alternating *Blue* and *White* bars on the grid-matrix were replaced by a simple grid with a grey background, effectively removing the "table cloth" effect. Live connections (previously denoted by small green and blue graphic circles [*Figure 5.29*].) are represented by much bigger green square boxes that fill the whole grid box in the new Broadcast patchbay version 2. The "Date and Time" labels at the top-right of *Figure 5.29* were removed and the patchbay legend moved and rearranged at the bottom of the patchbay to save space. The rearrangements left more space for displaying the "grid-matrix" and the "device trees". As shown in *Figure 5.30*, tree list components were resized to that node names are clearly displayed without any need for scrolling, unlike the case for the Broadcast patchbay version 1 [*Figure 5.29*].

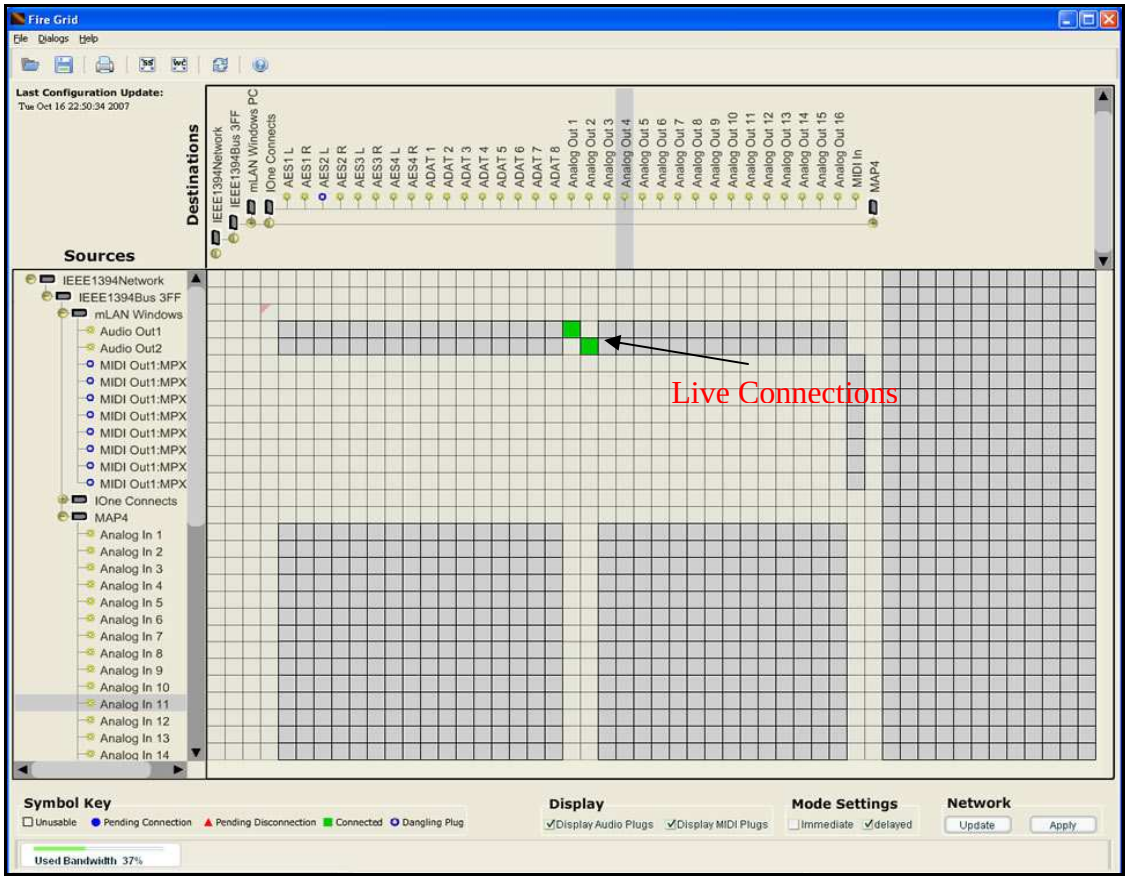

Figure 5.30: Broadcast Patchbay Version 2

Standard tool bar icons and shortcut keys were added to the Broadcast patchbay for frequently performed connection management tasks to enhance the speed by which the user executed tasks. In addition, roll back options were added for all connection management tasks performed on the Broadcast patchbay. Notably, the CANCEL button was included in all the patchbay panels and the user had the option of reversing the processes of establishing and breaking audio connections by clicking twice the grid-matrix box which the connection status is being reversed. For instance, assuming the sound engineer clicks the cross-point grid-matrix box of two plug nodes that are connected by mistake. To reverse the connection if the patchbay is in "Delayed Mode", the sound engineer clicks again the same cross-point grid-matrix box of the two newly connected plugs. Their connection should be broken.

5.2.2.3 Broadcast Patchbay Version 2 Test feedback

The Broadcast patchbay version 2 was further evaluated by the two prospective users. Feedback showed that both testers were satisfied by the modifications made. They found the interface, especially the grid-matrix, to be easy on the eyes and much

clearer, because of the inclusion of connection and disconnection square graphic boxes that fill the whole grid box in contrast to the small circles used in the version 1 of the patchbay. The rearrangement of the legend at the bottom of the patchbay created more space for displaying the “device trees” clearly, and allowed for the addition of four more rows and two columns on the grid-matrix.

5.3 Chapter Summary

Standard Human Computer Interaction (HCI) and Object Oriented Programming (OOP) concepts were utilised in conjunction with the Rational Unified Process (RUP) Iterative and Incremental Process for the development of the Broadcast patchbay for Broadcast networks. The Broadcast patchbay developed comprises three main panels, the *Control Window*, the *Settings* panel and the *Wordclock* panel. The *Control Window* is used for audio routing and modifying device properties such as the device nickname and device Plug Layout configurations. The *Settings* panel enables the user to specify the mCMS server DNS name, port number, and the number of source PC plugs for the host workstation. The *Wordclock* panel enables the sound engineer to set/clear word clock Master/Slave configurations which allow mLAN devices to transmit and receive packets at synchronised sample rates. It is a requirement in mLAN networks that any two communicating devices do so at the same sample rate. A heuristic evaluation was performed by the developer before the patchbay was sent to prospective testers, who performed the actual usability test based on a 45-question questionnaire shown in *Appendix C-C2*. Their findings resulted in the redesigning of the Broadcast patchbay to provide for maximum space usage and ensured the proper functioning of the *Wordclock* panel for setting/clearing word clock Master/Slave configurations within Broadcast networks.

In the next chapter, the Rational Unified Process’ Iterative and Incremental Process for software development is applied to the development of a patchbay for Project studio networks.

CHAPTER 6

6 Project Studio Patchbay Design and Development

Project studio networks are smaller than Broadcast networks, and typically deal with fewer device connections. They are generally operated by music producers who understand how the audio network routes audio from one output plug on a source device to a particular input plug on a destination device. Due to the small size of Project studio networks, sound engineers for these networks have the tendency to switch physical audio cables between the actual physical devices to perform audio routing. This has led to them developing mental models of their studio device topologies, for instance, they know where each device physically lies within the studio without necessarily looking at it.

As a result, graphic-based patchbays are normally used in these networks, because they use graphic representations (that are customisable) of the network devices, and therefore present sound engineers with a familiar working environment that matches their mental model of the real studio. A well designed graphic patchbay can allow sound engineers to customise *device blocks*[9] to match the layout of their real studio, and use *connector lines*[10] between *device blocks* to create the cable-switching capabilities as they would have in a real studio. The development of the Project studio patchbay followed the eight phases of the Iterative and Incremental Process (RUP), which includes the requirements phase, the analysis and design phase, the implementation phase, the deployment phase, and the testing phase [C*hapter 5: Figure 5.1*]. This chapter discusses in detail how these development stages guided the development of a Project studio patchbay for use in Project studio networks.

6.1 Project Studio Patchbay Requirements Analysis

Although graphic-based patchbays are commonly deployed within Project studio networks, it was not clear which patchbay layout and what functionality should be incorporated into the patchbay to be developed for Project studio networks. As a

[9] *Device blocks* are graphic boxes that represent network devices on the patchbay display [*Figure 6.1*].

[10] *Connector lines* are connection graphic lines that join two graphic plugs on the patchbay display [*Figure 6.1*].

result, a computer prototype of a typical graphic-based patchbay was developed for purposes of gathering Project studio patchbay requirements, and was also used for determining the best layout that would satisfy Project studio users. Nielsen (2004) defines a prototype as a "working model of the final system built to develop and test design ideas". A computer prototype was chosen for gathering requirements for the Project studio patchbay for the following reasons:

- It allows for interactive design of the software before its development. In addition, a computer prototype provides interactive feedback for the user, which enables users to visualise how the final system will behave.
- It is high-fidelity in appearance and interaction as it allows the user to explore the graphic design of the final system. It also allows for the choosing of appropriate colours, fonts, component alignments, icons, and the white space.
- It has low-fidelity in depth since it comprises most of the system's components with no backend.

The computer prototype was tested by eight typical Project studio users who fulfilled the following requirements:

- Have experience working within Project studio networks for at least 3 years.
- Are between the ages of 18 - 60 years.
- Have experience working with graphic-based patchbays.

Appendix C-C1 shows a detailed *User Profile Form* that was utilised for gathering the usability testing user information.

6.2 Project Studio Computer Prototype

Figure 6.1 shows the computer prototype that was sent to eight users within South Africa and abroad (Germany and Canada) for testing. This prototype included the following features and capabilities to:

- Establish audio connections.
- Break audio connections.

- Drag *device blocks*.
- Resize *device blocks*.
- Add a new *device block* using the *Add Device* button.
- Expand and collapse *plug blocks* to view clearly the names of individual plugs.

The prototype interface consisted of dummy graphic components, such as the bandwidth bar, the *Display* buttons, *the Mode Settings* buttons, and the *Network* buttons, that were incorporated into the final patchbay design.

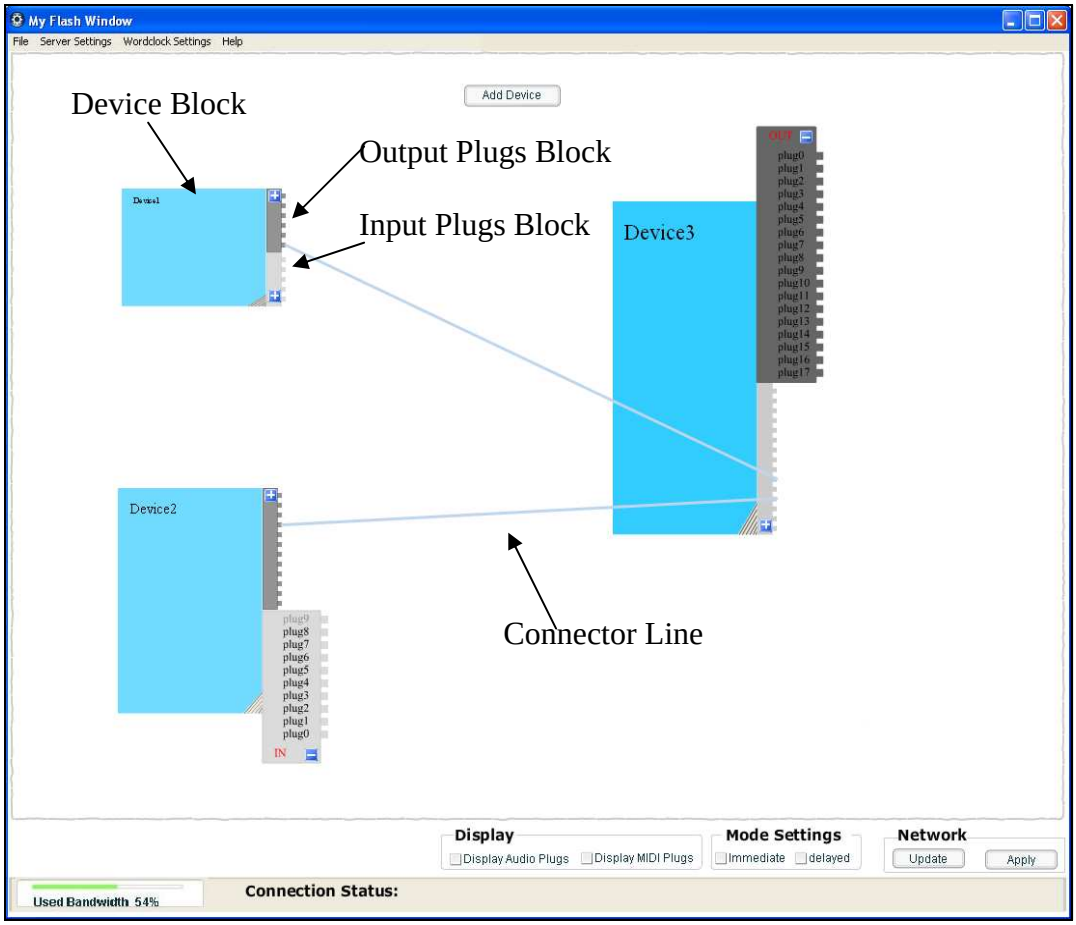

Figure 6.1: Project Studio Patchbay Prototype

A shorter version of the usability questionnaire shown in *Appendix C-C2*, with 12 questions, was completed by each user who tested the computer prototype. Sections of the usability questionnaire [*Appendix C-C2*], which were not relevant to the testing of the computer prototype but to the final system, were removed. This therefore, reduced the size of the questionnaire significantly. The questions removed evaluated the:

- Usefulness of error massages.
- System speed and its reliability capabilities.
- Panel components consistency (title font size and type).

These sections were only relevant if the patchbay had all its panels and the backend mCMS server was active.

The questions used evaluated:

- How easy the system was to use for a new user, and how much mental load the user had to carry to use the system.
- The flexibility of the system features.
- The size, colour and shape of the interface components (for instance the *device blocks* and button components).
- The arrangement and layout of the patchbay interface components.

The users were given a section to write any suggestions and comments they deemed important for the prototype design. The analysis of the gathered patchbay requirements resulted in the developer listing of important patchbay features. Stimulus/Response sequences were used to describe how these features were to work in the complete product. The prototype did not have a live connection to the mCMS server. *Appendix A-A3* provides a complete list of the system requirements for the Project studio patchbay that were gathered from the testers' feedback, which was collected using a usability questionnaire. The IEEE Recommended Practice for Software Requirements Specifications [IEEE Inc, 1998] was utilised for documenting these requirements.

6.3 Project Studio Patchbay Description

The Project studio patchbay developed was named "FireSwitch". It comprises four main panels, namely the *Control Window*, the *Settings* panel, the *Wordclock* panel, and the *Device Information* panel.

The *Control Window* [*Figure 6.2*] is the main panel of the Project studio patchbay, and incorporates important features that facilitate audio routing and connection management in mLAN Project studio networks. *Figure 6.2* shows a screenshot of the Project studio patchbay that displays a typical Project studio network with four mLAN compatible devices, namely the *mLAN Windows PC* device, two I/One breakout boxes (the *OGT - I/One Src* device and the *OGT - I/One Dest* device), and Yamaha Corporation's *MAP4* device. The *Control Window* displays mLAN devices on the network as square graphic boxes (referred to as *device blocks*) in different colours, which are assigned randomly when the *device blocks* are first created. Each *device block* displays the name of the device it represents, the word clock state of the device (either master or slave) and the device sample rate. It also has a button labelled "Information" that is used to display detailed information about a device.

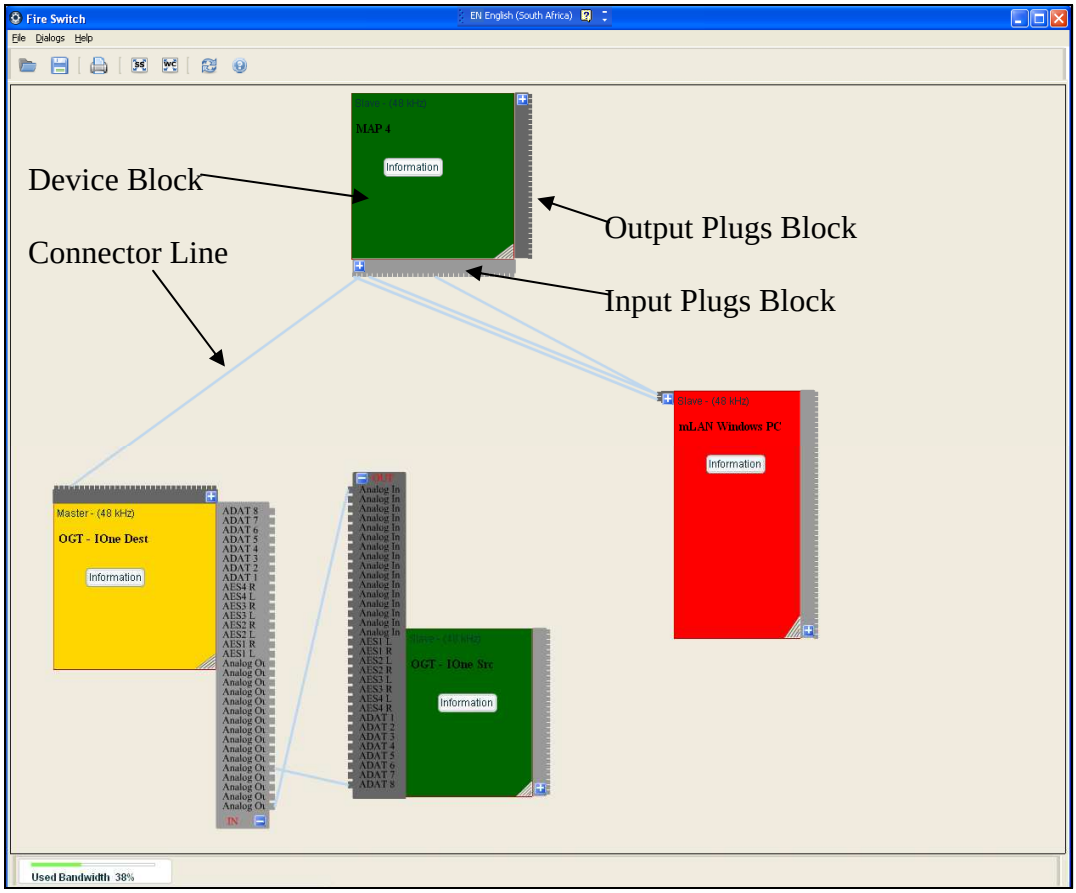

Figure 6.2: Project Studio Control Window

Each *device block* has two expandable/collapsible *plug blocks* attached to it. The *plug blocks* (the *input plugs block* and the *output plugs block*) display input plug graphics and output plug graphics for each device. *Figure 6.2* also shows the *connector lines* that represent live connections between device plugs. Connection management tasks that can be performed on the *Control Window* of the Project studio patchbay include:

- Establishing audio connections.
- Breaking audio connections.
- Renaming device nicknames.
- Clearing all device connections for a particular device.
- Viewing detailed information for a particular device on the mLAN network.
- Saving / Opening routing settings into / from a text file.
- Identifying a particular device on the network.

The Project studio patchbay uses the same *Settings* panel and the *Wordclock* panel as the Broadcast patchbay described in c*hapter 5* [*section 5.2*]. The *Settings* panel allows the sound engineer to:

- Configure the mCMS server name and port number.
- Configure the number of source plugs (for both MIDI and audio) of the workstation running the Project studio patchbay.

The *Wordclock* panel allows the sound engineer to:

- Set a global master unit for the whole mLAN network.
- Enslave a device to a particular master device.
- Release all slave devices for a particular master device.
- Remove a single slave device from its master device.
- Change a master device word clock sample rate and its word clock source as well as its work clock source output.

The *Device Information* panel [*Figure 6.3*] displays detailed information about a particular device. This panel is opened by clicking the *Information* button at the centre of each *device block* displayed on the patchbay interface. Device information that can be viewed includes [*Figure 6.3*]:

- The device vendor.
- The device model name.
- The device GUID.
- The device supported sample rates.
- The device current device Plug Layout.

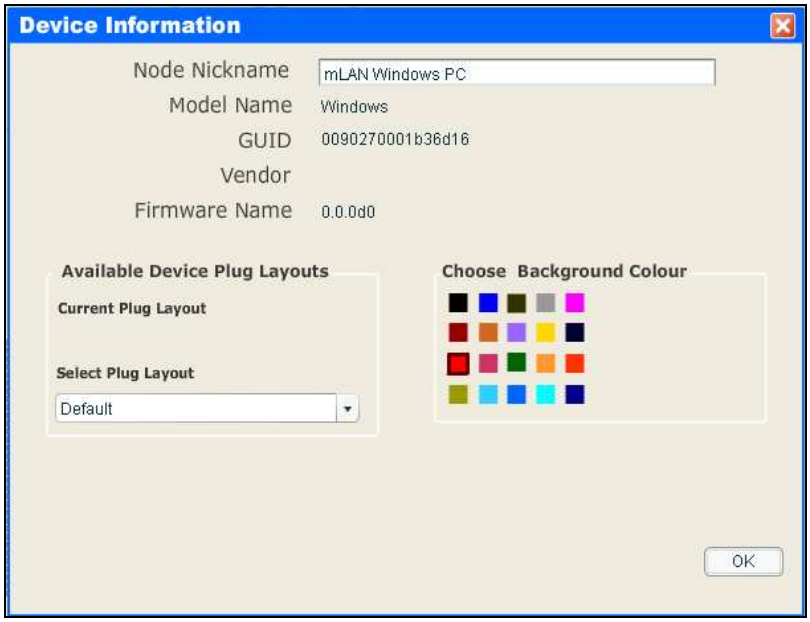

Figure 6.3: Device Information Panel

Besides viewing device information, the *Device Information* panel allows the sound engineer to:

- Rename device nicknames.
- Configure the device Plug Layout.
- Change the device background colour.

To rename a device on the *Device Information* panel, the sound engineer specifies the new name of the device in the "Node Nickname" text box [*Figure 6.3*] and clicks the

OK button. An XML "rename device" document [*Appendix B: Listing 11*] is created, and is sent to the mCMS server containing the GUID of the device to be renamed and its new name. If the renaming process succeeds, the label of the device on the patchbay interface is automatically changed to the new name specified on the *Device Information* panel.

To configure a device Plug Layout, the sound engineer selects the desired Plug Layout to be applied on the "Select Plug Layout" combo box [*Figure 6.3*] of the *Device Information* panel and clicks the *OK* button. An XML "Plug Layout" document [*Listing 6.1*] is created, and is sent to the mCMS server to request the implementation of the setting.

```
<?xml version="1.0" encoding="utf-16" ?>
- <mLANServerCommand version="1.0">
  - <object name="patch">
    - <method name="setCurrentPlugLayout">
        <parameter name="GUID" value="0013f00400000014" />
        <parameter name="plugLayoutID" value="1" />
      </method>
    </object>
  </mLANServerCommand>
```

Listing 6.1: XML "set current plug layout" Request Document

If the setting is successfully implemented, the mCMS server sends an XML "configuration" document to the patchbay that it uses to update the display. The number of device plugs are then reduced or increased depending on the Plug Layout chosen.

The sound engineer can change the device's background colour by simply selecting the proper colour box on the "Choose Background Colour" grid [*Figure 6.3*] of the *Device Information* panel and clicking the *OK* button. The selected colour is automatically applied to the *device block*. There is no client/server communication for changing the *device block* colour.

The sound engineer can also drag device *plug blocks* to any of the eight sides of the *device block* they are attached to by simply clicking and dragging the *plug block* to positions highlighted [*Figure 6.4*]. The *device blocks* can also be dragged to any

position within the patchbay display space. There is no client/server communication for this process.

If a *device block* is too small such that its labels and word clock information cannot be seen clearly, the sound engineer can zoom in and out as required by clicking and dragging the *Zoom* button at the bottom right corner of the *device block* to enlarge it.

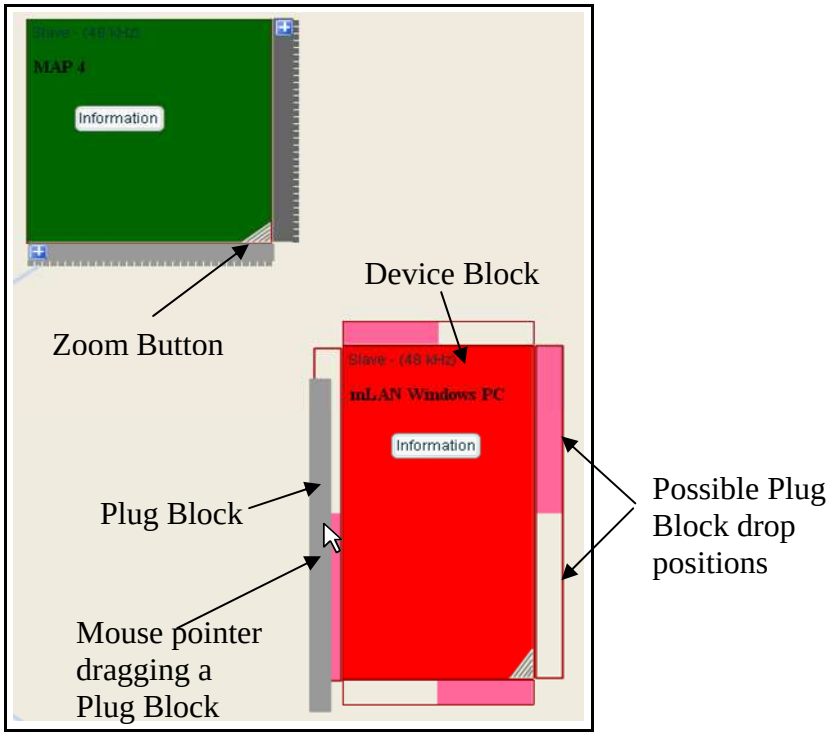

Figure 6.4: Moving Plug Blocks

6.4 Project Studio Patchbay Design and Implementation

This section discusses the design and implementation of the Project studio patchbay using UML sequence diagrams.

6.4.1 Project Studio Patchbay Implementation Sequence Diagrams

The Project studio patchbay uses the same Use Case Diagram as the Broadcast patchbay described in *chapter 5* [*section 5.3.1.1*], which shows the high-level functions of the Project studio patchbay and identifies two entities (Actors) that interact with the patchbay, namely the sound engineer and the mCMS Server. The high-level functions of the Project studio patchbay include:

- Managing files.

- Connecting to the mCMS Server.
- Establishing audio Connections.
- Breaking audio connections.
- Updating the Project studio patchbay.
- Applying changes made on the Project studio patchbay.
- Setting/Clearing word clock Master/Slave Configurations.
- Identifying a particular network Device.

Sections 6.3.1.1 through to *6.3.1.5* describe the implementation of some of these features in detail.

Figure 6.5 shows the Object Model that depicts the relationships and dynamic interactions between objects written for the Project studio patchbay. The *IEEE1394 Network* object is shown to contain an aggregation of *IEEE1394 Bus* objects. Each *IEEE1394 Bus* object contains an aggregation of *IEEE1394 Device* objects, which in turn each contains an aggregation of *Device Audio Plug* objects that can either represent an output or input mLAN plug. The model also shows the interrelationships between the various Project studio patchbay panels (the *Control Window*, the *Wordclock* panel, the *Settings* panel, the *Login* panel, and the *Device Information* panel) and their associated XML document objects. XML document objects are created for each XML document sent to the mCMS server when connection management tasks are performed.

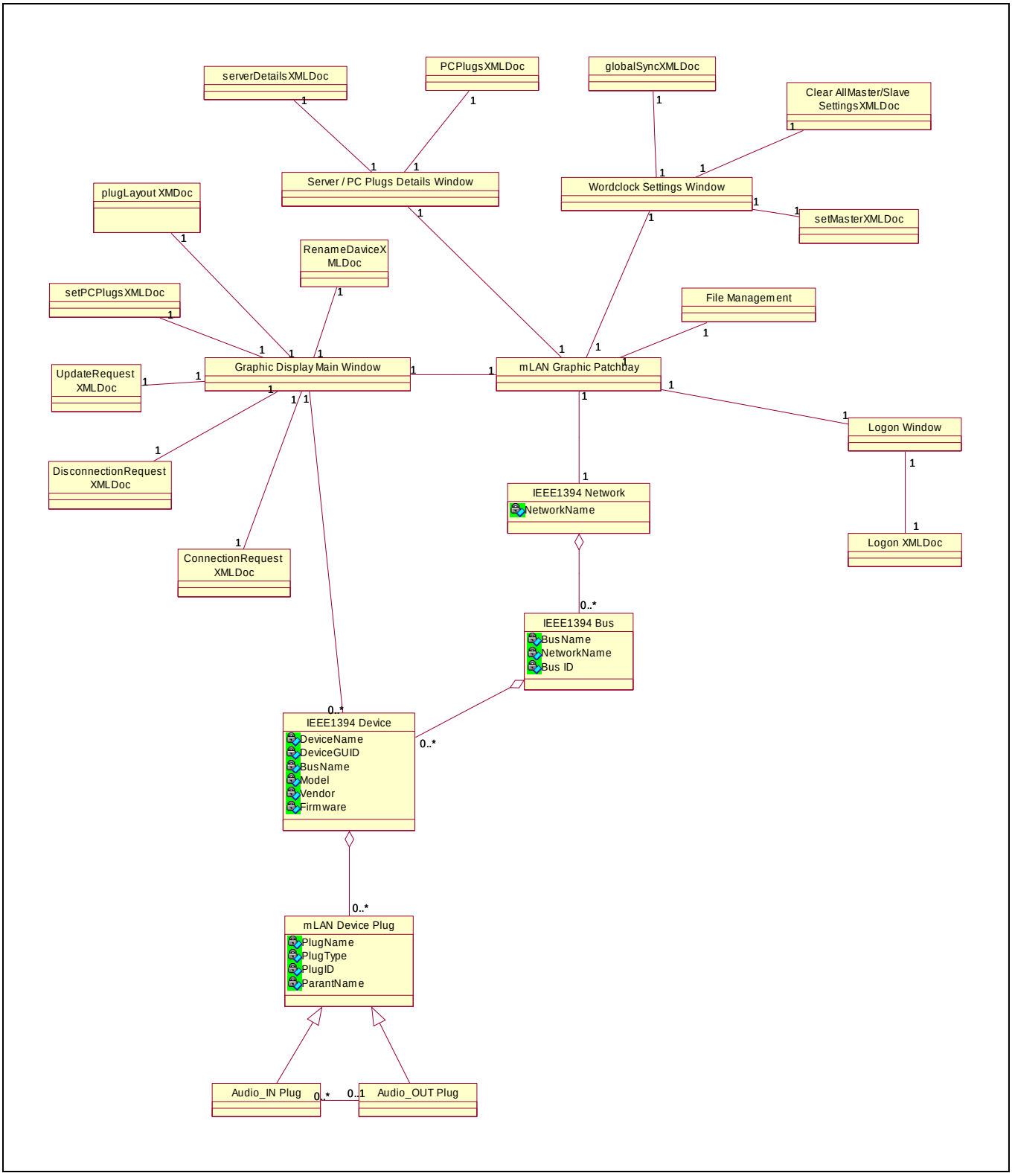

Figure 6.5: Project Studio Patchbay Object Model

6.4.1.1 "Connect to mCMS server" Use Case

At start-up, a TCP/IP connection is established between the Project studio patchbay and the mCMS server. The user logs into the mCMS server by specifying the

username and the password in the *Login* panel. When the patchbay successfully connects and authenticates with the mCMS server, the mCMS server sends an XML "configuration" document [*Listing 5.2*] to the patchbay that contains information of the current status of the network devices and their connections. On the patchbay, the XML elements of the XML "configuration" document are extracted, the network hierarchy recreated and the patchbay updated with the latest network information received as follows [*Figure 6.6*]:

- It loops through the XML "configuration" document *IEEE1394 Network* element and creates the *IEEE1394 Network* object.
- It loops through all *IEEE1394 bus* elements, and creates *IEEE1394 bus* objects for each XML bus element found.
- For each *IEEE1394 bus* element, it loops through all *IEEE1394 device* elements, and creates *IEEE1394 device* objects. For each *device* object, a *device block* graphic is created on the patchbay, which is assigned a random background colour. Its size is determined by the number of input and output plugs the device has.
- For each *IEEE1394 device* element, it loops through all its *mLAN plug* elements, and creates *mLAN plug* objects. It creates *plug blocks* for each device corresponding to each plug object, and attaches them to their *device blocks*.
- For each *connection* element, it loops through all its *patch* elements, and draws *connector lines* on the patchbay interface for device plugs with live connections.

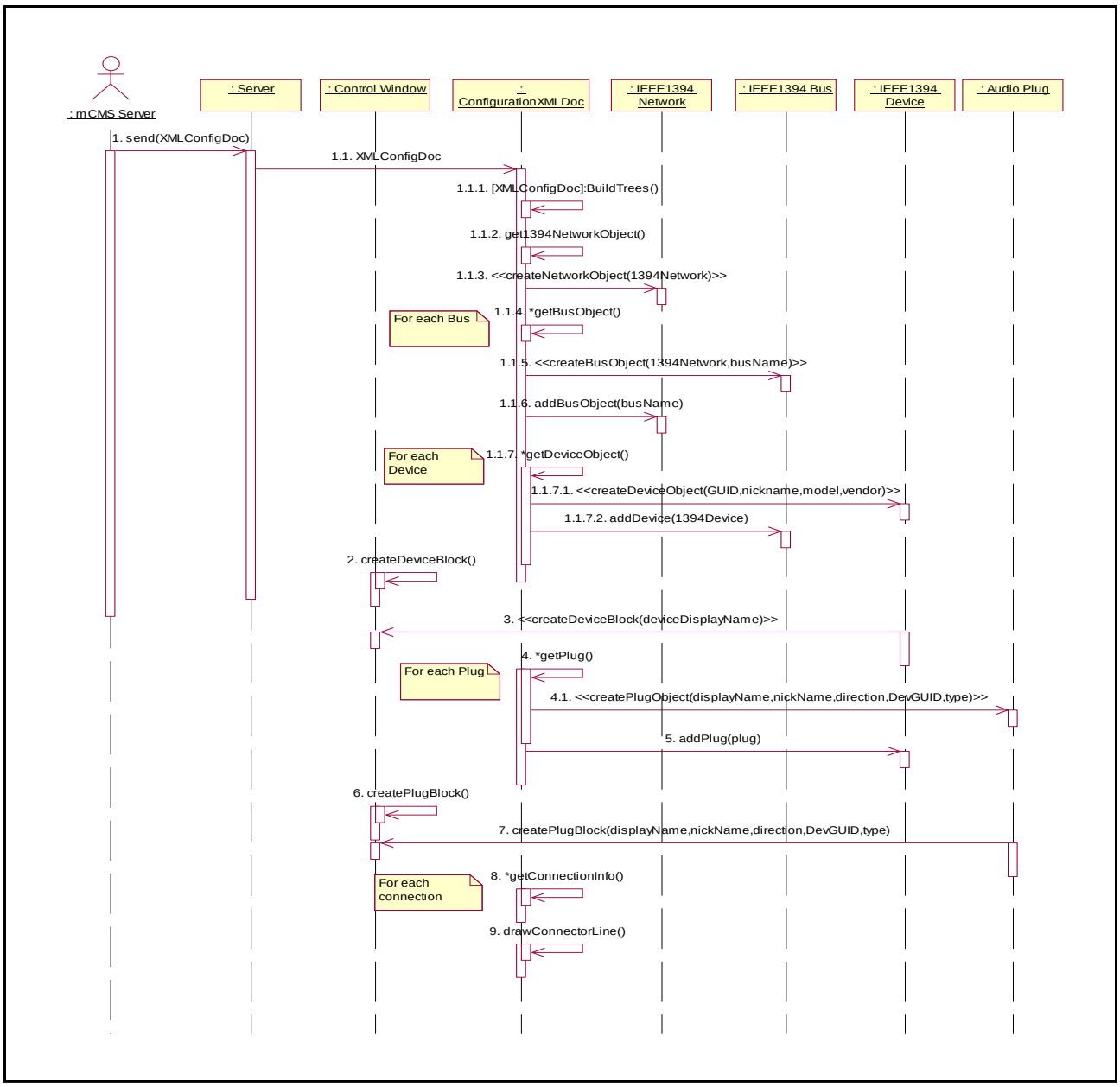

Figure 6.6: Receiving the XML Configuration Document Sequence Diagram

6.4.1.2 "Establishing Audio Connections" Use Case

The Project studio patchbay allows the sound engineer to establish audio connections between mLAN plugs of the same type and on different devices. To establish an audio connection, the sound engineer clicks the *maximise* icon to expand the input and output *plug blocks* of both devices to be connected [*Figure 6.7*]. This allows the sound engineer to clearly view plug names of each *plug block*.

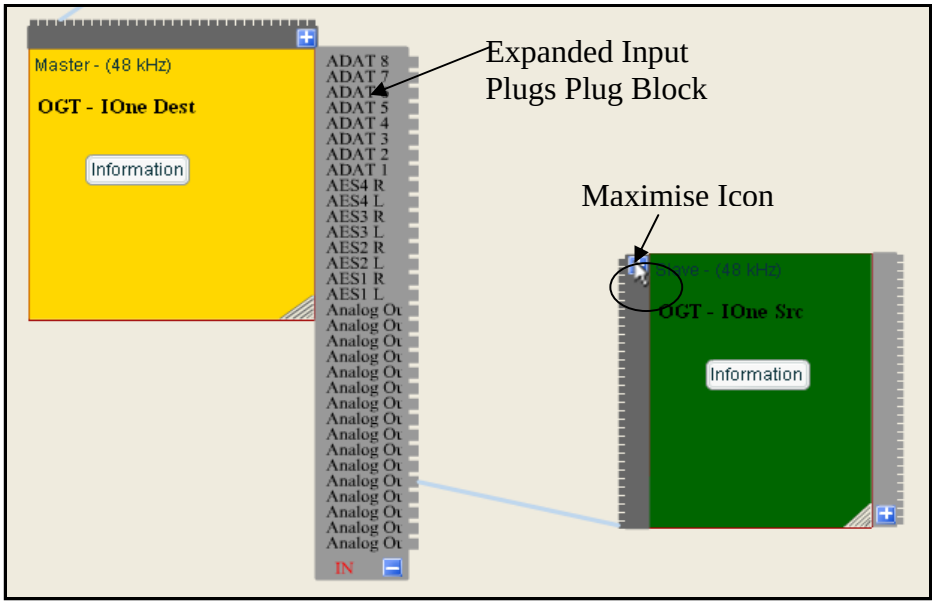

Figure 6.7: Maximising Inputs and Outputs Plug Blocks

With both *plug blocks* expanded, the sound engineer clicks either the output plug or the input plug graphic of the plug to be connected and drags the mouse pointer out of the plug graphic towards the other plug to connect to. A *connector line* is drawn that represents the new connection. In *Figure 6.8*, an audio connection is established from the output plug *Analog In 3* on the device *OGT-IOne Src* to the input plug *ADAT 3* on the device OGT-*IOne Dest*. When the sound engineer releases the mouse over the input plug *ADAT 3*, an XML "connection request" document [*Appendix B: Listing 5*] is created containing information of both plugs, and sent to the mCMS server. If the network has enough bandwidth, the connection is implemented by the Enabler module and a connector line is permanently drawn between the two newly connected plugs and audio is routed between the two plugs. When the user has expanded the *plug blocks* of the two devices to be connected and clicks and drags out the mouse pointer from one of the plug graphics:

- The patchbay creates three variables to hold the ID, the plug type and name of the plug the user is dragging from and draws a *connector line* to the mouse pointer that reflects the new connection.
- As the mouse pointer is moved to the next plug, the *connector line* is continuously redrawn.

- When the user releases the mouse over another plug graphic (that is not on the same device as the first plug), the patchbay creates three other variables to hold the *connector line* the ID, the plug type and name of the plug of the new plug.
- An XML "connection request" document is created that contains the information (the IDs, the plug types and names of the two plugs) and sent to the mCMS server.
- If the request is successfully implemented, a permanent *connector line* is drawn between the newly connected plugs and audio is routed between them.

Output plugs can have many connections, whilst input plugs can only have a single connection at a time. This is because output plugs can transmit audio to many input plugs, while input plugs can only receive input from only one output plug.

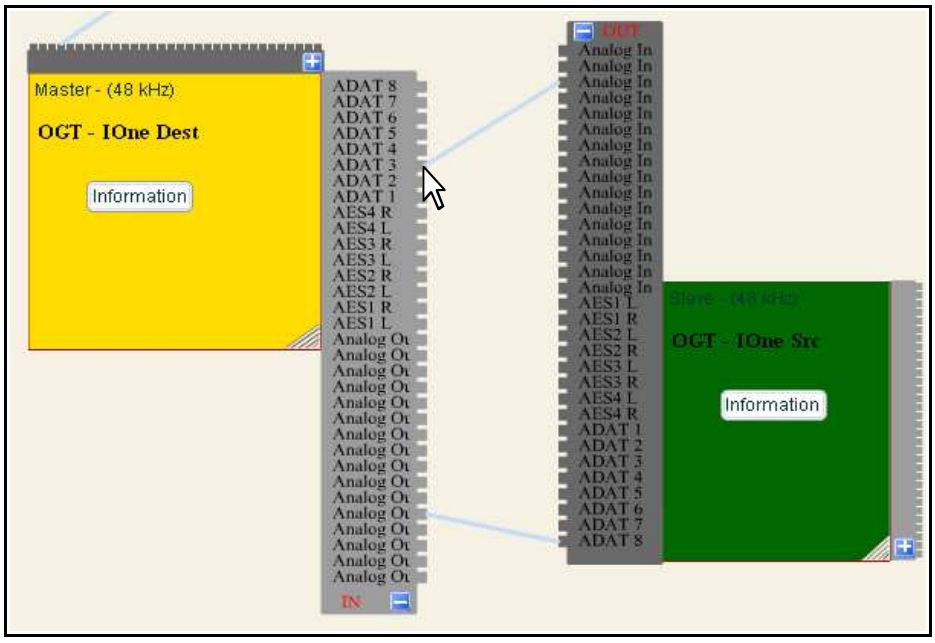

Figure 6.8: Establishing Audio Connections

6.4.1.3 "Breaking Audio Connections" Use Case

The Project studio patchbay also allows the sound engineer to break audio connections. To do this, the sound engineer clicks the *connector line* joining the two plugs to be disconnected and drags it [*Figure 6.9*]. The patchbay temporarily clears the dragged *connector line*. An XML "disconnection request" document [*Appendix B:*

Listing 6] is created that contains information of the input plug to be disconnected, and sent to the mCMS server which invokes the Enabler module to implement the request. If the request is successfully implemented, the cleared *connector line* is deleted permanently (in which case, audio stops streaming between the two plugs), otherwise if the disconnection failed the *connector line* is redrawn between the two plugs and audio continues to stream. When the user drags the *connector line* joining two plugs:

- The patchbay gets the ID, the plug type and name of the destination plug to be disconnected and temporarily deletes the *connector line*.
- An XML "disconnection request" document is created that contained the destination plug information (the ID, the plug type and name) and sent to the mCMS server.
- If the request is successfully implemented, the *connector line* remains deleted permanently otherwise it is redrawn and an error message from the server displayed.

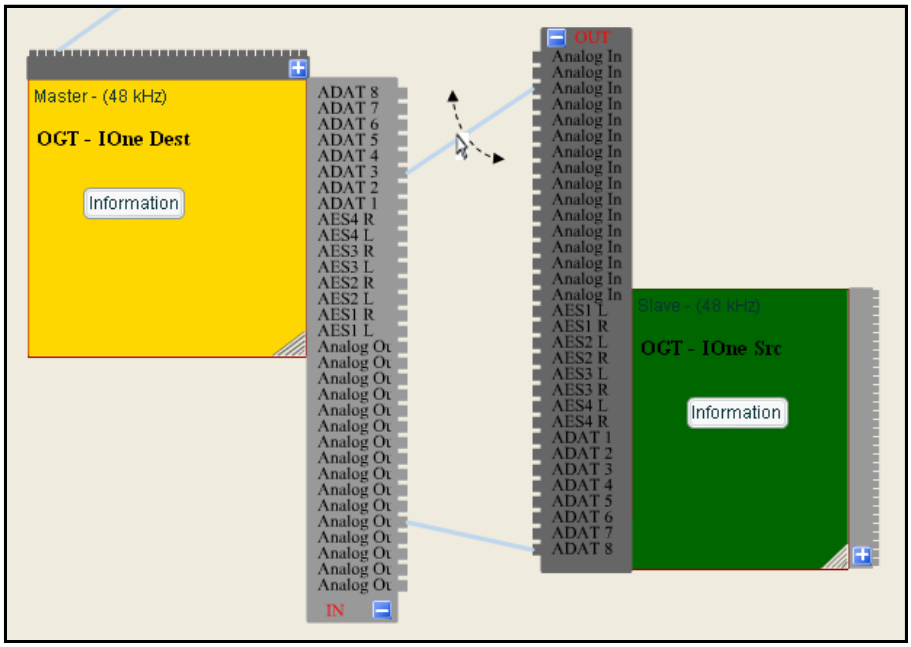

Figure 6.9: Breaking Audio Connections

6.4.1.4 "Setting/Clearing Wordclock Master/Slave Configurations" Use Case

The Project studio patchbay uses the same method for setting and clearing word clock Master/Slave configurations as the Broadcast patchbay described in *section 5.3.1.4*.

6.4.1.5 "Identify Device" Use Case

The sound engineer can identify a particular device on the Project studio patchbay by right-clicking the *device block* and selecting the "Identify Device" submenu item [*Figure 6.10*]. The patchbay automatically creates an XML "identify device" document [*Appendix B: Figure 12*] that is sent to the mCMS server. This XML request document contains the GUID of the device to be identified. If the request is implemented successfully, one or more device LEDs flash. When the user selects the "Identify device" menu item:

- The patchbay gets the GUID of the selected device and incorporates it into an XML "identify device" request document that is sent to the server.
- If the request is successful, one more device LEDs will flash otherwise an error message is displayed.

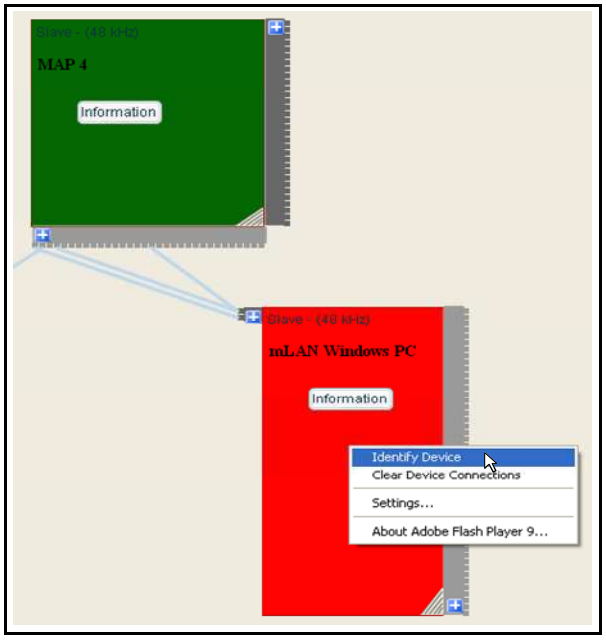

Figure 6.10: Identifying a Device

6.4.1.6 “Managing Files” Use Case

The Project studio patchbay allows the sound engineer to save routing settings into a text file as well as load saved routing settings into the patchbay. The sound engineer does this by selecting the “Save” menu item to save the routing settings into a text file and the “Open” menu item to open saved settings. When the sound engineer selects the “Save” menu item, the Project studio patchbay creates an XML document object and:

- Loops through all displayed *device blocks*. For each *device block* found, a corresponding device object is created and added to the XML object elements hierarchy. It then loops through the *device block* input and output plugs, creates a plug object for each plug found that is attached to the device XML element in the XML document object until all device blocks have been converted to XML objects.
- The XML document object is then saved into a text file within the workstation.

When the sound engineer selects the “Open” menu item on the “File” menu, the Project studio patchbay:

- Opens the “Open” file dialog box, where the user navigates to a specific text file location and selects the file to be opened. Clicking the *Open* button of the “Open” file dialog box loads the file data into the patchbay, and converts it into an XML object that is parsed as XML.
- The patchbay loops through all displayed *device blocks*, and compares them with the corresponding device XML objects.
- For each *device block* found, it gets each device plug displayed, and compares its variable “connected” value to that of the plug object for the loaded XML data. If the displayed plug variable value is “true”, whilst that of the loaded XML object is “false”, a disconnection is performed between the two device plugs, and the *connector line* between them deleted otherwise a connection is established between the two plugs and a connector line drawn that signifies the live connection between the two plugs.

6.5 Project Studio Patchbay Usability Testing

This section discusses the usability testing phase of the Project studio patchbay. This usability testing phase was not meant to be an exhaustive study but was done to ensure that Project studio patchbay requirements were sufficiently implemented, and the patchbay works properly.

Two different usability testing methods, namely the heuristic evaluation and the user testing, were selected to test the usability of the Project studio patchbay. A heuristic evaluation was the first technique applied. It detected minor usability problems and missing functionality in the software that were fixed as much as possible before the user testing of the software by prospective users. The heuristic evaluation process was based on the same "System Checklist" heuristic evaluation form that was used for evaluating the Broadcast patchbay [*section 5.4*], and is shown in *Appendix C-C3*.

The usability questionnaire used for testing the Project studio patchbay was completed by eight usability testing users who tested the system. Three of the users were local sound engineers, who were part of the Rhodes Computer Science Audio Engineering group. The other five users were asked to perform common connection management tasks on the patchbay according to their discretion and complete the usability testing questionnaire that was sent to them through email as they were located overseas (Germany and Canada) or in other parts of South Africa (Johannesburg and East London). They all fulfilled the test user requirements stated below:

- Have experience working within Project studio networks for at least 3 years.
- Are between the ages of 18 - 60 years.
- Have experience working with graphic-based patchbays.

6.5.1 Project Studio Patchbay Usability Testing Results

The users were asked to perform common studio tasks and to comment on their experience using a structured usability questionnaire that tested the Project studio

patchbay based on the following aspects, which are explained in detail in *section 5.4.2*:

- Consistency.
- Learnability.
- Screen positioning.
- Flexibility.
- Minimal action.
- Perceptual limitation**.**
- System capabilities.

The following sections present usability testing results for each aspect tested.

a) Consistency

No consistency issues were found on the Project studio patchbay. All users *strongly agreed* that the patchbay naming (font type and size) was consistent across displays and menu options, and the grouping and ordering of menu options was logical for all Project studio patchbay panels.

b) Learnability

All users found the Project studio patchbay to be easy to learn and use for a first time user. They all *strongly agreed* that the patchbay was easy to learn to use for performing connection management tasks and did not involve a steep learning curve. The Project studio patchbay allowed users to perform tasks easily, and they only needed to open very few panels to do this.

c) Screen Positioning

This section evaluated the positioning of different graphic components of the Project studio patchbay interface. All users *strongly agreed* that the patchbay's organization of information and components was logical.

d) Flexibility

All users *strongly agreed* that the Project studio patchbay allowed users to display patchbay components according to their needs and provided them with resizing

capabilities. The patchbay was found to be flexible enough, giving the user total control. Notable features of the Project studio patchbay included:

- Its drag and drop capabilities that allowed the user to drag *device blocks* to any point within the patchbay's working space. *Plug blocks* could be dragged to any of the eight positions of a *device block* [*section 6.2*].
- The resizing functionality, which allowed the user to resize the *device blocks* to view clearly the details of individual device graphics.
- The ability to expand and collapse *plug blocks* to view individual plug names when performing tasks such as establishing and breaking audio connections [*section 6.3.1.2*].

One of the users however, *disagreed* that the patchbay allowed users to display patchbay components according to their needs and perform connection management tasks flexible enough. He cited the following flexibility issues:

- Connections were disconnected by merely clicking on the connector lines joining connected devices. This meant when the sound engineer clicked a *connector line* by mistake, the connection was broken. This he thought was too much flexibility.
- When making connections, the system forced one to make the connection from the output *plug block* to the input *plug block*. From the way physical cable connections were made in a Project studio patchbay, one did not really "care" whether a cable was first plugged in the output or to the input. If the cable was plugged in the input first then one ensured that it goes to a source output and vice versa. Also, existing systems such as Yamaha's Graphic patchbay did not have such a restriction.
- He thought, when connecting two devices it made sense to place their input and output *plug blocks* facing one another. The patchbay did not allow this. The output *plug block* was always above the input *plug block* when both *plug blocks* were on the same side of a *device block*.

e) Minimal Action

Three users *disagreed* that the Project studio patchbay provided default values and function keys for frequently used control entries. It only provided the Tool bar icons for quickly accessing capabilities such as saving and opening routing settings, opening *Settings* and *Wordclock* panels and refreshing patchbay.

f) Perceptual Limitation

This section evaluated the aesthetic aspects of the Project studio patchbay interface components. All users *strongly agreed* that the system screen density was reasonable and the layout acceptable.

g) System Capabilities (Speed and Reliability)

All users *strongly agreed* that the system error messages were helpful and the patchbay error handling capabilities were accurate. Three users however, *disagreed* that the patchbay was reliable enough when performing connection management tasks. They found that when one reduces the size of a *device block*, for example, to something very small, clicking the *maximise* (+) button to expand the *plugs block*s did not clearly show all the plugs. Only when the size of the *device block* was increased, was it clearer, and. the resizing functionality seemed to be working well on the *device block* but not for the *plug blocks*, which remained squeezed together.

6.5.2 Improvements to the Project Studio Patchbay

The users suggested the following improvement to the Project studio patchbay, they thought:

- The patchbay should auto-detect devices on the network. Audio connections from the same device should be merged into a single *connector line* to reduce cable “clutter” on the interface.
- If one opens an input or output *plug block*, the *plug block* plugs should be listed with their connections. It will be nicer to incorporate a feature that when a user clicks and holds on a particular plug graphic (input or output) will show/highlight/draw a *connector line* to the other end – indicating the input or output it is connected to, and when the mouse is released, the connector lines disappear.

- If one opens an input or output *plug block,* both blocks merge into a window like the NAS Explorer but smaller so the user still has the graphic patchbay visible. The plug connections are listed as with the current NAS explorer (inputs left and outputs right).

6.5.3 Redesigning of the Project studio patchbay

The Project studio patchbay was redesigned to include some of the suggestions pointed out by the patchbay testers. Only the important and critical features of the patchbay were incorporated, and are discussed below:

- Instead of making audio connections only from the source plug, in the redesigned Project studio patchbay version, audio connections can be made from either the source plug or the destination plug.
- The resizing functionality was redesigned to make sure both *plug blocks* (input or output) zoomed proportionate to the *device block* they are attached to.
- When connecting two devices, their input and output *plug blocks* could be easily rearranged to face each other.

6.5.4 Good Features of the Project Studio Patchbay

The Project studio patchbay was found to be flexible enough, making it easy for the sound engineer to perform connection management tasks. The resizing feature made the device labels as well as the plug labels legible, assisting the user when routing audio.

6.6 Chapter Summary

This chapter demonstrates the power of using Adobe Flash for developing connection management applications. The Project studio patchbay involves a high degree of flexibility as it gives Project studio sound engineers total control of the application and allows connection management tasks to be executed easily. Adobe Flash's authoring IDE allowed the creation Project studio graphic components (MovieClips) and use ActionScript 2.0 to control these graphic components' behaviour. A computer prototype was used for gathering software requirement for the Project studio patchbay. A heuristic evaluation test assisted in ensuring the Project studio patchbay incorporated all standard GUI features before it was sent for usability testing by eight

usability testers locally (South Africa) and abroad (Canada and Germany). Using their feedback and suggestions, the Project studio patchbay was redesigned as required.

In the next chapter, the Rational Unified Process (RUP) for software development is applied in the development of a patchbay for Hospitality/Convention Centre networks.

CHAPTER 7

7 Hospitality/Convention Centre Patchbay Design and Development

Hospitality/Convention Centre networks are simpler than Broadcast and Project studio networks, and are usually operated by inexperienced users who have minimal or no knowledge of the inner workings of the studio network. Connection management applications for these networks hide away the physical complexity of the network. As a result, they are the easiest to develop and deploy compared to Broadcast and Project studio patchbays discussed in *chapters 5* and *6* respectively. Although at the time of this investigation, Hospitality/Convention Centre networks were common within the audio industry, very few connection management applications had been developed to control audio routing over these networks. Therefore, the first step in developing the Hospitality/Convention Centre patchbay was to determine which type of patchbay design (grid-based, list-based or graphic-based) would best suit these networks. Since a typical Hospitality/Convention Centre patchbay is usually operated by inexperienced users such a Hospitality/Convention Centre secretary who do not have to see the complexity of the network or understand how audio is actually routed on the network, the graphic-based patchbay design was chosen for the Hospitality/Convention Centre patchbay because it uses a pictorial representation of the network devices and their plugs, and hides away the complex naming conventions used in Broadcast and Project studio patchbays.

Having decided on the best design, the next step was to determine the layout for the Hospitality/Convention Centre graphic-based patchbay, and the functionality to be incorporated into the patchbay. To do this, three paper prototypes were designed and evaluated by five typical Hospitality/Convention Centre users to determine the best layout and the software requirements for the Hospitality/Convention Centre patchbay. According to Nielsen (1994), a paper prototyping is a cheap and fast technique for rapid iterative design of user interfaces. A human simulates the backend during paper prototype testing sessions. Advantages of using paper prototypes include the fact that they:

- Test the software design in the design phase before any code is written.
- Save money since cutting up paper and cards is not expensive in either material or time required.
- Save time since one can create a paper prototype in just a few hours not the days or weeks it takes to create a computer prototype.
- Are easy to learn because they are so simple.

The Hospitality/Convention Centre patchbay development process also followed the Iterative and Incremental Process (RUP). This chapter discusses how this process was used in the planning, designing and implementation of the Hospitality/Convention Centre patchbay.

7.1 Hospitality/Convention Centre Patchbay Requirements Analysis

The Hospitality/Convention Centre patchbay requirements analysis phase involved the use of three paper prototypes from which the initial design and layout of the patchbay was architected. The paper prototypes used included the following:

- The EMS-based hotel paper prototype – The EMS hotel system is a hotel system developed by a company called EMS in Johannesburg, South Africa. The paper prototype based on this system displays hotel rooms using picture graphic *zones*[11], and each *zone* source device is listed as a clickable button on the right of the main display [*Figure 7.1*].

[11] For purposes of this investigation, a *zone* is simply a room within a Hospitality/Convention Centre building.

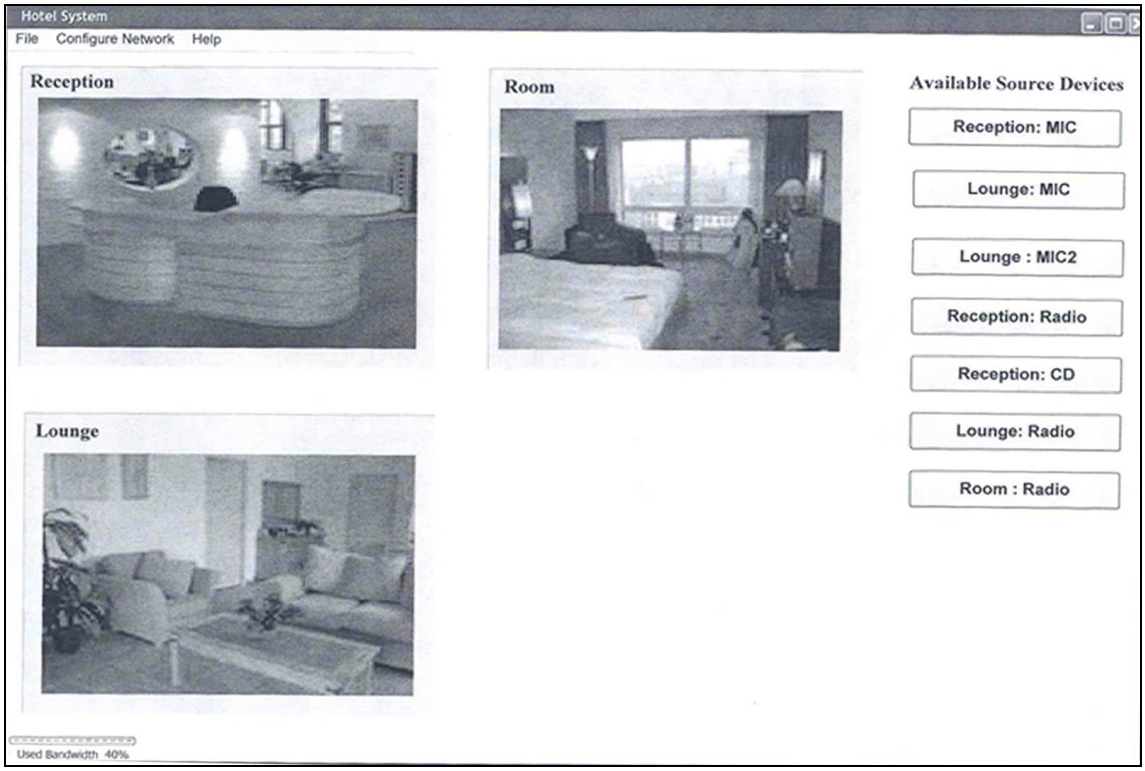

Figure 7.1: EMS-based Hotel Paper Prototype

- The daVinci-based hotel paper prototype – The daVinci hotel system is a customisable hotel system developed using the daVinci software program (also known as the Audio Digital Platform). daVinci is a software program designed to allow the creation and use of customized computer control screens with Audio and Nexia digital systems [Biamp Systems, 2007]. The paper prototype based on this system used a combination of buttons (representing a limited view of the network *zones* – building rooms) and dialog boxes (representing the detailed view of the network *zones* – building rooms) [*Figure 7.2*].

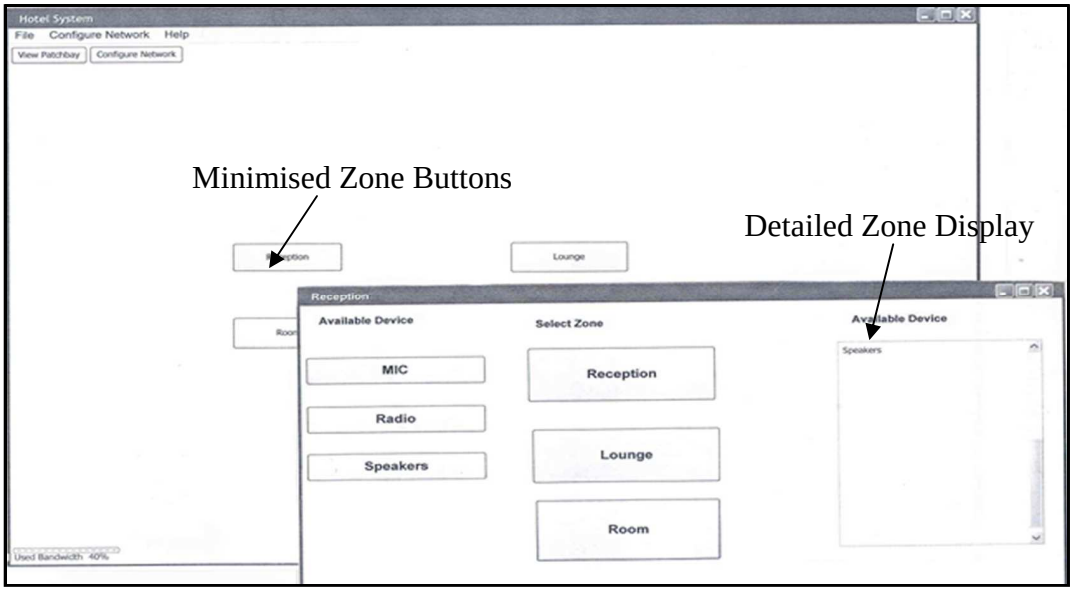

Figure 7.2: daVinc -based Hotel Paper Prototype

- Custom-built hotel paper prototype –A custom-built paper prototype was designed, which combines some features of the EMS and the daVinci hotel paper prototypes. It displays network devices as *zones* (square graphic boxes) and devices connected to the plugs of a displayed device (*zone*) are displayed using graphic picture icons that represent the device [*Figure 7.3*].

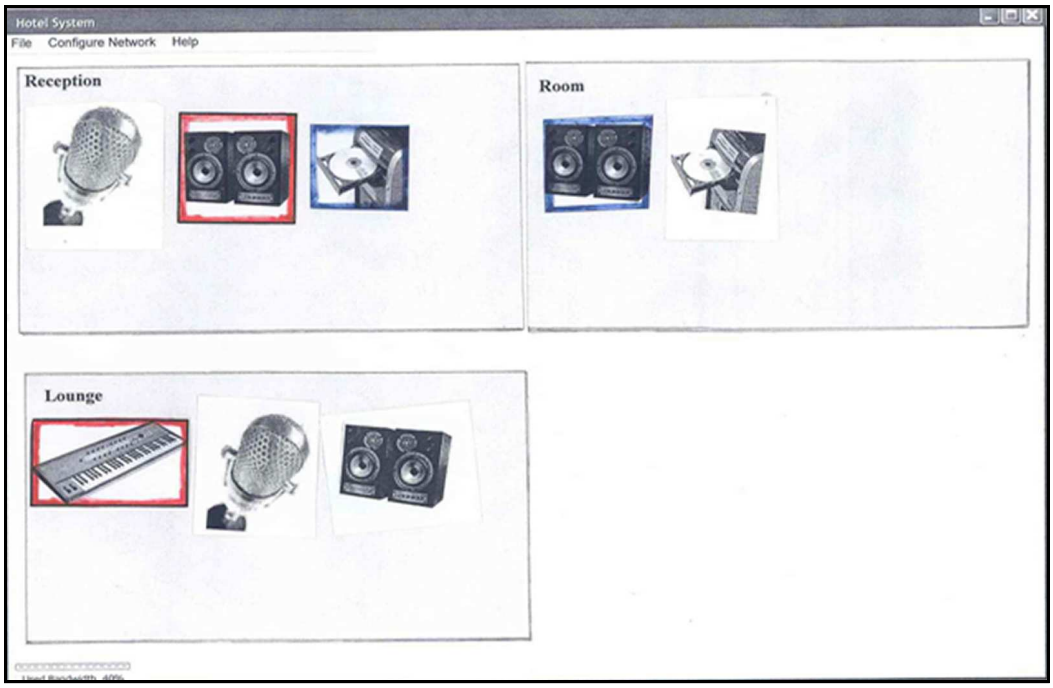

Figure 7.3: Custom-Built Hotel Paper Prototype

Five users were asked to perform five connection management tasks in face-to-face "think aloud" sessions. These sessions were video-taped and sound recorded for post

session feedback analysis. *Figure 7.4* shows the usability studio equipment that was setup and used for the face-to-face “think aloud” sessions.

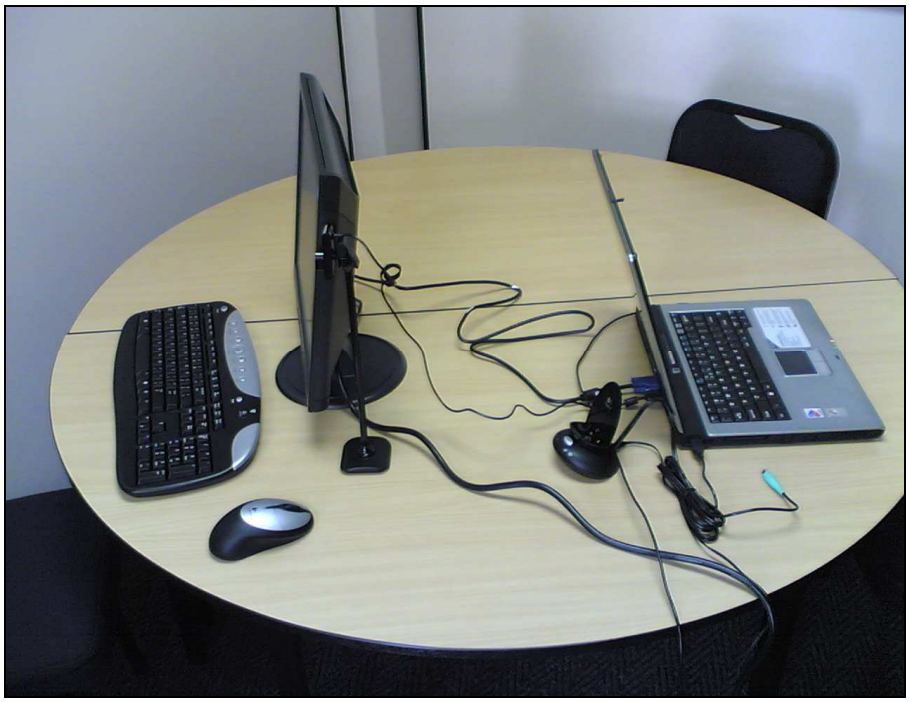

Figure 7.4: Usability Studio Equipment

Appendix C-C5 shows the tasks used that were used to test the paper prototype and the final patchbay software. The users fulfilled the following requirements:

- Had no prior experience working within audio networks.
- Between the age of 18 and 60 years.
- Had no experience using Hospitality/Convention Centre patchbays or any other connection management patchbay for routing audio.

Their preliminary feedback showed that they all preferred the custom-built hotel paper prototype, which displays each device within each *zone* using a picture graphic that reflects what that device is, for instance, they thought it was better to have a graphic picture of a speaker or a radio, not just a button labelled “speaker” or “radio”. This created a clear visual impression for the user. By looking at the interface, the user could see the devices in each *zone* and what they were. However, the interface displays fewer *zones* [*Figure 7.3*], so a way had to be found for displaying many *zones* at a time. The other two paper prototypes (the EMS-based and the daVinci-

based hotel paper prototypes) were found to have a difficult interface to use. The user was required to make too many clicks to perform a simple connection management task (on many different dialog boxes) such as making an audio connection. *Appendix A-A3* provides a complete Software Requirements Specification document (SRS) for the Hospitality/Convention Centre patchbay.The SRS gives a description of how each feature of the patchbay works using Stimulus/Response sequences. Using the user feedback, a Hospitality/Convention Centre patchbay was developed as described in the following section.

For purposes of this discussion, each *zone* displayed on the patchbay interface represents an individual mLAN device on the mLAN network while device icons displayed within each *zone* represent network devices that are connected to device plugs of devices represented by the *zones*.

7.2 Hospitality/Convention Centre Patchbay Description

The Hospitality/Convention Centre patchbay developed was named “FireZones” since it displays network devices as *zones*. It comprises four main panels; the *Control Window*, the *Settings* panel, the *Wordclock* panel, and the *Network Configuration* panel on which connection management tasks are performed.

The *Control Window* [*Figure 7.5*] is the main control panel, and has two sections, namely the “Additional Rooms” section, and the “Detailed Room Display” section. The “Detailed Room Display” section displays the detailed view of the devices in individual *zones* within a Hospitality/Convention Centre building. *Figure 7.5* displays three *zones* in the “Detailed Room Display” section of the Hospitality/Convention Centre patchbay, namely the *LOUNGE zone*, the *RECEPTION zone,* and the *ROOM 1 zone*. Devices in each of these *zones* are also displayed. For instance, the *LOUNGE zone* has three devices, namely the *MIC*, the *Speaker* and the *DVD Player* devices. The “Detailed Room Display” section displays a maximum of nine *zones* at a time. If the network has more than nine devices, the first nine of them are displayed in the “Detailed Room Display” section and the remaining devices are displayed as clickable buttons in the “Additional Rooms” section, such as the *ROOM 2* device [*Figure 7.5*]. The “Additional Rooms” section holds a maximum of 10 devices.

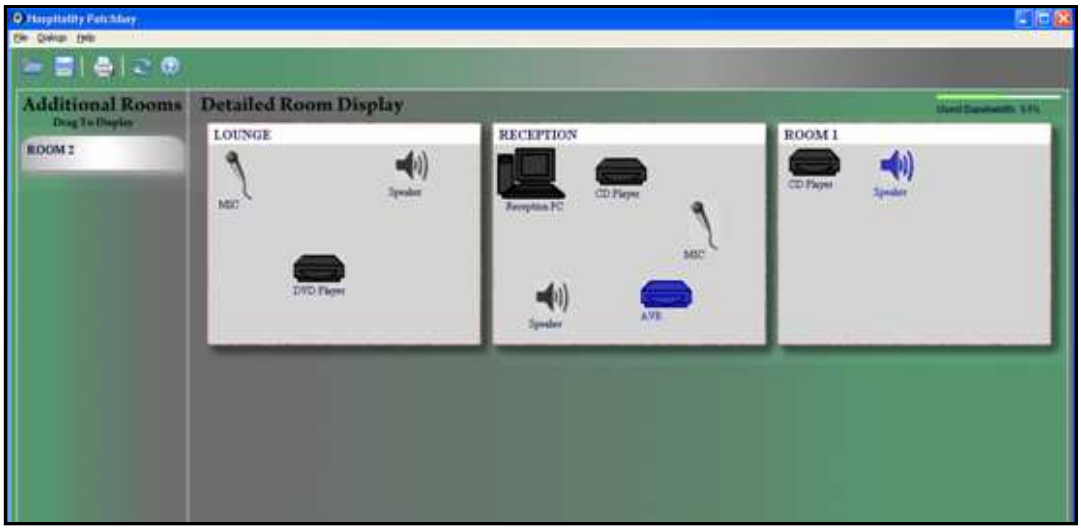

Figure 7.5: Hospitality/Convention Centre Patchbay Control Window

Connection management tasks that can be performed on the *Control Window* of the Hospitality/Convention Centre patchbay include:

- Establishing audio connections.
- Breaking audio connections.
- Clearing all device connections for a particular mLAN device.
- Saving/Opening routing settings into/from a text file.
- Identifying a particular device on the mLAN network.

The Hospitality/Convention Centre patchbay uses the same *Wordclock* panel and the *Settings* panel [*Chapter 5: section 5.1*] as the Broadcast patchbay and the Project studio patchbay described in *chapter 5.2*.

The *Network Configuration* panel [*Figure 7.6*] allows the sound engineers to name the network devices and their plugs. This avoids the display of meaningless names such as *OGT-IOne PC*, *MAP 4* and *Analog In 1*, which do not identify the device properly for a novice user in Hospitality/Convention Centre industries. The *Network Configuration* panel has two main sections, namely the "Network Devices and In/Out Plugs" section and the "User Names for Devices and Devices Connected to Plugs" section.

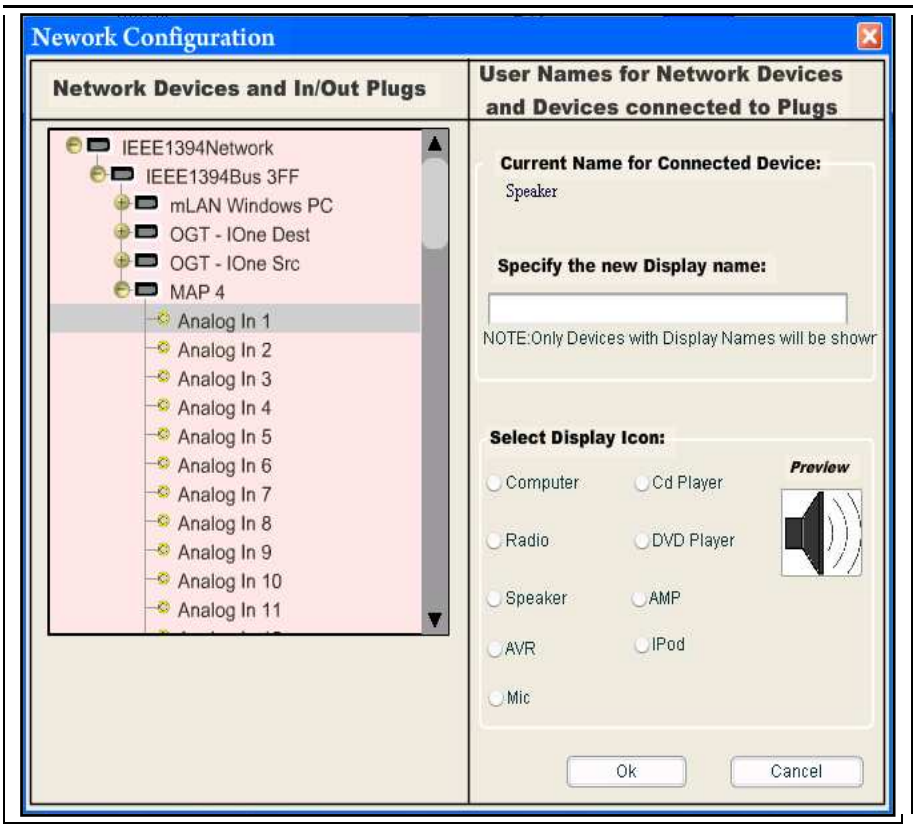

Figure 7.6: Hospitality/Convention Centre Patchbay Network Configuration Panel

The "Network Devices and In/Out Plugs" section displays a collapsible/expandable tree of the devices on the mLAN network and their plugs. The "User Names for Devices and Devices Connected to Plugs" section is where the sound engineer configures:

- The nicknames of the devices on the network.
- The names of input and output plugs of each device to reflect the devices attached to them (devices the plugs are streaming to/from).

To name a device, the sound engineer clicks the device node on the **"**Network Devices and In/Out Plugs" tree list. If the device already has a nickname assigned to it, it appears below the "Current Name of the Connected Device" in the "User Names for Devices and Devices Connected to Plugs" section otherwise "NOT SPECIFIED" is displayed. The new name is specified in the *Specify the New Display Name* text box. When the sound engineer clicks another node on the tree or clicks the *OK* button,

the name is automatically applied to the device. If the sound engineer is renaming or naming a plug, the plug is clicked on the “Network Devices and In/Out Plugs” tree list and specifies its new name, and selects the property display icon for it that shows the device this plug is streaming to/from. *Figure 7.6* shows the plug *Analog In 1* on the *MAP 4* device, which is streaming to a *Speaker*.

7.3 Hospitality/Convention Centre Patchbay Design and Implementation

This section describes the design and implementation of the Hospitality/Convention Centre patchbay using Sequence diagrams and the Object Model.

7.3.1 Project Studio Patchbay Sequence Diagrams and Implementation

The Hospitality/Convention Centre patchbay uses the same Use Case diagram as the Broadcast patchbay described in *chapter 5* [*section 5.2.1.1*] that shows the high-level functions of the Hospitality/Convention Centre patchbay and identifies two entities (Actors) that interact with the patchbay, namely the sound engineer and the mLAN Connection Management Server (mCMS). The Hospitality/Convention Centre patchbay high-level functions shown include:

- Managing files.
- Connecting to mCMS server.
- Establishing audio connections.
- Breaking audio connections.
- Updating system.
- Appling changes.
- Setting/Clearing word clock Master/Slave Configuration.
- Identifying a device.
- Changing device Plug Layout.

Figure 7.7 shows the Object Model that depicts the relationships and dynamic interactions between objects written for the Hospitality/Convention Centre patchbay. The model shows the objects for the Hospitality/Convention Centre patchbay human

interaction components and the mLAN network. The *IEEE1394 Network* object is shown to contain an aggregation of *IEEE1394 Bus* objects. Each *IEEE1394 Bus* object contains an aggregation of *IEEE1394 Device* objects, which in turn contains an aggregation of *Device Audio Plug* objects that can be either an output and input plug. The model also shows the interrelationships between the various Hospitality/Convention Centre patchbay panels (the *Settings* panel, the *Workclock* panel, *Login* panel, and the *Network Configuration* panel) and their associated XML document objects that represent XML messages used for communication purposes between the Hospitality/Convention Centre patchbay and the mCMS server.

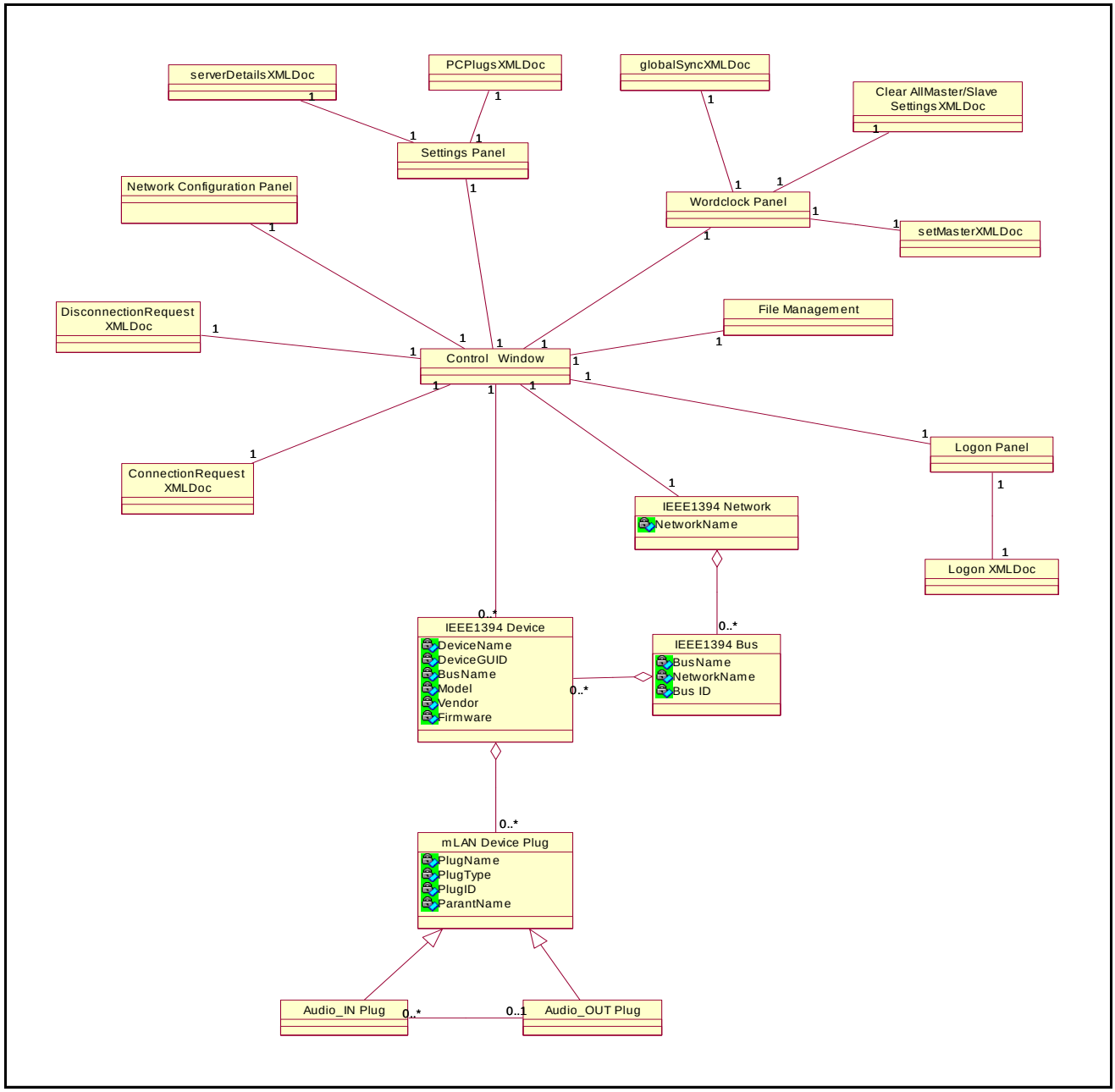

Figure 7.7: Hospitality/Convention Centre Patchbay Object Model

7.3.1.1 "Connect to mCMS Server" Use Case

At start-up, a TCP/IP connection is established between the Hospitality/Convention Centre patchbay and the mCMS server. When the user authenticates with the mCMS server, an XML "configuration" document is sent to the patchbay carrying the current status information of the mLAN network. The patchbay extracts the XML elements in the XML "configuration" document and recreates the mLAN network objects, and updates the Hospitality/Convention Centre patchbay. The sequence diagram, *Figure 7.8*, describes this process of network hierarchy recreation when the patchbay receives an XML "configuration" document.

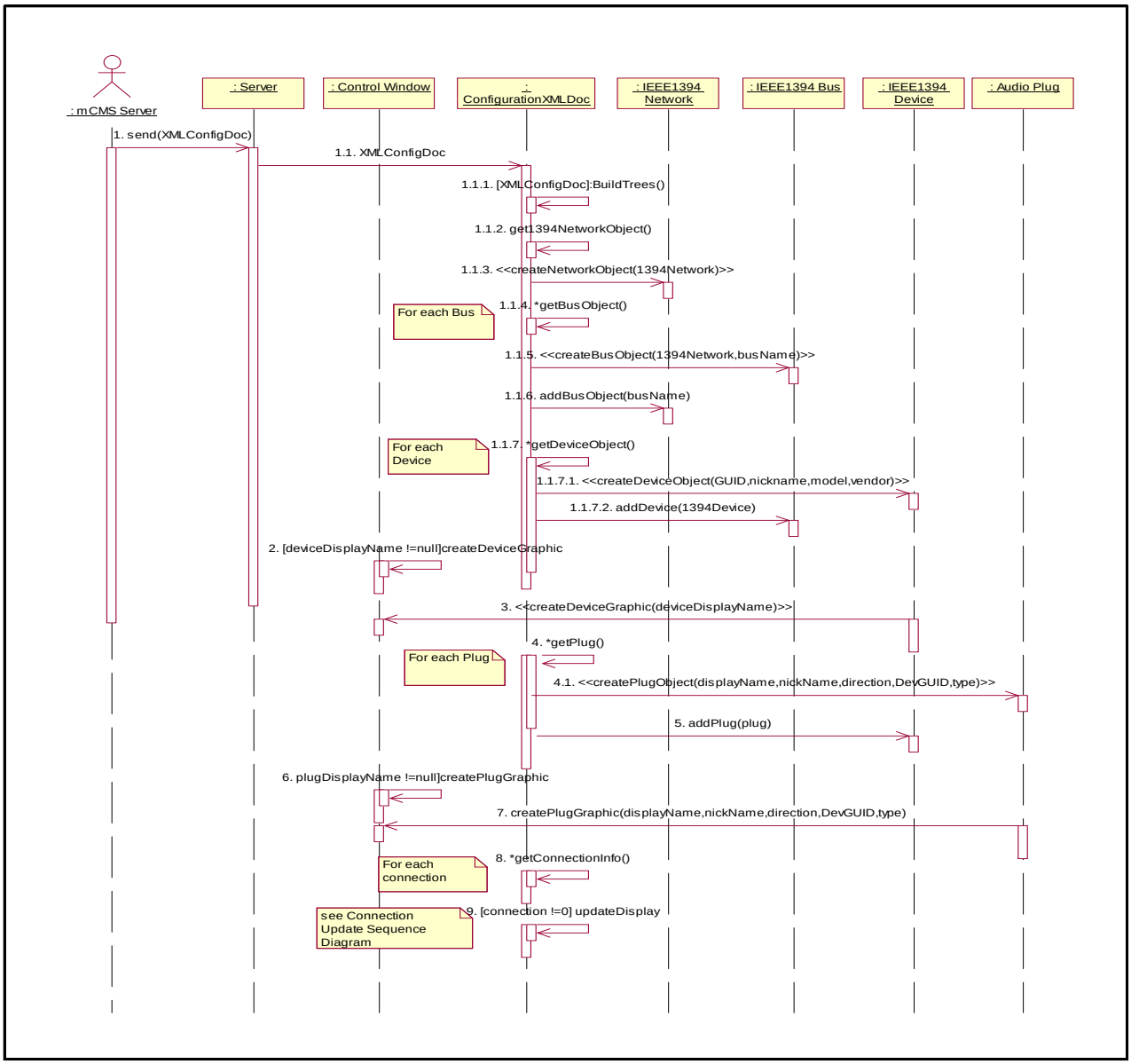

Figure 7.8: Receiving the Configuration XML Document Sequence Diagram

When the Hospitality/Convention Centre patchbay receives an XML "configuration" document from the mCMS server, it loops through its XML elements and recreate the network objects as follows:

- It gets the *IEEE1394 Network* XML element, and creates the *IEEE1394 Network* object representing the whole mLAN network.
- Loops through all *IEEE1394 bus* elements, and creates an *IEEE1394 bus* object for each XML *IEEE1394 bus* element found.

- For each *IEEE1394 bus* element, it loops through all *IEEE1394 device* elements, and creates *IEEE1394 device* objects. For each *IEEE1394 device* object created, if its "deviceDisplayName" variable value is not "null", a graphic square box (*zone*) is drawn on the patchbay interface.
- For each device element, it loops through all its plug elements, and creates *mLAN plug* objects, if the "plugDisplayName" variable value of this plug is not "null", a graphic representation of the plug is drawn, and attached to its device *zone* on the patchbay interface.
- For each *connection* element, loops through all its *patch* elements, and if any two devices attached to particular plugs have a live connection, a random colour is assigned as their background colour [*Figure 7.9*].

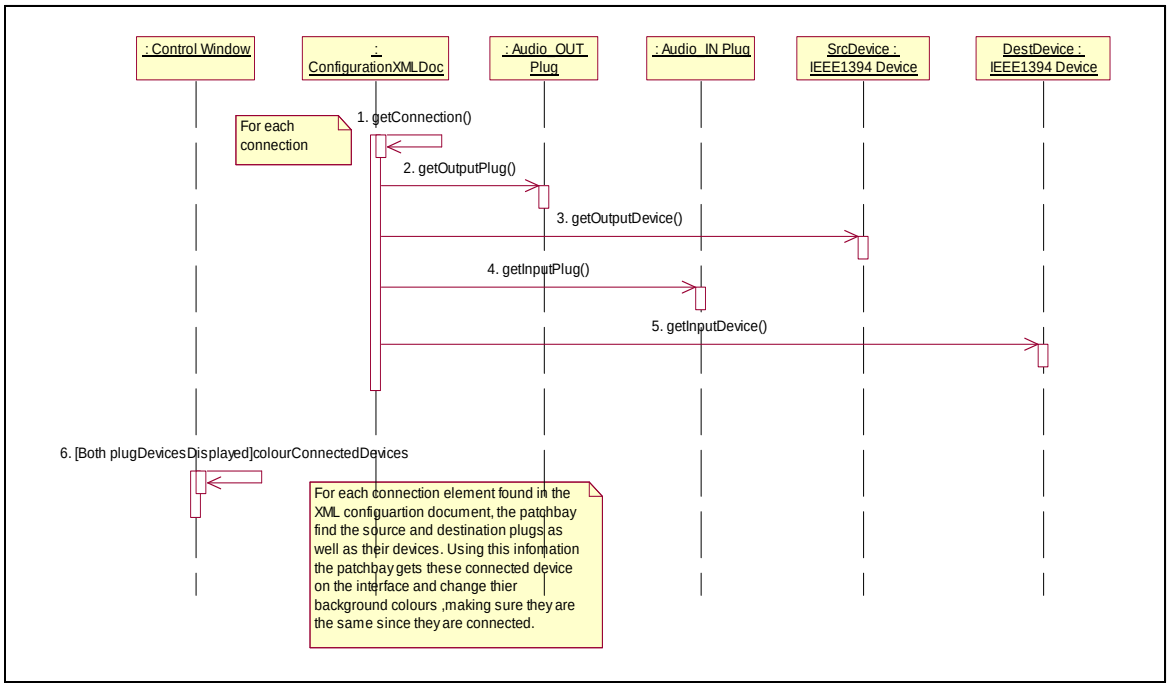

Figure 7.9: Update Plugs Connection Status Sequence Diagram

7.3.1.2 "Establishing Audio Connections" Use Case

The Hospitality/Convention Centre patchbay allows the sound engineer to establish audio connections between device plugs of the same type and on different devices. To establish the connection, the sound engineer simply clicks the device icon of the source device, its background colour changes to reflect that that device is now ready to be connected to another device. Having clicked the source device, the sound engineer clicks the output device like a Speaker to connect to in another *zone*. Devices

in the same *zone* cannot be connected. *Figure 7.10* is a sequence diagram that describes the process of creating audio connections.

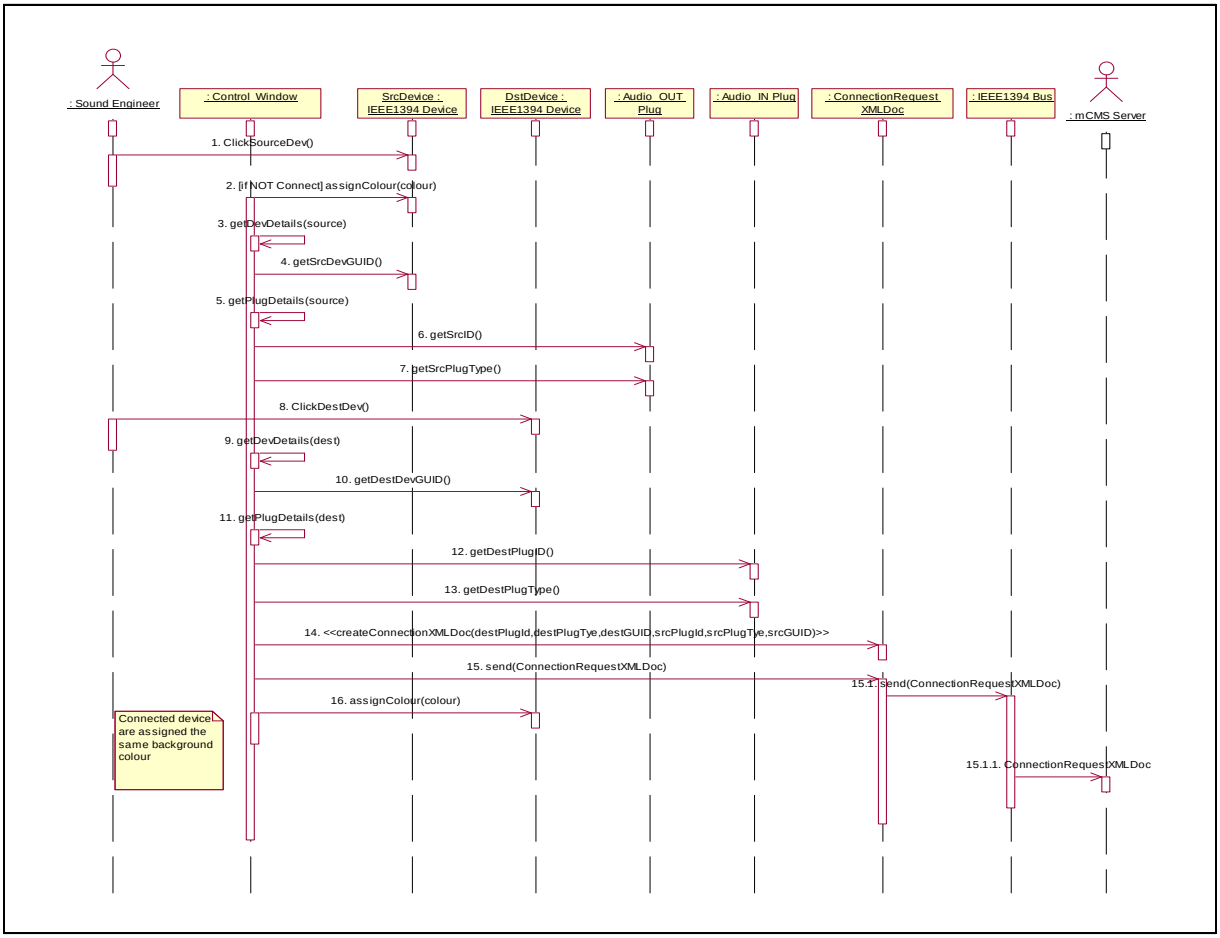

Figure 7.10: Establishing Audio Connection Sequence Diagram

When the sound engineer clicks both devices to be connected:

- The patchbay gets the details of both device plugs (the plug ID, the plug type and the plug nickname) that are added into an XML "connection request" document [*Chapter 5: Listing 5:3*], which is sent to the mCMS server that in turn implements the request through the Enabler module [*Figure 7.10*].
- A random colour is assigned to both devices, which represents the connection between both devices.
- Audio should be routed between the newly connected plugs.

7.3.1.3 "Breaking Audio Connections" Use Case

The Hospitality/Convention Centre patchbay allows the sound engineer to break audio connections made in the preceding section by right-clicking the destination device (for instance, the *Speaker* shown in *Figure 7.5*), and selects the "Disconnect Device" menu item. An XML "disconnection" request document is automatically created that contains information of the output device to be disconnected, and is sent to the mCMS server which invokes the Enabler to implement the request. If the request is successfully implemented, the output device icon colour is changed back to its default "grey". If this is the only connected device to a particular source device, its (the source device) icon colour is also changed to the default "grey", otherwise it retains its colour. It is also possible to disconnect all devices connected to a particular source device by right-clicking it and selecting the "Disconnect All" menu item. *Figure 7.11* is a sequence diagram that describes the process of disconnecting all audio connections for a particular source device.

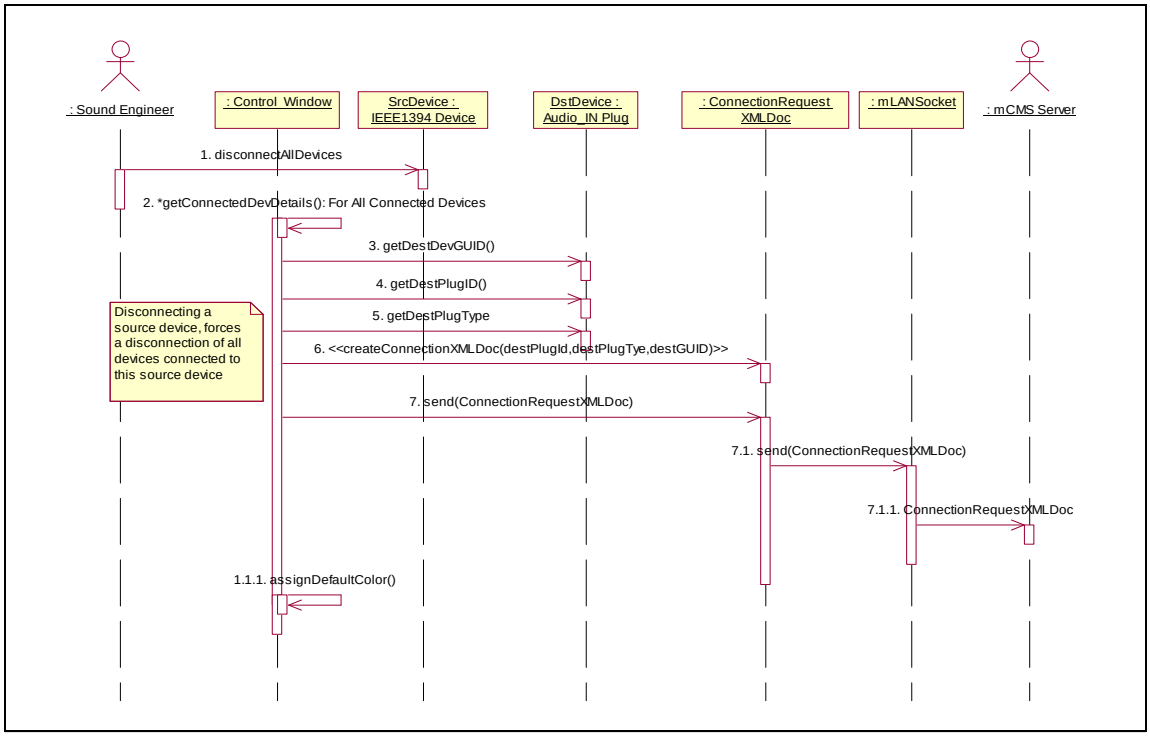

Figure 7.11: Breaking Audio Connections Sequence Diagram

If the patchbay is disconnecting all devices connected to a particular device [*Figure 7.11*]:

- It loops through all connected devices, and in each case an XML "disconnection" request document is created and sent to the mCMS server to disconnect that device (one of the destination device to be disconnected).
- The device icon colour of the disconnected device is changed to the default "grey".
- At the end of disconnecting all connected devices, the colour of the source device icon is also changed to the default grey.

7.3.1.4 "Setting/Clearing Word Clock Master/Slave Configurations" Use Case

The Hospitality/Convention Centre patchbay uses the same method for setting and clearing word clock Master/Slave configurations as the Broadcast and the Project studio patchbays described in *section 5.3.1.4*.

7.3.1.5 "Identify Device" Use Case

In large Hospitality/Convention Centre networks, the sound engineer identifies a particular *zone* device on the network by right-clicking the *zone* device graphic box on the patchbay interface, and selects the "Identify Device" menu item. The patchbay automatically creates an XML "identify device" request document that contains the GUID of the *zone* device to be identified, and is sent to the mCMS server [*Figure 7.12*]. If the request is implemented successfully, the device LEDs flash.

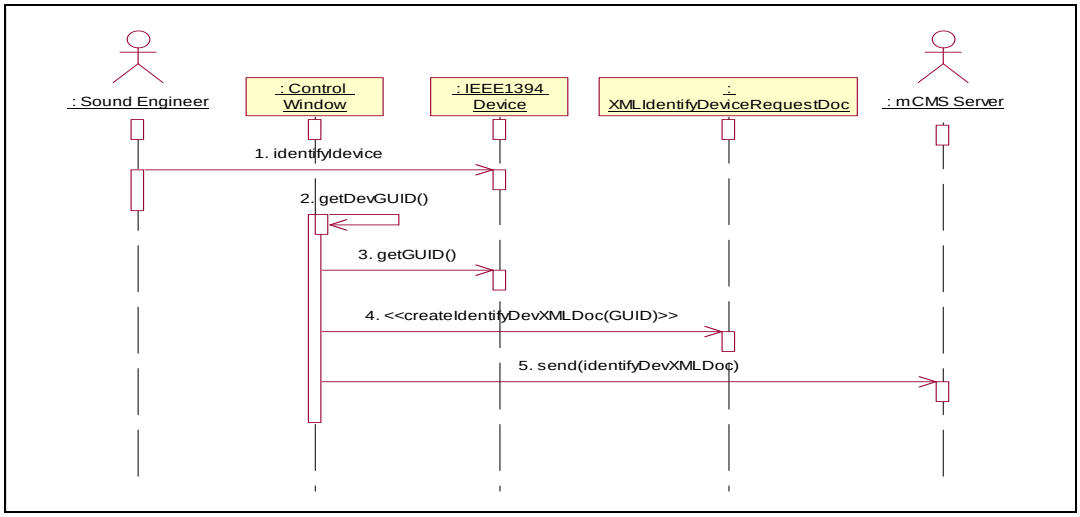

Figure 7.12: Identify Device Sequence Diagram

7.3.1.6 “Managing Files” Use Case

The Hospitality/Convention Centre patchbay also allows the sound engineer to save routing settings into a text file as well as load saved routing settings into the patchbay. The sound engineer does this by selecting the “Save” menu item to save the routing settings into a text file and the “Open” menu item to open saved settings. When the sound engineer selects the “Save” menu item, the Hospitality/Convention Centre patchbay creates an XML document object and:

- Loops through all displayed *zone blocks*. For each *zone block* found, a corresponding device object is created and added to the XML object elements hierarchy of the XML document object.
- For each *zone block*, the patchbay loops through all device icons, and create XML plug objects for each plug icon displayed within a particular *zone block*. The XNL plug elements are attached to their device XML element in the XML document object.
- The XML document object is then copied into a text file that is saved within the workstation.

When the sound engineer selects the “Open” menu item, the Hospitality/Convention Centre patchbay:

- Opens the “Open” file dialog box, where the user navigates to a specific text file location and selects the file to be opened. Clicking the *Open* button of the “Open” file dialog box loads the file data into the patchbay, which converts it into an XML object that is parsed as XML.
- The patchbay loops through all displayed *zone blocks*, and compares them with their corresponding device XML objects in the loaded XML document.
- For each *zone block* found, it loops through each device icon displayed, and compares their variable “connected” value to that of the XML plug object for the loaded XML data. If the displayed icon variable value is “true”, whilst that of the loaded XML object is “false”, a disconnection is performed between the two devices plugs and their colours updated appropriately otherwise a connection is established between the two plugs.

7.4 Hospitality/Convention Centre Patchbay Usability Testing

This section discusses the usability testing phase of the Hospitality/Convention Centre patchbay. This usability testing process was not meant to be an exhaustive study but was done to ensure that Hospitality/Convention Centre patchbay requirements captured in the beginning of this chapter using the paper prototypes were sufficiently implemented, and the patchbay works properly.

The five users used for testing the paper prototypes were asked to perform five connection management tasks shown in *Appendix C-C5* in face-to-face "think aloud" sessions. The tasks evaluated the main functionalities of the patchbay, which include:

- Establishing and breaking audio connections.
- Rearranging the layout of *device icons* with a *device zone*.
- Placing *device zones* at different positions of the "Detailed Room Display" section of the patchbay.
- Swapping *device zones* between the "Additional Rooms" and the "Detailed Room Display" sections of the patchbay.

Once again, the testing sessions were video-taped and sound recorded for post session feedback analysis. *Figure 7.4* shows the usability studio equipment that was used for the final face-to-face "think aloud" sessions for testing the Hospitality/Convention Centre patchbay.

As already mentioned, the test users fulfilled the following requirements:

- Had no prior experience working within audio networks.
- Between the age of 18 and 60 years.
- Had no experience using Hospitality/Convention Centre patchbays or any other connection management patchbay for routing audio.

At the end of each test session, the test coordinator and the user were involved in a short debriefing session where the user had the opportunity to provide additional comments about the system. This session was used to get feedback from the user about other aspects of the patchbay that were not covered by the five usability tasks such as the effect of the patchbay background on the user's eyes, the general layout of the device *icons* and *zones* and the overall performance of the patchbay, to mention but a few. Users were also given a usability questionnaire to complete. The usability video footage and the usability questionnaires were later analysed, and a list of usability issues drawn for each patchbay that were raised by the users. The user tasks and the usability questionnaire evaluated the following aspects of the patchbay, which are explained in detail in *section 5.4.2*:

- Consistency.
- Learnability.
- Screen positioning.
- Flexibility.
- Minimal action.
- Perceptual limitation**.**
- System capabilities.

The following section presents some of the critical user feedback that was used for redesigning the Hospitality/Convention Centre patchbay with respect to the above stated aspects.

7.4.1 Hospitality/Convention Centre Patchbay Usability Testing Results

This section discusses the usability testing finding for the Hospitality/Convention Centre patchbay.

a) Consistency

Inconsistency issues were found on the Hospitality/Convention Centre patchbay. When making audio connections, the sound engineer clicked the two device icons to be connected but when making an audio disconnection, the sound engineer had to

right-click the icon to be disconnected and select the “Disconnect Device” menu item. This resulted in one user *disagreeing* that the patchbay operations were consistent. He expected that to perform a disconnection, one would simply click the destination device icon to be disconnected and the disconnection is performed automatically. All users *strongly agreed* that the patchbay naming conventions (font type and size) used were consistent across displays and menu options and that the grouping and ordering of menu options was logical for all Hospitality/Convention Centre patchbay panels.

b) Learnability

All users found the Hospitality/Convention Centre patchbay to be easy to learn and use for a first time user. They all *strongly agreed* that the patchbay was easy to learn to use for performing connection management tasks and did not involve a steep learning curve for a new user. They thought the use of familiar graphic pictures for device icons within device *zones* made the patchbay visually pleasing and easy for the user to understand and manipulate.

c) Terminology, User Guidance and System Information

All users *strongly disagreed* that the patchbay provided user guidance information. The system did not provide the help system. They however, all *agreed* that the terminology used especially on the *Network Configuration* and *Wordclock* panels was related to the connection management tasks performed on those panels.

d) Screen Positioning

Screen positioning evaluated the placing of different graphic components on the Project studio patchbay interface and its panels. All users *strongly agreed* that the patchbay’s organization of information and components was logical and standard. All panels opened at the centre of the patchbay.

e) Flexibility

All users *strongly agreed* that the Hospitality/Convention Centre patchbay was truly flexible because it allowed the user to drag around individual device icons within a device *zone*. This way the user had the opportunity to customise the layout of device icons on the patchbay interface to fit one’s needs. *Zones* could also be dragged from

the “Detailed Room Display” section to the “Additional Rooms” section and vice versa.

f) Minimal Action

All users *agreed* that the Hospitality/Convention Centre patchbay provided tool bar icons for quickly accessing capabilities such as saving and opening routing settings, opening *Settings* and *Wordclock* panels and refreshing the patchbay.

g) Perceptual Limitation

This section evaluated the aesthetic aspects of the Hospitality/Convention Centre patchbay interface components. All users *strongly agreed* that the system components layout was logical and clear. They commended the use of the coloured background for differentiating the two main sections of the patchbay, namely the “Detailed Room Display” section and the “Additional Rooms” section.

h) System Capabilities (Speed and Reliability)

The Hospitality/Convention Centre patchbay speed when performing connection management tasks was commendable. Audio connections were made by simply clicking the two icons of the devices to be connected. However, some users found that when making audio connections, the colour of the clicked device icon sometimes did not change from just one click, but required the user to make many clicks. Sometimes only one click worked. As a result, three users *disagreed* that the system was reliable enough.

7.4.2 Redesigning of the Hospitality/Convention Centre Patchbay

The Hospitality/Convention Centre patchbay was redesigned to incorporate the important aspects and features from analysing usability test feedback. Notable improvements include the following:

- Instead of making audio disconnections by right-clicking the destination device and selecting the “disconnect device” menu item, in the redesigned version, one needs to just click the destination device to be disconnected and the disconnection should automatically be implemented.

- Making audio connections was modified such that only one click on each device is sufficient for the connection to be implemented.
- A simple user-guide help system was included with the application.

7.4.3 Good Features of the Hospitality/Convention Centre Patchbay

The Hospitality/Convention Centre patchbay displays devices in each *zone* clearly using graphic pictures that represent the actual devices on the network. This makes the patchbay simple to use since the user can immediately recognise what devices are in a particular *zone* without necessarily having to read their labels, which are sometimes illegible. Audio connections could be made and broken easily, and with least effort. The patchbay background colour was also friendly to the eyes, and also makes efficient use of the available space. The shadows on each *zone* graphic made it easy for them to visually stand out as clickable buttons that the user could not miss.

7.5 Chapter Summary

This chapter describes the use of Adobe Flash for developing a Hospitality/Convention Centre patchbay for use in Hospitality/Convention Centre networks. Although the Hospitality/Convention Centre patchbay was the simplest of the three patchbays developed in this investigation, it tested the Adobe Flash's programming and graphic authoring capabilities. Three paper prototypes that were evaluated by five potential hotel users were used for gathering system requirements at the beginning of the Hospitality/Convention Centre patchbay development process. Stimulus/Response sequences were utilised to describe features of the patchbay before the RUP software development process was followed in the development of the patchbay. The heuristic evaluation was done on the patchbay to determine missing functionality in the Hospitality/Convention Centre patchbay before usability testing was performed. The development of the Hospitality/Convention Centre patchbay was made easier by Adobe Flash's graphic authoring IDE and ActionScript 2.0 capabilities. This chapter shows the power of using ActionScript 2.0 code to create and control the behaviour of graphic MovieClip symbols. MovieClip symbols were created at run time depending on the devices currently on the network using ActionScript 2.0 code, which are a referred to as *zones* and graphic device icons on the patchbay interface. Not only was ActionScript 2.0 code able to create MovieClip symbols but also attached to them code handlers to listen and handle user interactions

with the symbols, hence controlling their behaviour. The Adobe Flash's ActionScript 2.0 XML API and the XMLSocket class were important in the creation of various XML request documents for communication between the patchbay and the mCMS server.

The next chapter concludes this investigation and presents lessons learnt from using Adobe Flash Professional 8 for developing connection management applications for different sound installation networks as discussed in *chapters 5*, *6* and *7*.

CHAPTER 8

8 Adobe Flash Professional 8 Tools for developing mLAN Client/Server Patchbays

This chapter discusses Adobe Flash Professional 8 IDE tools, and ActionScript capabilities that were employed in the development of three patchbays, namely the Broadcast patchbay, the Project studio patchbay, and the Hospitality/Convention Centre patchbay, which are described in *chapters 5*, *6* and *7*, respectively. As already mentioned, the mLAN Client/Server configuration uses XML messages for communication between the client applications and the mCMS server. This grants client applications independence from the Enabler module, and therefore client applications can be developed using any language as long as it provides XML capabilities.

8.1 Development of the Patchbay User Interfaces

This section describes the Adobe Flash graphic and ActionScript capabilities that were utilised for developing the user interface components for the three patchbays.

8.1.1 Adobe Flash Built-In Components

Adobe Flash incorporates common IDE built-in components offered by other common development IDEs such as Visual Studio.Net for developing graphic user interfaces. Some of these components that were useful in developing the three patchbays include; the Button, the TextInput, the ComboBox, the Label, the Checkbox, the RadioButton and the ProgressBar components. However, although Adobe Flash provides these basic components, it does not provide complex built-in components such as Windows Forms components, the TabControl component, and the FontDialog component, to mention but a few, which enable much GUI development. As a result, combinations of these basic components were used for creating simple panels such as the “Device Information” panel [*Figure 8.1*] that were in turn used for creating GUI applications.

Figure 8.1 shows the “Device Information” dialog box used by the Project studio for displaying detailed information for a particular device on the mLAN network. The

diagram shows the use of the TextInput, the ComboBox, and the Button components. When the user opens the "Device Information" dialog box of the Project studio patchbay, by clicking the *Information* at the centre of each *device block*, the patchbay queries the device object attached to the *device block* for information and then updates the panel with the device information.

Adobe Flash provides ActionScript event handlers for all built-in components and they can be used to control their behaviour. Adobe Flash also provides programming interfaces that enable the specification of commands using ActionScript code, which add various kinds of functionalities to components as discussed in the following sections. The TextInput component is used for displaying the nickname of the device, and allows the user to input a new name for the device. The Label component displays non-editable device information such as the device model name, the device GUID, the device vendor, the device firmware name, and the device current Plug Layout.

The ComboBox component is used for displaying all supported Plug Layouts for the device and also allows the user to change the current Plug Layout by choosing the desired Plug Layout from the component dropdown list. The Button component is used by all three patchbays for different purposes. In *Figure 8.1*, the Button component is used as the *OK* button for saving new device settings (the new device name and the Plug Layout), and for exiting the dialog box.

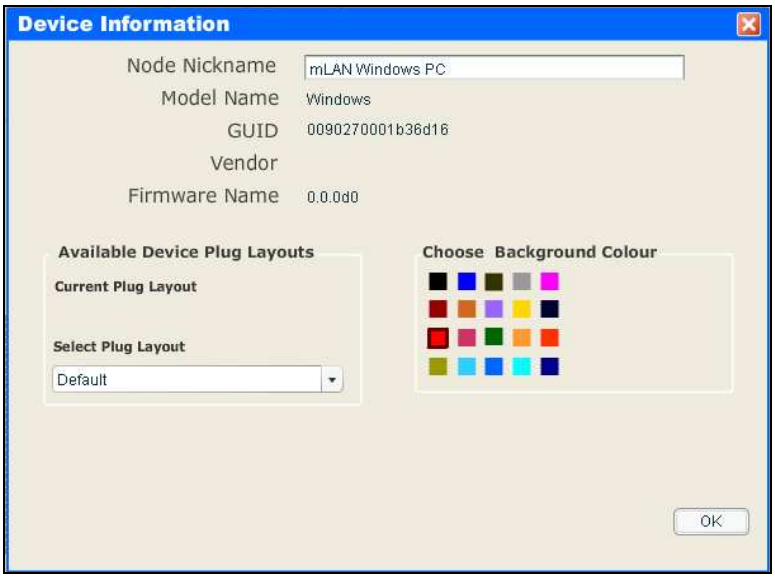

Figure 8.1: Project Studio Patchbay – Device Information Dialog Box

To use the built-in components, the user drags the component from the Adobe Flash *Components* panel onto the *Stage*[12]. An instance of that component is created on both the *Stage* and the Adobe Flash *Library* [*Chapter 4*]. The user can then set the properties of the component on the *Stage* using the *Properties Window*[13]. The user can use ActionScript code or the *Properties Window* to set different component properties.

8.1.2 Adobe Flash Graphic Authoring IDE and ActionScript Capabilities

As discussed in the preceding three chapters [*Chapters 5*, *6* and *7*], the patchbays developed required the use of different kinds of graphic components. The basic element for creating graphic components in Adobe Flash is a symbol known as a MovieClip [*Chapter 4*]. A MovieClip symbol is a reusable piece of flash computer graphic, usually consisting of one or more graphic or button symbols [Adobe Macromedia, 2005]. A MovieClip object is assigned by default to all MovieClip symbols that are created in Adobe Flash. Three types of symbol formats were useful for this investigation, namely the graphic symbol, the MovieClip symbol, and the button symbol. They all have different behaviours, and can be controlled using ActionScript code. In the current discussion, only the MovieClip symbol attributes and behaviours are discussed in detail since they formed the basis for all graphic components and panels created for the patchbays, and in some instances incorporated the other two symbols, namely the graphic symbol and the button symbols.

8.1.2.1 Adobe Flash Graphic Tools and ActionScript Capabilities for Developing the Broadcast Patchbay

The Broadcast patchbay comprises four main parts, namely the menu and tool bars, the *Destinations* and the *Sources* tree, the grid-matrix and, the legend [*Figure 8.2*]. This section explains how each of these parts was developed using Adobe Flash graphics and Adobe Flash ActionScript capabilities.

---

[12] As already stated in *Chapter 4*, the *Stage* in Adobe Flash is the working space where the developer place and manipulates media when developing applications.

[13] *Chapter 4* describes the *Properties Window*.

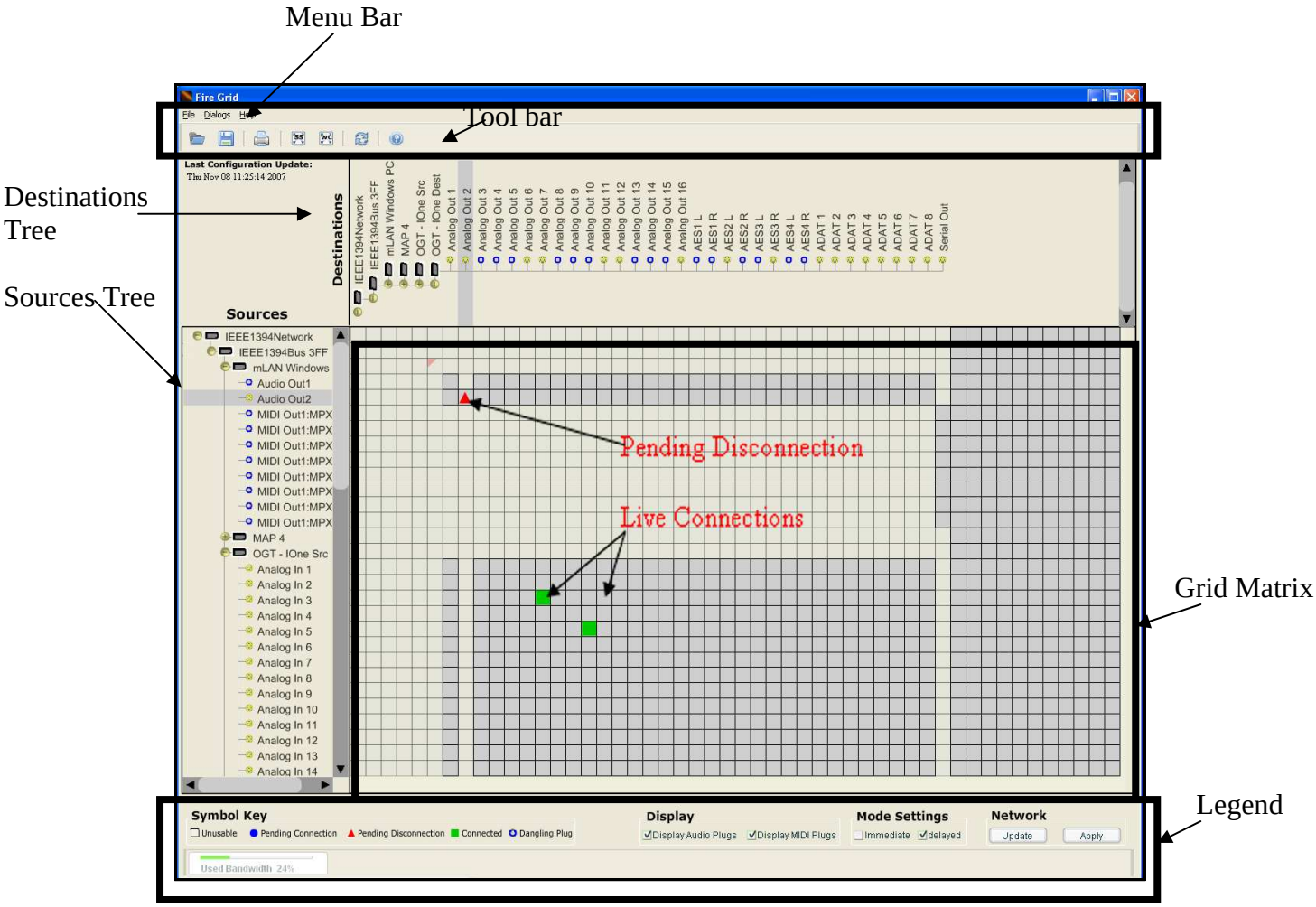

Figure 8.2: Broadcast Patchbay Interface Components

a) Menu and Tool Bars

The Adobe Flash graphic authoring IDE played an integral role in creating the menu and tool bars. The menu bar is created at run-time using ActionScript code and a handler is attached to each menu bar item, which enables it to provide the functionality it is supposed to achieve. Examples of the menu bar item's functions include opening and saving routing settings into a text file, and opening various patchbay panels. These functions and more are also provided by the tool bar icons. The tool bar icons are created by inserting graphic pictures, that reflect the function of the icon, inside MovieClip symbols, and adding interactivity to them using ActionScript code.

b) Destinations and Sources Tree

An important feature of the Broadcast patchbay is to display as many possible devices on the mLAN network as possible. This was achieved by using two tree list structures, namely the *Destinations* and the *Sources* trees [*Figure 8.2*]. Using the *Macromedia Extension Manager*[14], a new customisable tree component called the "advancedTree" component[15] was used for displaying the *Destinations* and the *Sources* trees. This is because Adobe Flash does not provide an advanced tree component to display the complex tree structures. Moreover, the Adobe Flash IDE does not provide the source code for built-in components to which additional event handlers and functionality could be added. This was a requirement since the intention was to modify the look of the component and add more functionality to make it suitable for use in a Broadcast patchbay.

The "advancedTree" component displays tree lists inside a rectangular scrollable component. Each tree list item has two parts, the node icon and the node label. Node icons were assigned to tree list nodes depending on the type of the node, whether it is a plug or not. Each node icon is a custom-made graphic that is assigned to the node by setting its *leafGraphic* attribute value, either using ActionScript code, or the *Properties Window*. Node labels are assigned automatically when the patchbay receives the XML configuration document from the server and loads them into the components. Each tree list node is assigned an object that keeps information about the node, its connection status and methods that control its behaviour. This information and methods were used for fulfilling connection management tasks requested by the user.

The"advancedTree" component, with its scroll bar and scroll buttons made it possible to display many device nodes on the patchbay thus achieving the requirements for the Broadcast studio network, which deal with hundreds of plugs at a time. In addition to this, the "advancedTree" component provided built-in event handlers for handling mouse events, thus enabling the specification of ActionScript logic for opening the

[14] *Macromedia Extension Manager* is an application that comes with Adobe Flash software that enables the importation of new media into the Adobe Flash IDE.

[15] http://www.flashloaded.com/userguides/advancedtree/ [*Flashloaded*, 2007].

nodes and updating the grid-matrix to display the connection status of the newly opened nodes.
ActionScript code was used to skin the “advancedTree” component so that it fitted well with other components of the patchbay interface.

c) Grid-Matrix

The grid-matrix was created using a single MovieClip symbol (with black borders) that was duplicated at run-time to create a 29 row by 50 column grid-matrix. To make a MovieClip symbol, a graphic element is drawn on the *Stage* using the graphic authoring tool provided by the Adobe Flash IDE, and then converts the created graphic into a MovieClip as described in *chapter 4*. There is need for writing lines of code to create graphic components as is the case for third-generation IDEs. Each grid-matrix box is made of two parts, namely the MovieClip symbol and the ActionScript object that is attached to it, which contains properties and methods that control its behaviour. Each MovieClip symbol on the grid-matrix is a cross-point of two nodes displayed on the *Destinations* and *Sources* trees, and displays their connection status. Five states can be displayed on each grid-matrix box. The states include:

- The “connected” state – This state exists only if two plug nodes of the same type and on different devices, are connected, and is represented on the grid-matrix by a green grid-matrix box.
- The “not connected” state – This state exists between two plug nodes that are not connected, and is represented on the grid-matrix by the default grey grid-matrix box.
- The “cannot be connected” state – This state exists between two nodes that cannot be connected because:

    a) They are of different types. For instance, a MIDI plug node and an audio plug node. Only plug nodes of the same type can be connected.
    b) One of them or both are not plug nodes. Only plug nodes can be connected.

c) They are plug nodes on the same device. Internal routing is not allowed, and as a result they can not be connected to each other.

This state is represented on the grid-matrix by a yellowish grid-matrix box.

- The “pending connection” state – This state exists between two newly connected plug nodes if the system is in “Delayed Mode”, and the connection has not been committed to the actual physical audio network but only exists on the patchbay. This state is represented on the grid-matrix by a blue round grid-matrix box.
- The “pending disconnection” state – This state exists between two newly disconnected plug nodes if the system is in “Delayed Mode”, and the disconnection has not been committed to the actual physical audio network but only exists on the patchbay. This state is represented on the grid-matrix by a red triangle grid-matrix box.

d) Legend

The Broadcast patchbay legend uses the Checkbox component, the Button component, and the ProgressBar component. The “Display” Checkbox components allow the viewing of only the MIDI plugs, only the audio plugs, or both plug types nodes on the *Sources* and *Destination*s trees by simply clicking the appropriate Checkbox component. Each Checkbox component is a graphic symbol that is converted into a MovieClip, and has action handlers added to it. Assuming the user wants to display only MIDI plug nodes, he unselects (if the Checkbox is selected) the audio plugs Checkbox component by clicking it, and making sure the MIDI plugs Checkbox component is selected. This action invokes an action handler that contains ActionScript code that will tell the patchbay to loop through both tree lists and deleted all audio plug node for each available device node. When the nodes are delete from the *Sources* and *Destination*s trees, the grid-matrix is updated so that it displays the connection status of only the displayed MIDI nodes.

The “Mode Settings” Checkbox components allow the user to set the mode state of the patchbay to either “Immediate Mode” or “Delayed Mode”. These components work just like the “Display” Checkbox components described above. They use

ActionScript code to change the state of the patchbay depending on which component was selected or unselected.

The ProgressBar component is used to display the amount of available bandwidth units on the network. At run-time the server sends the XML configuration document with the latest “used network bandwidth” value, and the patchbay uses the ActionScript code to calculate the “available network bandwidth” value and automatically invoke the ProgressBar component *load ( )* handler to load the value units.

The Button component was used to create the *Apply* and *Update* button instances for the “Network” button section. When clicked, the *Apply* button applies the changes made (audio connections established in “Delayed Mode”) on the patchbay to the actual network, while the *Update* button uses its ActionScript handlers to force a refresh process by creating and sending an XML “refresh” document to the server, which in turn responds by sending an XML “configuration” document to the patchbay.

8.1.2.2 Adobe Flash Graphic Tools and ActionScript Capabilities for Developing the Project Studio Patchbay

Different MovieClip symbols were used for creating the Project studio patchbay *device* and *plug blocks* [*Figure 8.3*]. Each *device block* is made of a graphic element that is converted into a MovieClip symbol. The *device block* incorporates a button and a label. The button uses its ActionScript code defined in its *onPress ( )* handler to open the *Device Information* panel, which displays more information about a particular device node. One of the labels displays the name of the device represented by the *device block* while the other displays clock synchronisation information for the device. When the sound engineer changes the Master/Slave relationship and the sample rate information of devices on the network, the labels displaying clock synchronisation information for all displayed *device blocks* are updated accordingly. *Device blocks* each have an ActionScript object attached to them that provide handlers, which control their behaviour. Each *device block* handles mouse actions. The ActionScript *startDrag ( )* and *stopDrag ( )* handlers enabled the incorporation of

logic that allows the dragging and dropping of the *device blocks* at any positions within the patchbay "work space".

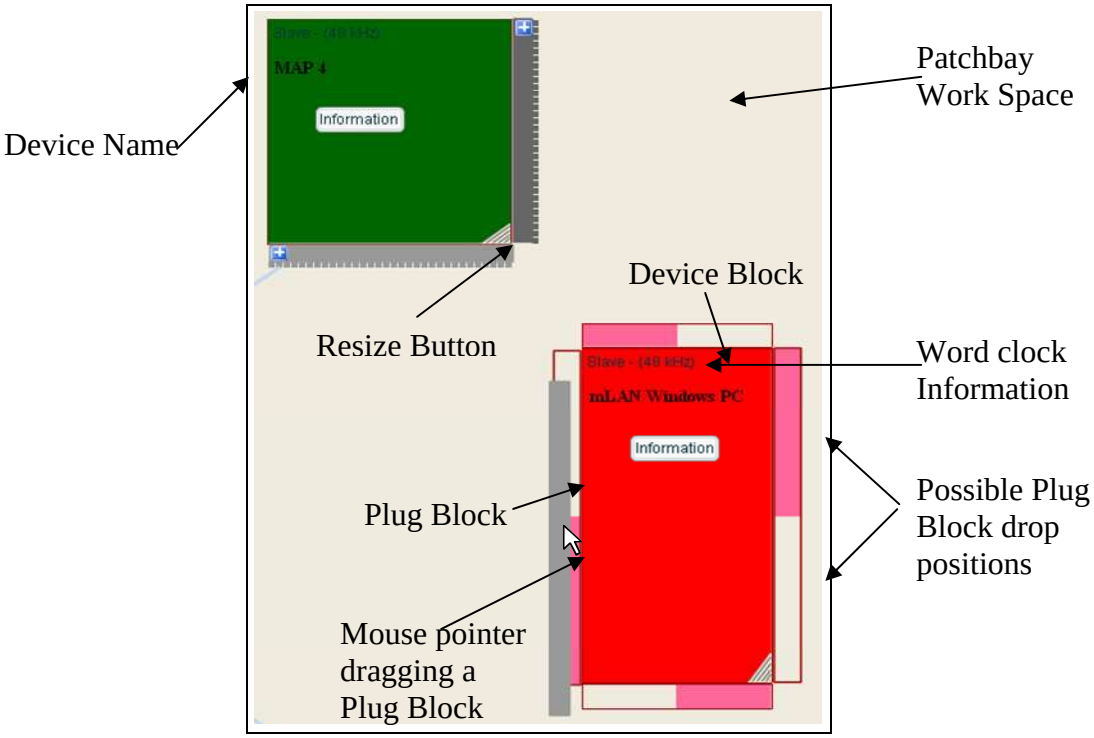

Figure 8.3: The Project Studio Device and Plug

The *device blocks* also incorporate a *Resize* button, which is located at the bottom right corner of each *block* that enables the resizing of both *device* and *plug blocks*. The *Resize* button is a dummy graphic element that shows the user where to click and drag to resize the *device block*. The actual resizing capability is provided by a combination of ActionScript *onMouseOver ( )* and *onMouseMove ( )* handlers, which contain code that stretches the *blocks* as required. The sizing of the *blocks* is calculated by comparing the x and y registration coordinates of the *device block* being resized and the xmouse and ymouse coordinates. The xmouse and ymouse coordinates identify the current position of the mouse pointer on the patchbay "work space". As the mouse moves, the ActionScript *onMouseMove ( )* handler is continuously invoked, which contains the logic that redraws the *block*. Once the user releases the mouse, the *onMouseRelease ( )* handler is invoked. It contains logic that redraws the *plug blocks* to match the size of the main *device block* they are attached to. When the user is resizing the *device block*, the *plug blocks' _visible* attribute is set to "false" so

that the user can not see the plug block and the attached individual plug graphics resized.

Plug blocks are MovieClips just like the *device blocks*, and contain inner individual plug MovieClips. Just like the *device blocks* they can also be dragged and dropped at any of the eight positions around the *device block* they are attached to, using a combination of the x and y coordinates of the *device block* and those of the mouse pointer where the user released the mouse. ActionScript code is used to keep the *device* and *plug blocks* together by restricting the user from dragging and dropping *plug blocks* only to the highlighted locations around the *device block* [*Figure 8.3*]. If the user drops one *plug block* (say an input *plug block*) at a position already occupied by another *plug block* (say an output *plug block*), their positions are randomly reassigned automatically.

Individual plug MovieClips use ActionScript event handlers that enabled the user to implement the functionality for establishing and breaking audio connections. Assuming a user wants to establish a connection between two plugs, and clicks and drags-out from one of the plugs. The *onDragOut ( )* handler is invoked that contains logic which uses the Adobe Flash drawing API to draw a line from the plug to the mouse pointer. The *onDragOut ( )* handler contains the *onMouseMove ( )* handler, which ensures that from the moment the user drags the mouse-out from a plug, the *connector line* is continuously redrawn, following the mouse, until it is released.

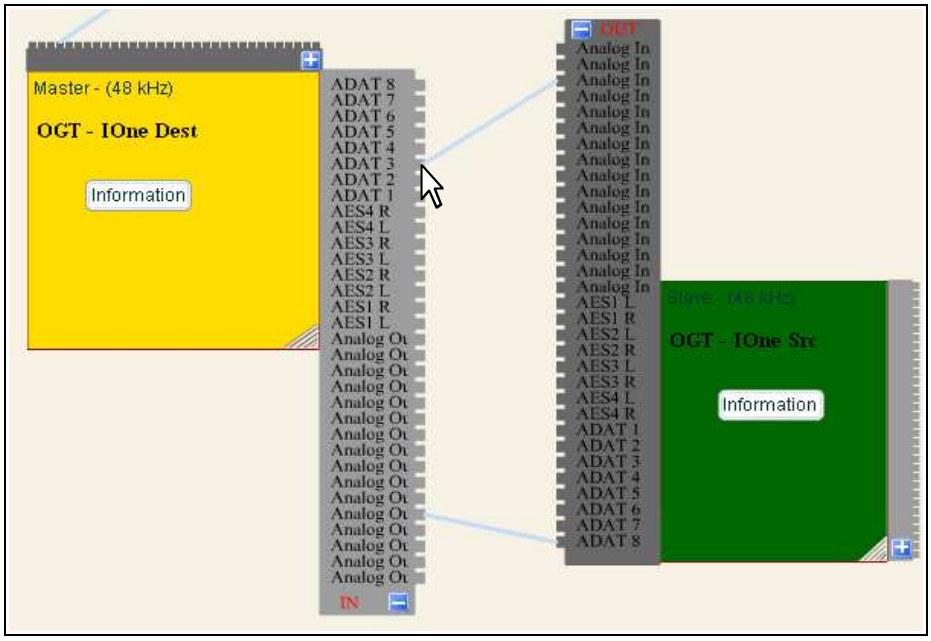

Figure 8.4: Establishing Audio Connections

The *connector line* is line MovieClip which containes methods and event handlers to control the connector line behaviour. When the user releases the mouse on another plug, on a different device, the *onMOuseOver ( )* event handler of that plug is invoked. It contains logic that checks if that plug is not on the same device as the first plug and the *connector line* is redrawn permanently between the two plugs.

One important feature of the Project studio patchbay is the ability of to expand and minimise the *plug blocks* so as to view clearly the names of the plugs when performing connection management tasks. Each *plug block* incorporates a maximise/minimise MovieClip button that contains ActionScript logic for redrawing the *plug block* and its plugs as well as its *connector lines* when the button is clicked to maximise or minimise the block.

The *connector line* MovieClip responds to a mouse drag-out by invoking the *onMouseDragOut ( )* handler that contains code for breaking audio connections between two plugs [*Figure 8.5*].

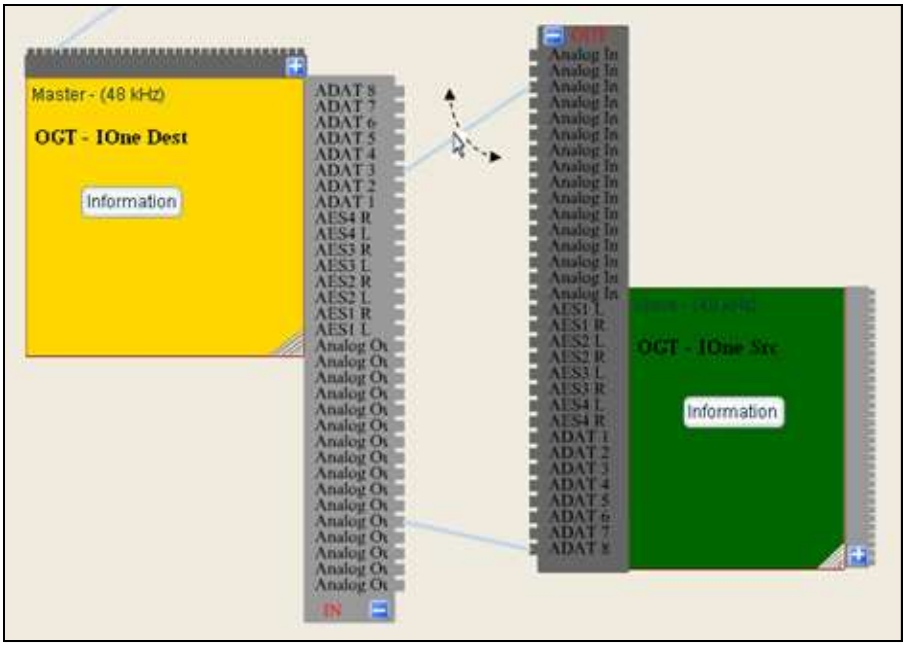

Figure 8.5: Breaking Audio Connections

Like the Broadcast studio patchbay the Project studio patchbay also incorporates the menu and tool bars for quick access to frequently used features.

8.1.2.3 Adobe Flash Graphic Tools and ActionScript Capabilities for Developing the Hospitality/Convention Centre Patchbay

The main parts of the Hospitality/Convention Centre patchbay include; the *device zones* and the *plug icons*, and are implemented as MovieClips [*Figure 8.6*]. Each *device zone* MovieClip has an ActionScript object attached to it that controls what the *zone* can do. The *zones* are simple light grey MovieClips that can be dragged to different positions on the "Detailed Room Display" section and to/from the "Additional Rooms" section of the patchbay. ActionScript handlers are used to do this, as is done for *device blocks* for the Project studio patchbay. MovieClip shadow effects were added to each *device zones* displayed on the "Detailed Room Display" section of the patchbay [*Figure 8.6*]. The shadows were added to individual *zones* so that they seem to be clickable areas within the patchbay interface, thus differentiating them from the bare areas of the interface.

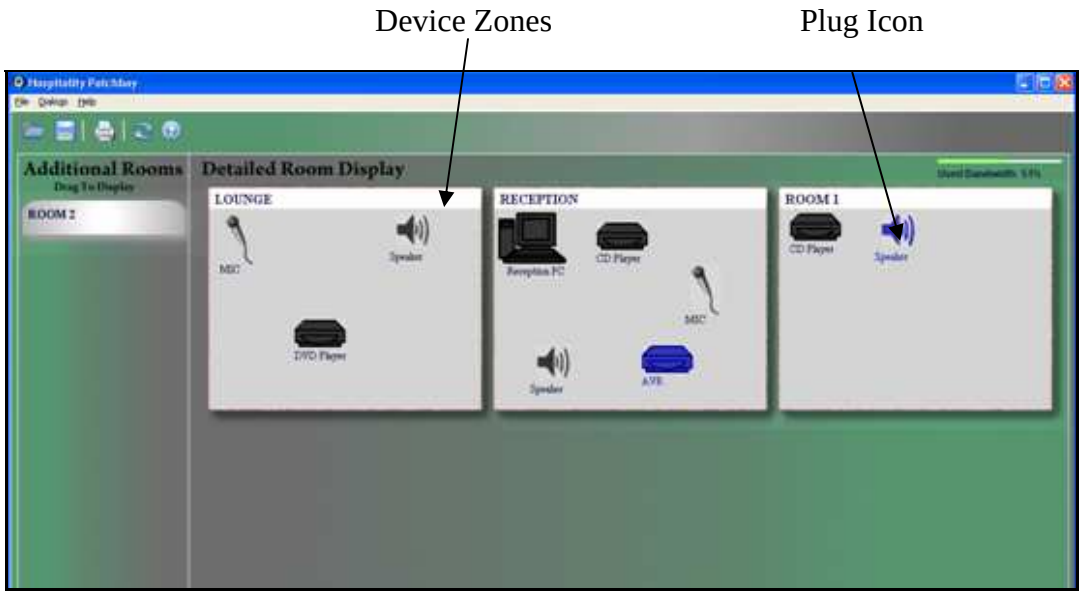

Figure 8.6: Hospitality/Convention Centre Patchbay Interface

Adobe Flash allows the importation and easy manipulation of raster graphics, which enabled the creation of *plug icons* using pictures similar to the actual devices they represent on the network. This fulfilled a requirement of Hospitality/Convention Centre patchbays, namely that network devices be graphically represented on the patchbay. These Hospitality/Convention Centre patchbays are usually operated by individuals who need not see the complexity of the network but be presented with a simple interface that displays devices they are familiar with. *Plug icons* are graphic pictures that are wrapped inside MovieClip symbols. This made it possible to use built-in MovieClip action handlers to control the icons when performing connection management tasks such as establishing and breaking connections. Third-generation language IDEs' will require many lines of code to draw graphic component such as the *device zones* and *plug icons*.

MovieClip symbols were used for creating various patchbay panels, which include:

- The *Control Windows*.
- The *Settings* panel.
- The *Wordclock* panel.
- The *Network Configuration* panel.
- The *Plug Layout* panel.

The ability to just create graphic symbols (using common tools) and to modify and control their behaviour so easily was an advantage of using Adobe Flash. Third-generation language IDEs also allow the creation of graphic components but at a high cost in terms of effort and time required, since these graphic components have to be developed in code.

8.1.3 Adobe Flash Application Portability

Applications developed using Adobe Flash are browser or platform independent. This is because Adobe Flash applications run on the same Flash Player, which is developed to run on any platform. The three patchbays were developed on a Windows workstation and were successfully ported onto an Apple Macintosh machine. No additional coding was required. They just installed and worked properly. All that was required was the correct Apple Macintosh Flash player plug-in to install and run the software on a Apple Macintosh workstation.

8.2 Development of the Patchbay Back-end Components

As already stated in *chapter 2*, the mLAN Connection Management Server (mCMS) communicates with the mLAN clients through a TCP/IP socket using XML messages. This section describes Adobe Flash XML and XMLSocket object methods that were fundamental in creating these XML documents.

8.2.1 Adobe Flash XML Capabilities

One of the most helpful features of the Adobe Flash IDE that was fundamental in the development of the Broadcast patchbay, the Project studio patchbay, and the Hospitality/Convention Centre patchbay was its XML API. Adobe Flash provides a built-in XML class that comprises methods and properties for loading, parsing, sending, building, and manipulating XML document trees. An XML document is represented in Adobe Flash by an XML class. Each element in the XML document is represented by an XMLNode class. The XMLNode class methods and properties that were essential in the development of XML documents include [Adobe Macromedia, 2005]:

- childNodes *:* *Array* - An array of the specified XML object's children.

- firstChild : *XMLNode* - Evaluates the specified XML object and references the first child in the parent node's child list.
- lastChild : *XMLNode* - An XMLNode value that references the last child in the node's child list.
- nextSibling : *XMLNode* - An XMLNode value that references the next sibling in the parent node's child list.
- hasChildNodes ( ) : *Boolean* - Specifies whether or not the XML object has child nodes.
- removeNode ( ) : *Void* - Removes the specified XML object from its parent.

These methods were utilised for creating all XML documents that were required for communication purposes between the patchbays and the mCMS server. The Adobe Flash XML API was easier and simpler to use than other API's. All important XML document building blocks were provided by default in the XML class. All that had to be done, was to join the blocks together using simple ActionScript logic and provided necessary XML element attribute values to create XML documents.

8.2.2 Adobe Flash XMLSocket Class

Adobe Flash incorporates an XMLSocket class that implements client sockets, which allow a computer running Flash Player to communicate with a server computer identified by an IP address or a domain name [Adobe Macromedia, 2005]. The XMLSocket class methods that were useful for connecting to the mCMS server, and sending and receiving XML messages to and from the mCMS server include the following [Adobe Macromedia, 2005]:

- *connect (url:String, port:Number) : Boolean* - Establishes a connection to the specified server using the specified TCP port (must be 1024 or higher), and returns true or false, depending on whether a connection is successfully established.
- *onConnect (XMLSocket.onConnect handler)* - Is invoked by the Flash Player when a connection request initiated through the XMLSocket.connect ( ) has succeeded or failed and handles all responses from the server.

- *send (data:Object) : Void* - Converts the XML object or data specified in the object parameter to a string and transmits it to the server.

The Adobe Flash XMLSocket sends XML messages over a full-duplex TCP/IP stream connection. Combining the Adobe Flash XML API and the XMLSocket methods provides a powerful means of creating and sending (and receiving) XML messages to the mCMS server.

8.3 Adobe Flash ActionScript Limitations

Although the Adobe Flash IDE provided many programming and graphic authoring interfaces, it did not provide some fundamental capabilities that were integral to the development and functioning of mLAN Client/Server patchbays. Some missing functionalities include:

- Low-level methods for directly traversing the File system from the Adobe Flash application. This capability was required for saving routing settings into a text file on the host workstation, as well as opening these files when needed.
- Methods for directly accessing the host workstation registry. This functionality was required by mLAN Client/Server patchbays to store the mCMS server name and port number, and the operator's username and password. This was necessary so that the patchbay did not have to continuously ask the user for these details every time it is started after its initial configuration, but instead could just get the details from the computer's registry.

To deal with these issues, an application called Zinc™ v2.5 was utilised. According to Multimedia Limited (2006), Zinc™ v2.5 is a Rapid Application Development tool that extends Adobe Flash by ensuring the availability of methods and functionality provided by popular application development environments and languages such as C++ and Visual Studio.Net. Zinc™ v2.5 provides easy-to-use methods for saving data into a text file as well as opening the saved files. These methods included:

- mdm.FileSystem.saveFile ("c:\\NetworkAudioData\set1.txt", "My text data") – Saves text data into a text file (named *set1*) located in the c drive, in a folder called *NetworkAudioData*.
- var savedData = mdm.FileSystem.loadFile("c:\\NetworkAudioData\set1.txt") – Loads data in the file *set1* located in the c drive, in a folder called *NetworkAudioData* to the variable *savedData*.
- mdm.FileSystem.deleteFile ("c:\\NetworkAudioData\set1.txt ") – Deletes the file *set1* located in the c drive, in a folder called *NetworkAudioData*.

It also supports a wide array of methods for accessing the computer's registry in a simple way that enabled and empowered Adobe Flash in the development of the three patchbays. These included the:

- mdm.System.Registry.createKey(*keyBranchID:Number, keyName:String, keyDefaultValue:String*)*:Void* – Create a new Key in the System Registry at the location specified.
- mdm.System.Registry.saveString( ) – Writes a STRING value to the System Registry.
- mdm.System.Registry.load(*keyName : String*) *: String* – Loads the data from the specified Key in the System Registry.

8.4 Chapter Summary

Adobe Flash's graphic authoring IDE, and its XML and XMLSocket classes played an integral role in the development of the Broadcast patchbay, the Project studio patchbay, and the Hospitality/Convention Centre patchbay. These programming interfaces provided built-in capabilities for quickly creating:

- Graphic and MovieClip symbols for creating various patchbay panels and icons.
- XML documents, which are used for communication between the patchbays and the mCMS server.
- The TCP/IP socket through which XML messages were passed between the patchbay and the mCMS server.

The next chapter gives the conclusion to the investigation and presents guidelines that were derived after using a high-level graphic tool, Adobe Flash Professional 8, for creating connection management applications for controlling audio routing in different sound installation networks.

CHAPTER 9

9 Conclusion

This investigation set out to explore the possibility of using a high-level graphic tool, Adobe Flash Professional 8, for developing connection management applications (also known as *patchbays*) for high-speed audio networking. *Chapters 2* and *3* give the background information required to understand the context of this investigation. A number of audio networking technologies have been developed that deal with end-to-end connection management in various audio networks. These (audio networking technologies) use a combination of hardware and software protocols to enable true end-to-end connection management by providing the capabilities of routing audio and MIDI data between device plugs using a user-level application that exposes virtual plugs of devices on the network. The Yamaha Corporation's mLAN Digital Network Interface Technology is one of these audio networking technologies that is based on Firewire, which was utilised in this investigation. It was chosen for this investigation because its extensive development was done by the Rhodes Audio Engineering Group in collaboration with Yamaha Japan. As a result, the researcher had cheap and easy access to mLAN resources and expertise. Various architectures have been developed for the mLAN Digital Network Interface Technology, which enables the separation of connection management between the device and the workstation. These include; the mLAN Client/Server architecture and the Enabler/Transporter architecture.

The mLAN Client/Server architecture uses XML messages for communication between the front-end client applications and the back-end mCMS server application. The use of XML messages enables the decoupling of the clients from the underlying Enabler module. As a result, client applications can be developed using any language as long as the application can communicate with the mCMS server using XML messages. This feature made it possible to use Adobe Flash designed for creating mLAN connection management applications for three types of sound installation networks, namely the Broadcast network, the Project studio network, and the Hospitality/Convention Centre network. Adobe Flash is traditionally designed for developing applications for display on web pages. It motivated this investigation to

see how Adobe Flash will copy when used to develop complex Object Oriented applications for real time audio routing.

The most complex patchbay (the Broadcast patchbay) was adequately implemented using Adobe Flash Professional 8, leading to the conclusion that Adobe Flash Professional 8 has indeed reached a mature stage and can be used not only for web graphic design purposes but also for creating highly interactive and complicated Object Oriented applications. Not only did Adobe Flash Professional 8 allow for the quick development of patchbay graphic components, but through its ActionScript, provided a powerful way of writing objects and methods for controlling the behaviour of the graphic components. ActionScript enabled the inclusion of complex functionality that may not be easily implemented using third-generation languages such as C, C++ and C#. Some of these features incorporated into the three patchbays include:

- The Broadcast grid-matrix that was used to display the connection status of device nodes visible on the *Sources* and *Destinations* trees. Each grid-matrix box was implemented as a MovieClip graphic that has an ActionScript MovieClip object attached to it by default, which controls its behaviour when interacted with or the sound engineer scrolls the device trees. The grid was actually created by repeatable duplicating one MovieClip symbol stored in the Adobe Flash Library using a simple ActionScript for loop. There was no need to directly code the graphic and its handlers as is the case with third-generation languages.
- The dragging and dropping capabilities of the *device blocks and plug icons* to any positions within the Project studio patchbay "workspace" required the use of built-in MovieClip action handlers, *startDrag ( )* and *stopDrag ( )*, in which lines of code were written to listen to mouse events.
- ActionScript code was used to maximise or minimise Project studio patchbay *plug blocks* to view detailed information for individual plugs. This feature was provided by using the Adobe Flash's graphic authoring IDE for redrawing the *plug blocks* and *connector lines* depending on whether the user maximised or minimised the block.

- The simple Adobe Flash XML model enabled quick and efficient creation of XML messages. Adobe Flash provides an XML class that contain methods and properties for creating XML messages. To create XML messages, in-built XML object bits were joined together using simple ActionScript code. In third-generation languages many lines of code are required to create XML messages, required far more time than when using Adobe Flash's XML model.

In conclusion, this research showed that Adobe Flash Professional 8 using ActionScript code can be used for developing connection management applications for high-speed audio networking in different sound installation environments. More importantly, Adobe Flash allows for the inclusion of complex GUI features. Adobe Flash was adequately utilised for quickly creating graphic components for the three patchbays, namely the Broadcast patchbay, the Project studio patchbay, and the Hospitality/Convention Centre patchbay. Network requirements for three environments were adequately incorporated into the patchbays far more easily using the Adobe Flash IDE tools and ActionScript.

REFERENCES

[Anderson, 1999]. Anderson, D. *'FireWire System Architecture - IEEE 1394a'*, Addison Wesley, Canada. 1999.

[Axia Audio, 2004]. Axia Audio, *'Professional Networked Audio'*. [Online]. Available: http://www.axiaaudio.com/brochures/axia_9-9-2005_screen.pdf . [Access Date 2005/12/10].

[Axia Audio/TLS Corporation, 2005]. Axia Audio/TLS Corporation, *'Professional Networked Audio'*. [Online]. Available: http://www.axiaaudio.com/brochures/axia_9-2005_screen.pdf. [Access Date 15/11/07].

[Adobe Macromedia, 2005]. Adobe Macromedia. *'Flash 8 Documentation*, [Online]: Available: http://livedocs.adobe.com/flash/8/main/wwhelp/wwhimpl/js/html/wwhelp.htm?href=00000001.html . [Access Date 05/10/07].

[Adobe Systems, 2007]. Adobe Systems, *'Getting Started with Flex 2'*, [Online]: Available: http://livedocs.adobe.com/flex/201/html/wwhelp/wwhimpl/js/html/wwhelp.htm?href=Part1_Intro_005_1.html. [Access Date 12/10/07].

[Aviom Inc, 2007]. Aviom Inc., *'Aviom Pro64 Products'*. [Online]. Available: http://www.aviom.com/pro64_index.cfm. [Access Date 10/09/07].

[Axia Audio, 2007]. Axia Audio, "PathfinderPC Router Control Software User's Guide", [Online]: Available: http://www.softwareauthority.com/axia/Ver4_0/PathFinderPCHelp.pdf. [Access Date 20/09/07].

[Benedek and Miner, 2002]. Benedek J. and Miner T. *'Product Reaction Cards'*, Microsoft Corporation, 1 Microsoft Way, Redmond, WA 98052 [Online]: Available:

http://www.microsoft.com/usability/UEPostings/ProductReactionCards.doc. [Access Date 18/11/07].

[Booch, Rumbaugh and Jacobson, 2005]. Booch G, Rumbaugh J and Jacobson I. *'The Unified Modelling Language User Guide'* 2nd Edition, Addison Wesley Professional, United States of America, May 19, 2005.

[Biamp Systems, 2007]. Biamp Systems, *'Audio Digital Audio Platform'*, Operation Manual. USA, 2007. [Online]: Available: http://www.biamp.com. [Access Date 18/12/07].

[Chin, Diehl and Norman, 1988]. Chin, J.P., Diehl, V.A., Norman, K.L (1988). *"Development of an Instrument Measuring User Satisfaction of the Human-Computer Interface"*.ACM CHI'88 Proceedings, 213-218. [Online]: Available: http://oldwww.acm.org/perlman/question.cgi?form=QUIS. [Access Date 20/10/07].

[Cirrus Logic, 2007]. Peak Audio Inc., *'CobraNet Networked Digital Audio'*. 2007. [Online]. Available: http://www.cobranet.info/en/products/pro/areas/PA106.html. [Access Date 10/09/07].

[Digigram, 2007]. Digigram, *'EtherSound Technology: Overview'*, [Online]. Available: http://www.ethersound.com/technology/overview.php. [Access Date 11/09/07].

[Digigram SA, 2007a]. Digigram SA, *'EtherSound EScontrol'*, [Online]. Available:http://www.digigram.com/products/getinfo.php?prod_key=14150. [Access Date 28/09/07].

[D and R Electronica B. V., 2007]. D and R Electronica B. V., *'CobraNetTM Manager'*. [Online]: Available: http://www.cobranetmanager.com/index.html. [Access Date 28/09/07].

[Fujimori and Foss, 2003]. Fujimori J. and Foss R., *'Convention Paper: A new Connection Management Architecture for the Next Generation of mLAN'*, Audio

Engineering Society: Presented at the 114th Convention, Amsterdam. 2003. pp. 389.

[Fujimori, Foss, Klinkradt and Bangay, 2003]. Fujimori J., Foss R., Klinkradt B. and Bangay S., *'Convention Paper: An mLAN Connection Management Server for Web-Based, Multi-User, Audio Device Patching'*, Audio Engineering Society: Presented at the 115th Convention, New York. 2003.

[Foss, 2005]. Notes compiled by Foss, R. *'Audio Engineering - Computer Science Honours level course notes'*, Rhodes University, Department of Computer Science. 2005.

[Flashloaded, 2007]. Flashloaded (2007). *"AdvancedTree Userguide v 1.0.1"*. [Online]: Available: http://www.flashloaded.com/userguides/advancedtree/ . [Access Date 25/01/08].

[Huang, 2006]. Huang E., *"Introducing Adobe Flash Player 9"*, ADOBE [Online]: Available: http://www.adobe.com/devnet/logged_in/ehuang_flashplayer9.html. [Access Date 20/09/07].

[IEEE Inc, 1998]. Institute of Electrical and Electronics engineers , Inc. '*IEEE Recommended Practice for Software Requirements Specifications*', IEEE Std 830-1998.

[IEC, 2005]. International Electrotechnical Commission, Inc. '*Consumer audio/video equipment – Digital interface – Part 6: Audio and music data transmission protocol'*, 2nd Edition 2005-10.

[IMT Inc, 2005]. Intelligent Media Technologies Inc., '*SmartBuss'*, IMT Inc., 2005. [Online]. Available: http://www.intelligentmedia.us/products/smartbuss/index.php. [Access Date 18/09/07].

[I/One Connects, 2007a]. I/One Connects, '*NAS Explorer Patchbay '*, User Manual, 2007.

[I/One Connects, 2007b]. I/One, *"I/One Firewire Snake"*, [Online]. Available: http://www.ioneconnects.com/. [Access Date 18/09/07].

[Kirakowski, 1994]. Kirakowski, J. *'The Use of Questionnaire Methods for Usability Assessment'*. [Online]: Available: http://sumi.ucc.ie/sumipapp.html. [Access Date 20/12/07].

[Kruchten, 2000]. Kruchten, P. *'The Rational Unified Process - An Introduction'*, 2nd Ed. Addison Wesley, London. 2000.

[Klinkradt, 2004]. Klinkradt, B. *'Draft working document: mLAN Connection Management Server -Client Server Communication 0.0.2 (Yamaha Confidential)'*. Rhodes University, Grahamstown. 2004.

[Lin, Choong and Salvendy, 1997]. Lin, H.X. Choong, Y.-Y., and Salvendy, G. (1997) *"A Proposed Index of Usability: A Method for Comparing the Relative Usability of Different Software Systems"*. [Online]: Available: http://oldwww.acm.org/perlman/question.cgi?form=PUTQ. [Access Date 20/10/07].

[Laszlo Systems, 2007]. Laszlo Systems, *'Software Engineer's Guide to Developing OpenLaszlo Applications'*, [Online]: Available: http://www.openlaszlo.org/lps4/docs/developers/index.html#0. [Access Date 20/11/07].

[Maksimchuk and Naiburg, 2004]. Maksimchuk R.A. and Naiburg E.J. *'UML for Mere Mortals'*, Addison Wesley Professional, United States of America, October 26, 2004.

[Multidmedia Limited, 2006]. Multidmedia Limited. *'Zinc™ v2.5 Help Manual'*, [Online]: Available: http://www.multidmedia.com/support/learning/help/HTML/zinc/2.5/index.html. [Access Date 03/09/07].

[Microsoft Corporation, 2007a]. Microsoft Corporation. 'Silverlight Overview', [Online]: Available: http://msdn2.microsoft.com/en-us/library/bb404708.aspx. [Access Date 15/10/07].

[Microsoft Corporation, 2007b]. Microsoft Corporation. 'Silverlight Overview', [Online]: Available: http://www.microsoft.com/silverlight/why-flexible.aspx. [Access Date 05/11/07].

[Nielsen, 2004]. Nielsen, J. **'***Usability engineering'*. Morgan Kaufmann, San Francisco. 2004.

[Nielsen, 2005]. Nielsen, J. **'***Ten Usability Heuristics***'**, [Online]: Available: http://www.useit.com/papers/heuristic/heuristic_list.html . [Access Date 24/01/08].

[Ousterhout., 1998]. Ousterhout J K., *'Scripting: Higher Level Programming for the 21st Century'*, [Online]: Available: http://home.pacbell.net/ouster. [Access Date 02/10/07].

[OTARI, 2001]. OTARI, *'ND-20 Network Audio Distribution Unit'*, Operation Manual. Edition 1.1.6a7 OTARI, Japan 2001.

[OTARI, 2005]. OTARI, *'mLAN Control Software'*, Operation Manual. Edition 1.1.6a7 OTARI, Japan 2005.

[Okai-Tettey, 2005]. Harold A. Okai-tettey,. *'High Speed End-to-end Connection Management in a Bridged IEEE 1394 Network of Professional Audio Devices'*, PhD Thesis, 2005.

[Pierotti, 1994]. Deniese Pierotti. "Usability Techniques:Heuristic Evaluation - A System Checklist". [Online]: Available: http://www.stcsig.org/usability/topics/articles/he-checklist.html. [Access Date 20/10/07].

[Prechelt, 2002]. Prechelt L., *'Are Scripting Languages Any Good? A Validation of Perl, Python, Rexx, and Tcl against C, C++, and Java '*A chapter for Advances in Computers. August 18, 2002.

[Rational SDC, 1998]. Rational the Software Development Company. '*Rational Unified Process: Best Practices for Software Development Teams*', Rational Software White Paper TP026B, Rev 11/01.1998.

[Sound On Sound Ltd, 1999]. Sound On Sound Ltd, *'Patchbays'***,** [Online]: Available: http://www.soundonsound.com/sos/dec99/articles/patchbay.htm. [Access Date 15/09/07].

[Siegle, 2004]. Del Siegle, *'Likert Scale'*. [Online]: Available: http://www.gifted.uconn.edu/siegle/research/Instrument%20Reliability%20and%20Validity/Likert.html. [Access Date 20/12/07].

[Sun Microsystems, 2007]. Sun Microsystems, *'JavaFX Technology - At a Glance'*, [Online]: Available: http://java.sun.com/javafx/. [Access Date 20/09/07].

[Yamaha Corporation 2004a]. Yamaha Corporation, '*mLAN Graphic Patchbay Owner's Manual'*, [Online]: Available: http://www2.yamaha.co.jp/manual/pdf/emi/english/synth/GraphicPatchbay_en_om_v16a.pdf. [Access Date 15/09/07].

[Yamaha Corporation, 2004b]. Yamaha Corporation, '*Graphic Patchbay User Manual'*, [Online]: Available: http://www2.yamaha.co.jp/manual/pdf/emi/english/synth/graphicpatchbay_e.pdf. [Access Date 20/09/07].

[Yamaha Corporation, 2004c]. Yamaha Corporation, *'mLAN Information: Just What is mLAN Anyway'*. [Online]: Available: http://www.mlancentral.com/mlan_info.php. [Access Date 20/11/07].

APPENDIX A: Software Requirements Specification Documents

A1 Broadcast Patchbay Software Requirements

Specification Document

1. Introduction

1.1 Revision History

Name	Date	Description/Reason For Changes	Version
Phathisile Sibanda	27/2/06	Initial Requirements Specification for a Broadcast Studio Routing Patchbay.	1.0
	10/03/06	Refined Requirements Specification. More elaborate **Stimulus/Response Sequences** for all features.	1.1
	15/06/06	Changed **Stimulus/Response Sequences** for the establishing and breaking audio connections.	1.2
	20/07/06	Product name was changed to reflect the functionality of the application from “Grid-based Patchbay” to “FireGrid” which means the application performs connection management on a grid for firewire networks.	1.2

1.2 Purpose

This Software Requirements Specification (SRS) document explains and gives a brief high level view of what a grid-based patchbay for Broadcast networks should do within a music Local Area Network (mLAN) to enhance connection management between mLAN compatible devices. It also gives a description of the User Interface features of a grid-based patchbay.

1.3 Document Conventions

The format followed in writing this Requirement Specification Document was adopted from the IEEE Standard 830-19998 [IEEE Inc, 1998] for writing requirement specification documents.

Acronym/abbreviation	Description
mLAN	mLAN stands for "music Local Area Network" and defines a network that allows for the transport of audio and music data between audio devices[Fujimori and Foss, 2002].
Firewire	A high-speed serial-bus standard that offers enhanced connectivity and data transfer for video, audio and storage peripheral applications through a universal input/output (I/O) interface. Firewire is also known as 'IEEE 1394 networking technology'.
mCMS	mCMS stands for "mLAN Connection Management Server". This server in responsible for delivering information about the configuration of the mLAN network to the remote clients in the form of XML (Extensible Mark-up Language) documents.
Unit	Refers to an mLAN compatible device (for example the Otari ND – 20B).

1.4 Intended Audience

This Requirements Specification Document is intended for Broadcast patchbay developers to demonstrate their understanding of the system from the users' point of view. It will be used by the system clients and the developer to agree on the functions and capabilities of the system before the development process began.

1.5 Product Scope

The proposed name of the application to be developed is "FireGrid". As its name suggests the application will be incorporated into an existing mLAN Client/Server system for purposes of providing audio routing capabilities for communicating mLAN network devices. In addition to making and breaking connections the application should allow the user to view the network topology of the mLAN network as well as open and edit properties of IEEE 1394 nodes connected to a particular IEEE 1394 bus and set/clear Master/Slave configurations. The application will work as a flexible mLAN network and audio routing management tool that will be installed on XP workstations.

2. Overall Description

2.1 Product Perspective

As already explained in the previous section 1.4, the Graphical User Interface application (Broadcast patchbay) is a component part of a Client/Server system that already exists in the Rhodes Computer Science Engineering Department. It is the client side of the Client/Server system that interacts directly with the user, making it the gateway for the user into the system. Figure 1.1 below shows the current configuration environment of the mLAN Client/Server system on which the complete application will be deployed.

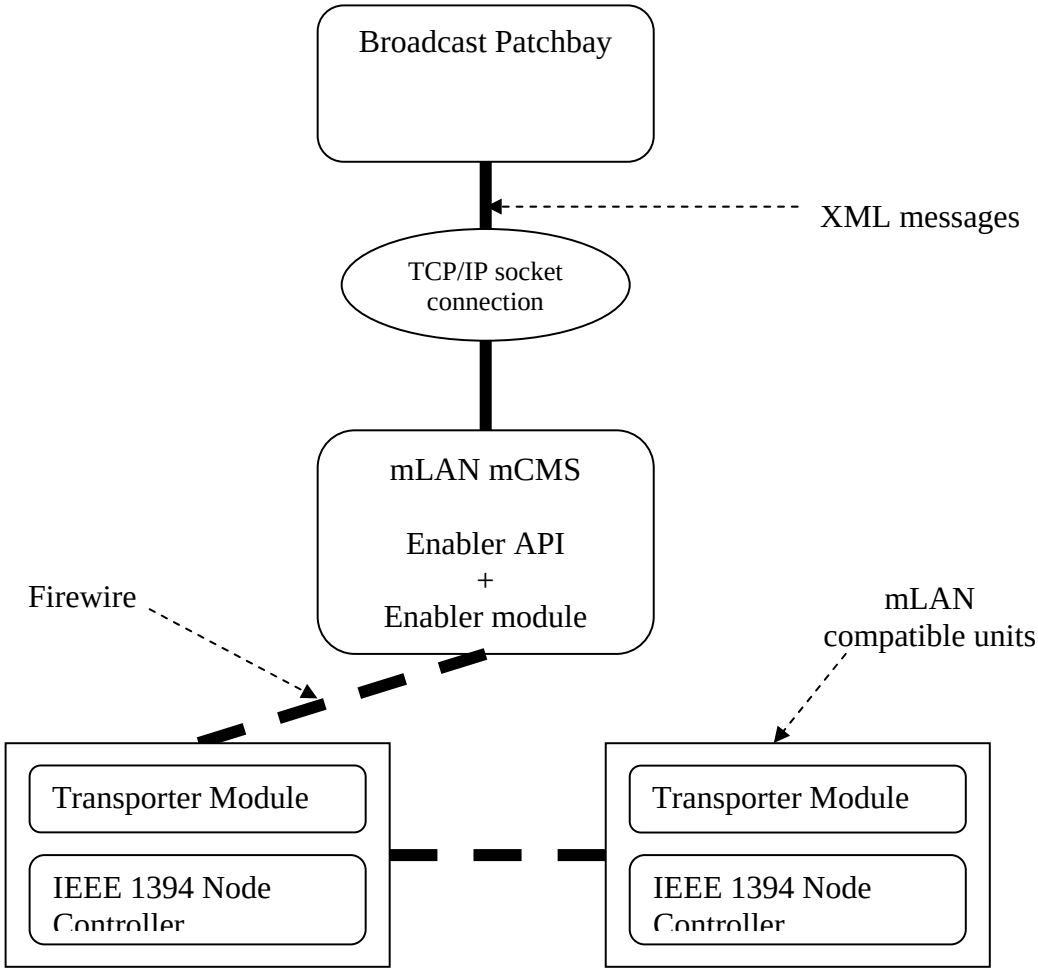

Figure 1.1: mLAN Client/Server Configuration

2.2 Product Functions

The Broadcast patchbay will have two main panels. There will be the Configuration and the Control windows. The Configuration window will allow the user to edit and manage saved file routing setups while the Control window will offer four main functionalities. The user will be able to”

- View the mLAN bus topologies.
- Confirm units connected to a certain mLAN bus on the Audio Pane.
- Perform signal routing and connection management on the Routing Matrix.
- Set Master/Slave configurations on the Clock Setup pane of the Control window.

2.3 User Classes and Characteristics

The patchbay to be developed will mainly be used by Broadcast studio operators. It is planed that the product will provide two user levels; the administrator and general users. Administrator will generally have the ability to manage user accounts and general maintenance of the system

2.3 Design and Implementation Constraints

No design and implementation constraints were identified at the product conception.

2.4 User Documentation

Documentation to be released with the product includes user manuals, requirement documents, use-case diagrams as well as Object models.

2.5 Assumptions and Dependencies

One assumption will be adhered to in the first release of the application, namely that the application is designed for a general user. A unified application will be developed at a later stage that will incorporate two user levels, the administrator and a general operator.

3. External Interface Requirements

3.1 User Interfaces

The user interface will be a graphical grid in nature with associated tabs and panes clearly labelled to reflect their functions. Within each pane there will be buttons for different actions that the user can execute once the panel is opened.

3.2 Hardware Interfaces

Figure 1.2 below shows how the graphical user interface will communicate with the server. The patchbay forms the client side of the system and communicates with the mLAN Connection Management Server (mCMS) through a TCP/IP socket. Any communication medium (wireless or wired) can be used for communication as long as the server and the client can communicate over the Internet Protocol. Extensible Mark-up Language (XML) documents will be used to store and pass information regarding devices, plugs and plug connections between the server and clients [Fujimori, Foss, Klinkradt and Bangay, 2003].

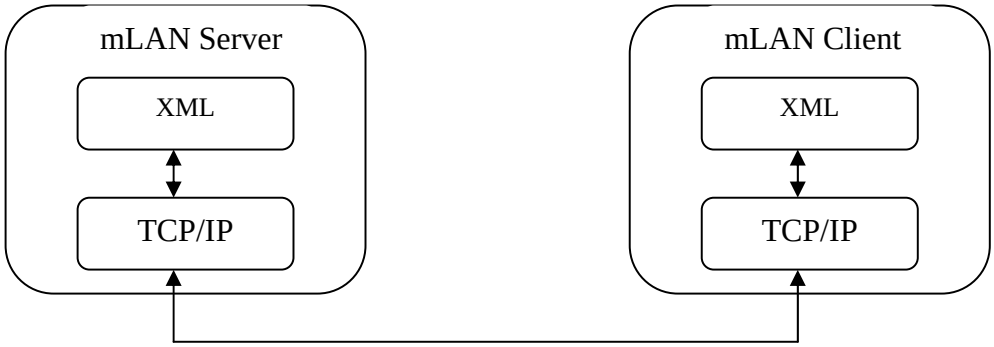

Figure 1.2: Client/Server Communication Model

3.3 Software Interfaces

The user interface will only communicate with the Request Server application in the mCMS. This will in turn talk to internal server software like the Enabler API for the extraction of information from the IEEE bus. The section below elaborates on the communication interfaces involved in the system communication channels.

3.4 Communication Interfaces

Figure 1.3 below shows how the Patchbay will communicate with the mCMS. Three communication components/interfaces within the Client/Server system that will aid communication between the Client and Server applications comprise the Patchbay, the Request server, and the Device interface as shown in *Figure 1.3*. The client is responsible for handling the users' requests. The Request server is the middle platform that deals with and forwards the users requests and responses to and from the devices interface via a TCP/IP socket. The Devices interface uses the Enabler API to get the Enabler module to solve user requests by accessing the IEEE 1394 Bus for answers. XML documents are used to store and communicate information between the Client and the Server modules.

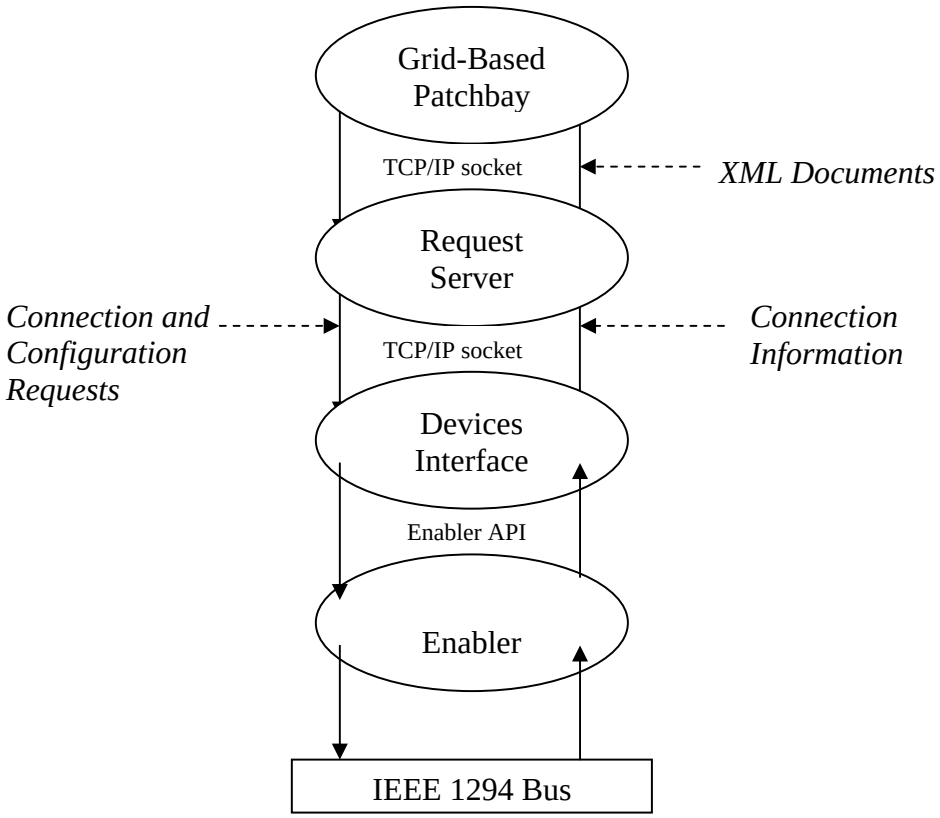

Figure 1.3: mLAN Client/Server Components

4. Functional Requirements: System Features

This section describes the features found on the Patchbay application to be designed and their associated actions and interaction with the Client/Server system and the user. Note: The system features described in this initial requirement specification pertain mainly to a general user. A second version of this specification will be released that recognizes two distinct users, the administrator and a general.

The Use Case diagram [*Figure 1.4*] gives the general feature overview of the Broadcast patchbay.

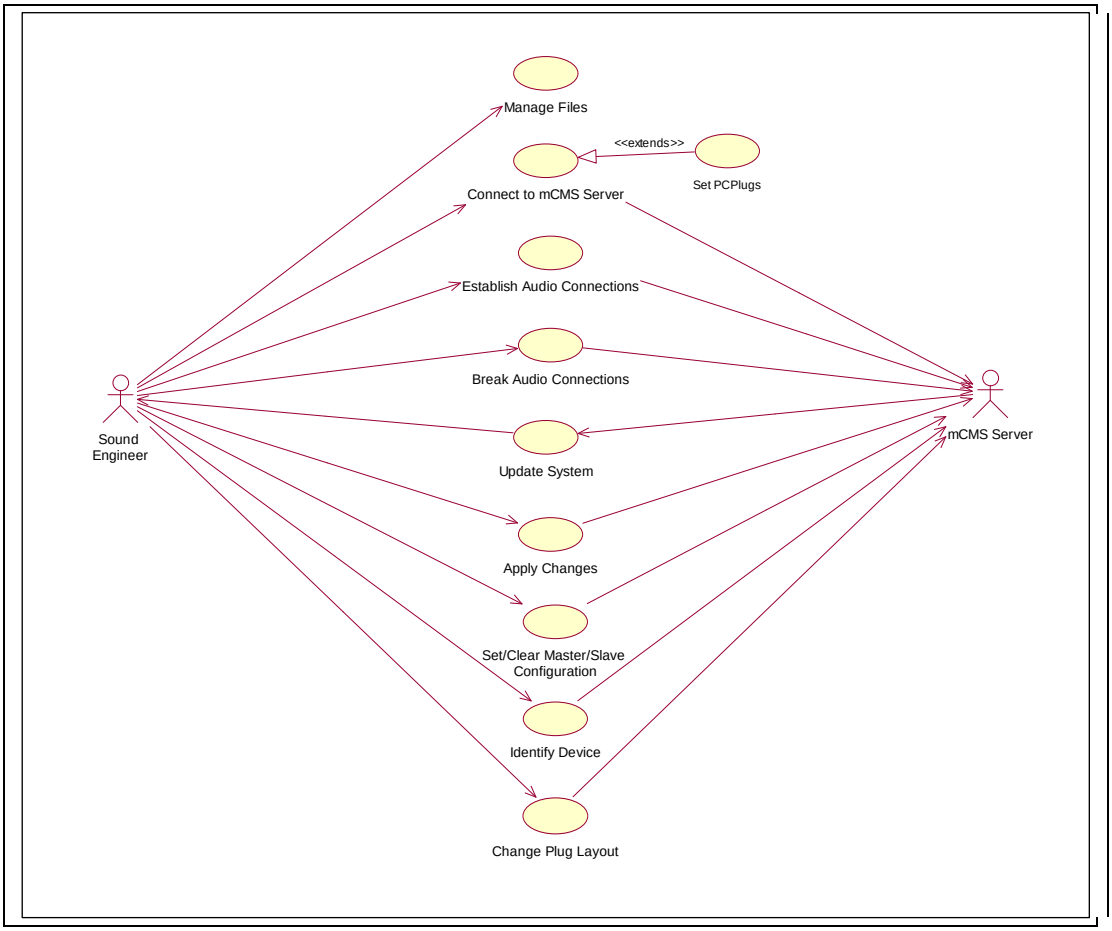

Figure 1.4 Broadcast Patchbay Use Case Diagram

a) Managing Files (Saving and Opening saved routing settings to/from a file)

Table 4.1: "Manage Files" Use Case Stimulus/Response Sequences

Heading	***Description***
Use Case Name	Manage Files
Actors	sound engineer, mCMS Server
Use Case Summary	• The sound engineer uses the system to save audio routing settings into a .nsl file on the host workstation. • The sound engineer uses the system to open saved audio routing settings.
Pre-Condition	• System information can only be saved into a file with the .nsl extension. • The sound engineer can save audio routing settings if there is enough free space in the host workstation hard disk. • The sound engineer can open .nsl file if there are already saved audio routing settings.
Description/Main Flow	• To save routing setting into a .nsl file, the sound engineer clicks the "File" menu.

Stimulus/Response Sequences	• The System displays a sub menu with four menu items (“Open”, “Save”, “Print” , “Exit”)and , • The sound engineer clicks the “Save” menu item and specifies the file name in the dialog box that appears and click the “Save” button to save the file to the workstation. • To open saved routing setting, the sound engineer clicks the “File” menu and select the “Open” menu item on the sub menu that appears. • The system displays the “Open File” dialog box • The sound engineer navigates the file system to where the file to be opened is saved and selects it and clicks the “Open” button.
Post-Condition	The sound engineer successfully manages and manipulates .nsl files on the local host workstation.
Alternate Flow(s)	The sound engineer can use the “Save” and “Open” file icons for access the save to a file and open file dialog boxes.
Exceptions	
Used Use Cases	**Update System, Apply Changes**

b) Establishing a Connection to mCMS Server

Table 4.2: “Connect to mCMS Server” Use Case Stimulus/Response Sequences

Heading	***Description***
Use Case Name	Connect to mCMS Server
Actors	sound engineer, mCMS Server
Use Case Summary	The system connects to the mCMS server.
Pre-Condition	To connect to the mCMS server, correct connection and authentication information should be provided by the sound engineer. Required information include: • The server name, server port number • The operator’s username and password
Description/Main Flow **Stimulus/Response Sequences**	• The sound engineer starts the system. • On start-up the system presents the sound engineer with two dialog boxes, the Login and Server Settings dialog boxes. • The sound engineer inputs the username and password, server name and server port number if this is the first time the system is started after installation. • The system uses the server name and server port number to connect to the mCMS server and authenticate the username and password. • If the authentication process is successful, the mCMS server sends a configuration XML document to the system. • On receiving the configuration XML document, the system is updated appropriately.
Post-Condition	• The system successfully connected to the mCMS server. • The system is updated using configuration from the

	mCMS server.
Alternate Flow(s)	
Exceptions	• If the server is not found or the server name and port number are incorrect. • If the username and password authentication fails.
Used Use Cases	**Update System**

c) Establishing Audio Connections

Table 4.3: "Establish Audio Connections" Use Case Stimulus/Response Sequences

Heading	***Description***
Use Case Name	Establish Audio Connections
Actors	sound engineer, mCMS Server
Use Case Summary	The sound engineer establishes audio connections on the system based on the Mode set, either Immediate or "Delayed Mode".
Pre-Condition	• The sound engineer sets the Mode to be used, either Immediate or "Delayed Mode". • The sound engineer ensures that Master/Slave Configuration is set. • The sound engineer identifies and selects two plugs to be connected. • Plugs to be connected should be on different devices since internal routing is not allowed.
Description/Main Flow **Stimulus/Response Sequences**	• The sound engineer identifies the plugs to be connected and their cross-point on the grid-matrix. • The sound engineer clicks the cross-point of the two plugs to be connected. • The system, based on the set mode, implements the connection.
Post-Condition	The sound engineer successfully established audio connection(s) and audio stated routing to the designated destination plugs.
Alternate Flow(s)	
Exceptions	If there is not enough bandwidth on the mLAN network, the request will fail with an error message.
Used Use Cases	**Apply Changes, Set/Clear Master/Slave Configuration**

d) Breaking Audio Connections

Table 4.4: "Break Audio Connections" Use Case Stimulus/Response Sequences

Heading	*Description*
Use Case Name	Break Audio Connections
Actors	sound engineer, mCMS Server
Use Case Summary	The sound engineer breaks audio connections on the system based on the Mode set, either Immediate or Delayed.
Pre-Condition	The sound engineer identifies and selects two plugs to be disconnected.
Description/Main Flow **Stimulus/Response Sequences**	• The sound engineer identifies the plugs to be disconnected and their cross-point on the grid-matrix. • The sound engineer clicks the cross-point of the two plugs to be disconnected. • The system, based on the set mode, implements the disconnection.
Post-Condition	The sound engineer successfully disconnected two plugs and audio stopped routing.
Alternate Flow(s)	
Exceptions	
Used Use Cases	**Apply Changes**

e) Updating the System

Table 4.5: "Updating" Use Case Description

Heading	*Description*
Use Case Name	**Update System**
Actors	sound engineer, mCMS Server
Use Case Summary	The sound engineer forces system update using latest information from the mCMS Server
Pre-Condition	
Description/Main Flow **Stimulus/Response Sequences**	• The sound engineer clicks the Update button. • The system creates and sends an Update XML document to the mCMS server. • In response the mCMS server sends a configuration XML document to the system. The configuration XML document carries current mLAN network information. • On receiving the configuration XML document, the system updates the device trees and the grid matrix appropriately.
Post-Condition	The sound engineer successfully updated the system with latest information provided by the mCMS server.
Used Use Cases	

f) Appling Patchbay Changes to the Network

Table 4.6: "Apply Changes" Use Case Stimulus/Response Sequences

Heading	*Description*
Use Case Name	Apply Changes
Actors	sound engineer, mCMS Server
Use Case Summary	The sound engineer applies changes made on the system to the physical mLAN network. Example changes include changed Master/Slave synchronisation settings, new connections and disconnections.
Pre-Condition	
Description/Main Flow **Stimulus/Response Sequences**	• The sound engineer clicks the Apply button. • The system loops through the device trees checking for pending connections and disconnections. If they are found the system follows the **Establish Audio Connections** and **Break Audio Connections** use case to commit the found connections and disconnections to the mLAN network.
Post-Condition	The system applied the changes to the physical mLAN network.
Alternate Flow(s)	
Exceptions	If there is not enough bandwidth on the mLAN network, the request will fail with an error message.
Used Use Cases	**Establish Audio Connections , Break Audio Connections**

g) Setting/Clearing Master/Slave Configurations

Table 4.7: "Set/Clear Master/Slave Configuration" Use Case Stimulus/Response Sequences

Heading	*Description*
Use Case Name	Set/Clear Master/Slave Configuration
Actors	sound engineer, mCMS Server
Use Case Summary	• The sound engineer sets/clears Master/Slave Configuration on the system.
Pre-Condition	
Description/Main Flow **Stimulus/Response Sequences**	Setting/Clearing Master/Slave Configurations can be done in many ways on the Wordclock panel.
Post-Condition	The sound engineer successfully changed/set Master/Slave Configurations.
Alternate Flow(s)	
Exceptions	If there is not enough bandwidth on the mLAN network, the request will fail with an error message.
Used Use Cases	**Update System, Apply Changes**

h) Identifying a Device on the Network

Table 4.8: "Identify Device" Use Case Stimulus/Response Sequences

Heading	*Description*
Use Case Name	Identify Device
Actors	sound engineer, mCMS Server
Use Case Summary	In the event of a large network with many devices, the sound engineer can identify a particular device on the system.
Pre-Condition	
Description/Main Flow **Stimulus/Response Sequences**	• The sound engineer right-clicks a particular node either on the Destinations or Source tree on the system. • The system presents to the user a submenu with the menu items that include, Rename Device, Clear All Device Connections, More Device Info, Change Plug Layout and Identify Device. • The sound engineer selects the "Identify Device" menu item. • The system creates and sends a identify device XML message to the mCMS server.
Post-Condition	The sound engineer successfully identified a particular device on the mLAN network through the system. The Light Emitting Diodes of the device identified flashes.

i) Changing the Plug Layout of a Device

Table 4.9: "Change Plug Layout" Use Case Stimulus/Response Sequences

Heading	*Description*
Use Case Name	Change Plug Layout
Actors	sound engineer, mCMS Server
Use Case Summary	The sound engineer changes Plug Layout for any device on the mLAN network on the system.
Pre-Condition	The sound engineer selects the Plug Layout from the list of Plug Layouts of the device whose Plug Layout is to be changed.
Description/Main Flow **Stimulus/Response Sequences**	• The sound engineer right-clicks a particular node either on the Destinations or Source tree on the system. • The system presents to the user a submenu with the menu items that include, Rename Device, Clear All Device Connections, More Device Info, Change Plug Layout and Identify Device. • The sound engineer selects the "Change Plug Layout" menu item. • The system pops up a dialog box showing all available Plug Layouts for the chosen device. • The sound engineer chooses the required Plug layout and clicks the OK button. • The system creates and sends a Plug Layout XML message to the mCMS server. • If the request is implemented successfully, the mCMS server sends the latest configuration information to the system with the appropriate number of plugs for the device whose Plug Layout was changed. • The system updates display.
Post-Condition	The sound engineer successfully changed the Plug Layout of a particular device on the mLAN network through the system. The system is updated accordingly.
Alternate Flow(s)	
Exceptions	
Used Use Cases	**Update System**

Clearing Dangling Connections

a) Description and Priority

A dangling connection is a destination-less routing left only at the sending or receiving side of a communicating device. Plugs with dangling connections are reflected on the Broadcast patchbay tree by blue round icons.

b) Stimulus/Response Sequences

- To clear the blue dangling connection icon of plug, the user clicks the plug icon to be cleared.
- The system changes the blue dangling connection icon colour to yellow.
- The system sends to the server an XML “clear dangling connection” request.

A2 Project Studio Patchbay Software Requirements Specification Document

1. Introduction

1.1 Revision History

Name	Date	Description/Reason For Changes	Version
Phathisile Sibanda	15/01/07	Initial Requirements Specification for the Graphic Patchbay.	1.0
	25/01/07	Additions: • Dragging *device block* • Dragging *plug block* to any of the four sides of the main device block.	1.1
	01/02/07	Eliminated: Hiding/Showing Output and/or Input plug connectors for a particular device	1.2
	05/04/07	Additions: • Making a connection • Deleting a connection	1.3
	05/04/07	Updated: Drag *plug block* to any of the four sides of the main device block.	1.4
	10/07/07	Product named “FireSwitch” patchbay.	1.4

1.2 Purpose

This Software Requirements Specification (SRS) document seeks to explain and give a brief high level view of what a Project studio patchbay should do within a music Local Area Network (mLAN) network to enhance connection management between mLAN compatible devices. Also given is a description of the user interface features of the Project studio patchbay. Functional and non-functional requirements for the Project studio patchbay are also presented.

1.3 Document Conventions

See *Appendix A-A1: section 1.3*

1.4 Intended Audience

This Requirements Specification Document is intended for Project studio patchbay developers to demonstrate their understanding of the system from the users' point of view. It will be used by the system clients and the developer to agree on the functions and capabilities of the system before the development process began.

1.5 Product Scope

The proposed name of the application to be developed is "FireSwitch". As its name suggests the application will be incorporated into an existing mLAN Client/Server system for purposes of providing audio routing capabilities for communicating mLAN network devices. In addition to making and breaking connections the application should allow the user to view the network topology of the mLAN network as well as open and edit properties of IEEE 1394 nodes connected to a particular IEEE 1394 bus and set/clear Master/Slave configurations. The application will work as a flexible mLAN network and audio routing management tool that will be installed on XP workstations.

2. Overall Description

See *Appendix A-A1: section 2*

3. External Interface Requirements

See *Appendix A-A1: section 3*

4. Functional Requirements: System Features

This section describes the features of the Project studio patchbay to be designed and their associated actions and interactions with the Client/Server system, and the user. The Use Case diagram [*Figure 1.1*] gives the general feature overview of the Broadcast patchbay.

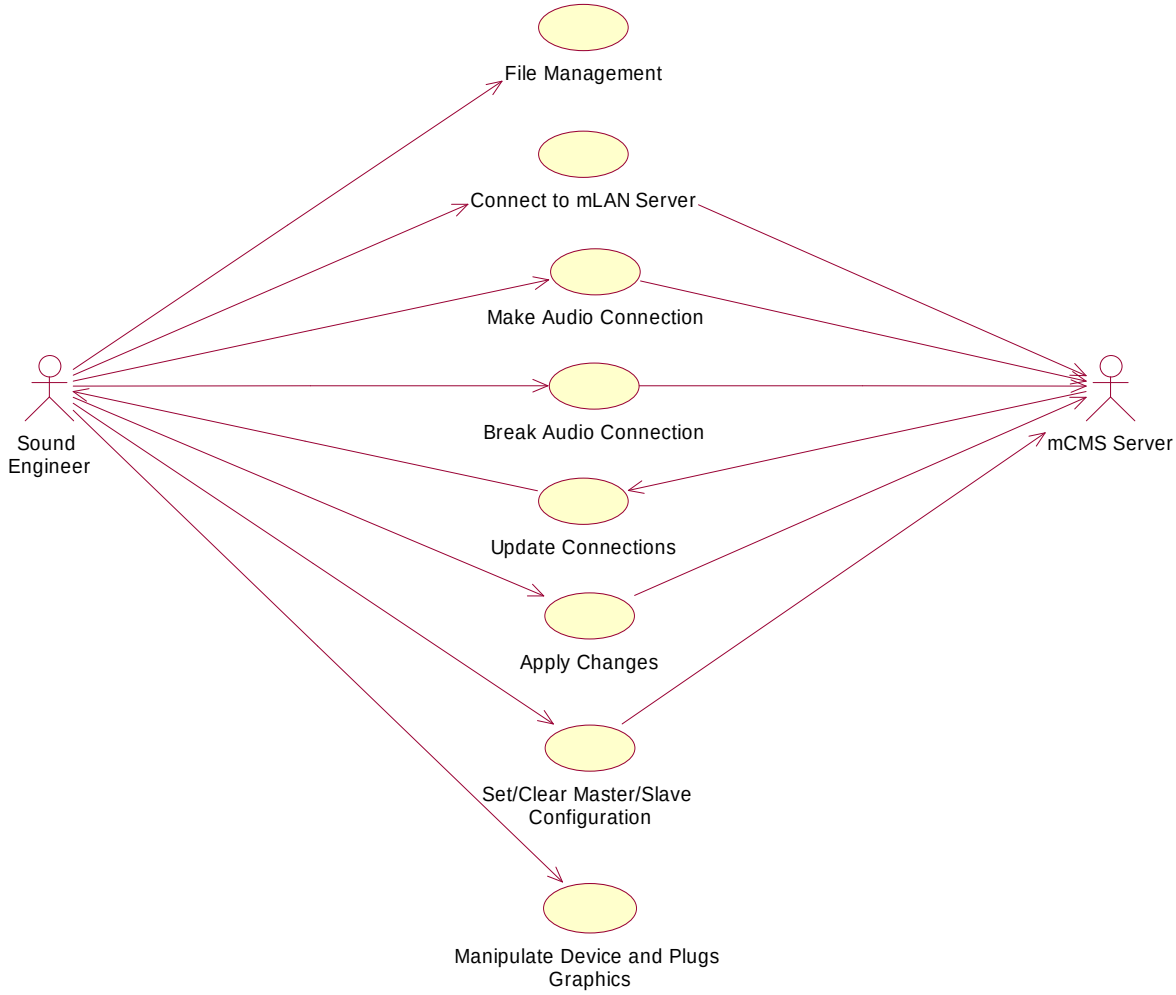

Figure 1.1 Project Studio Patchbay Use Case Diagram

a) Managing Files (Saving and Opening saved routing settings to/from a file)

See *Appendix A-A1: section 4 (Table 4.1).*

b) Establishing a Connection to mCMS Server

See *Appendix A-A1: section 4 (Table 4.2).*

c) Establishing Audio Connections

Table 4.3: "Establish Audio Connections" Use Case Stimulus/Response Sequences

Heading	*Description*
Use Case Name	Establish Audio Connections
Actors	Sound engineer, mCMS Server
Use Case Summary	The sound engineer establishes audio connections on the system based on the Mode set, either Immediate or "Delayed Mode".
Pre-Condition	
Description/Main Flow **Stimulus/Response Sequences**	• The sound engineer ensures that Master/Slave Configuration is set. • The sound engineer maximises the input and output *plug blocks* to the two devices to be connected. • The sound engineer clicks and drags the mouse out from one of the plug graphics to be connected. • The system draws a connector line to the mouse. • The sound engineer releases the mouse over another plug to connect to. • The system sends an XML "connection" request message to the mCMS server and draws the a *connector line* between the two plugs.
Post-Condition	The sound engineer successfully established audio connection(s) and audio stated routing to the designated destination plugs.
Alternate Flow(s)	
Exceptions	If there is not enough bandwidth on the mLAN network, the request will fail with an error message.
Used Use Cases	**Apply Changes, Set/Clear Master/Slave Configuration**

d) Breaking Audio Connections

Table 4.4: "Break Audio Connections" Use Case Stimulus/Response Sequences

Heading	*Description*
Use Case Name	Break Audio Connections
Actors	sound engineer, mCMS Server
Use Case Summary	The sound engineer breaks audio connections on the system based on the Mode set, either Immediate or Delayed.
Pre-Condition	The sound engineer identifies and selects two plugs to be disconnected.
Description/Main Flow **Stimulus/Response Sequences**	• The sound engineer clicks and drags out the *connector line*. Identifies the plugs to be disconnected and their cross-point on the grid-matrix. • The system deletes the *connector line* and sends an XML "disconnect" request message to the mCMS server.
Post-Condition	The sound engineer successfully disconnected two plugs and audio stopped routing.
Alternate Flow(s)	
Exceptions	
Used Use Cases	**Apply Changes**

e) Updating the System

Table 4.5: "Updating" Use Case Description

Heading	*Description*
Use Case Name	**Update System**
Actors	sound engineer, mCMS Server
Use Case Summary	The sound engineer forces system update using latest information from the mCMS Server
Pre-Condition	
Description/Main Flow **Stimulus/Response Sequences**	• The sound engineer clicks the Update button. • The system creates and sends an Update XML document to the mCMS server. • In response the mCMS server sends a configuration XML document to the system. The configuration XML document carries current mLAN network information. • On receiving the configuration XML document, the system updates the device trees and the grid matrix appropriately.
Post-Condition	The sound engineer successfully updated the system with latest information provided by the mCMS server.
Alternate Flow(s)	

f) Appling Patchbay Changes to the Network

See *Appendix A-A1: section 4 (Table 4.6).*

g) Setting/Clearing Master/Slave Configurations

See *Appendix A-A1: section 4 (Table 4.7).*

g) Identifying a Device on the Network

Table 4.8: "Identify Device" Use Case Stimulus/Response Sequences

Heading	***Description***
Use Case Name	Identify Device
Actors	sound engineer, mCMS Server
Use Case Summary	In the event of a large network with many devices, the sound engineer can identify a particular device on the system.
Pre-Condition	
Description/Main Flow **Stimulus/Response Sequences**	• The sound engineer right-clicks the *device block* for the device to be identified, and selects the "Identify Device" menu item. • The system creates and sends a identify device XML message to the mCMS server for implementation.
Post-Condition	The sound engineer successfully identified a particular device on the mLAN network through the system. The Light Emitting Diodes of the device identified flashes.
Alternate Flow(s)	
Exceptions	
Used Use Cases	

h) Changing the Plug Layout of a Device

Table 4.9: "Change Plug Layout" Use Case Stimulus/Response Sequences

Heading	*Description*
Use Case Name	Change Plug Layout
Actors	sound engineer, mCMS Server
Use Case Summary	The sound engineer changes Plug Layout for any device on the mLAN network on the system.
Pre-Condition	The sound engineer selects the Plug Layout from the list of Plug Layouts of the device whose Plug Layout is to be changed.
Description/Main Flow **Stimulus/Response Sequences**	• The user opens the Device Information panel by clicking the "information" button at the centre of each *device block*. • The system opens the Device Information panel. • In the Device Information panel, the sound engineer selects the new Plug Layout from the "Plug Layout" combo box and clicks the *Ok* button. • The system sends to the mCMS server the Plug Layout XML document to implement the request.
Post-Condition	The sound engineer successfully changed the Plug Layout of a particular device on the mLAN network through the system. The system is updated accordingly.
Alternate Flow(s)	
Exceptions	
Used Use Cases	**Update System**

A3 Hospitality/Convention Centre Patchbay Software Requirements Specification Document

1. Introduction

1.1 Revision History

Name	Date	Description/Reason For Changes	Version
Phathisile Sibanda	05/04/07	Initial Requirements Specification for the Hospitality Patchbay.	1.0
	10/07/07	Product named “FireZones” patchbay	1.1

1.2 Purpose

This Software Requirements Specification (SRS) document seeks to explain and give a brief high level view of what a Hospitality/Convention Centre patchbay should do within a music Local Area Network (mLAN) network to enhance connection management between mLAN compatible devices. Also given is a description of the user interface features of the Hospitality/Convention Centre patchbay. Functional and non-functional requirements for the Hospitality/Convention Centre patchbay are also presented.

1.3 Document Conventions

See *Appendix A-A1: section 1.3.*

1.4 Intended Audience

This Requirements Specification Document is intended for Hospitality/Convention Centre patchbay developers to demonstrate their understanding of the system from the users’ point of view. It will be used by the system clients and the developer to agree on the functions and capabilities of the system before the development process began.

1.5 Product Scope

The proposed name of the application to be developed is “FireSwitch”. As its name suggests the application will be incorporated into an existing mLAN Client/Server system for purposes of providing audio routing capabilities for communicating mLAN network devices. In addition to making and breaking connections the application should allow the user to view the network topology of the mLAN network as well as open and edit properties of IEEE 1394 nodes connected to a particular IEEE 1394 bus and set/clear Master/Slave configurations. The application will work

as a flexible mLAN network and audio routing management tool that will be installed on XP workstations.

2. Overall Description

See *Appendix A-A1: section 2.*

3. External Interface Requirements

See *Appendix A-A1: section 3.*

4. Functional Requirements: System Features

This section describes the features found on the Hospitality/Convention Centre patchbay to be designed and their associated actions and interaction with the Client/Server system and the user. Note: The system features described in this initial requirement specification pertain mainly to a general user. A second version of this specification will be released that recognizes two distinct users, the administrator and a general.

The Use Case diagram [*Figure 1.1*] gives the general feature overview of the Broadcast patchbay.

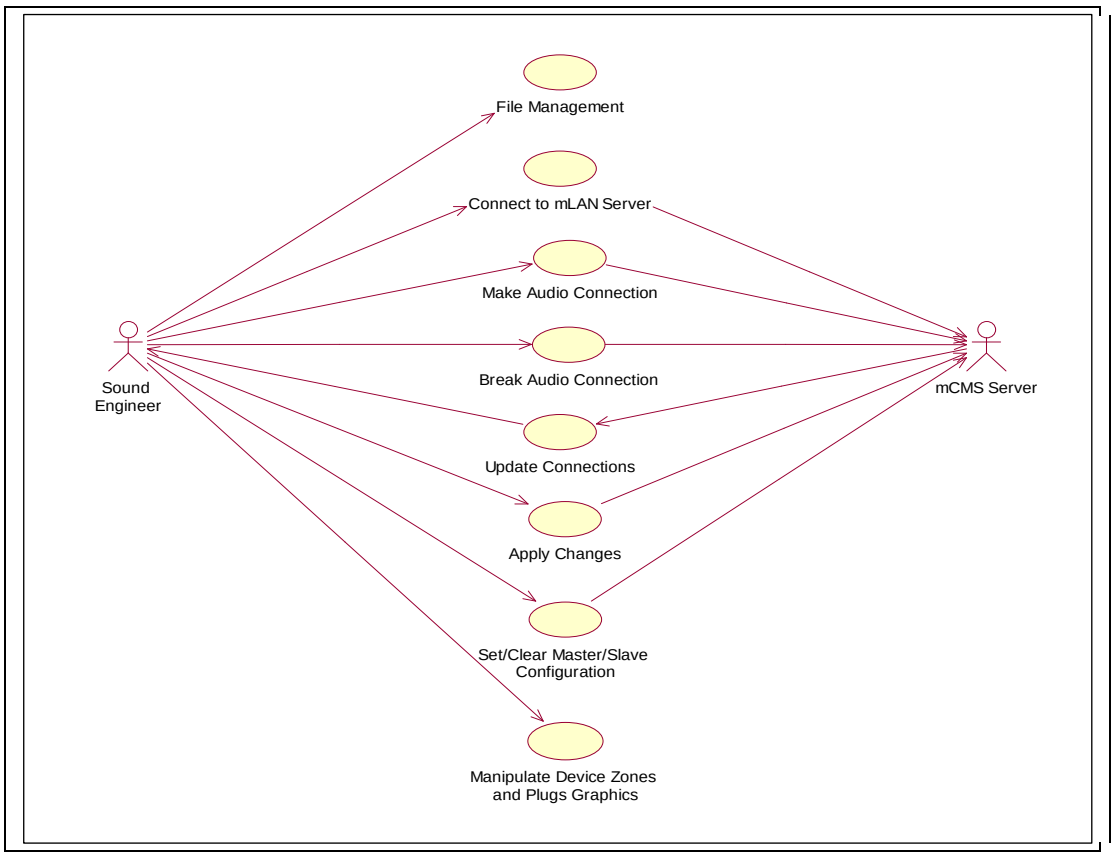

Figure 1.1 Broadcast Patchbay Use Case Diagram

a) Managing Files (Saving and Opening saved routing settings to/from a file)

See *Appendix A-A1: section 4 (Table 4.1).*

b) Establishing a Connection to mCMS Server

See *Appendix A-A1: section 4 (Table 4.2).*

c) Establishing Audio Connections

Table 4.3: "Establish Audio Connections" Use Case Stimulus/Response Sequences

Heading	***Description***
Use Case Name	Establish Audio Connections
Actors	sound engineer, mCMS Server
Use Case Summary	The sound engineer establishes audio connections.
Pre-Condition	
Description/Main Flow **Stimulus/Response Sequences**	• The sound engineer clicks the two *plug icons* of the devices to be connected. • The system gets plug information of the plugs too be connected, creates and sends a connection request XML document to the server to implement the request.
Post-Condition	The sound engineer successfully established audio connection(s) and audio stated routing to the designated destination plugs.
Alternate Flow(s)	
Exceptions	If there is not enough bandwidth on the mLAN network, the request will fail with an error message.
Used Use Cases	**Apply Changes, Set/Clear Master/Slave Configuration**

d) Breaking Audio Connections

Table 4.4: "Break Audio Connections" Use Case Stimulus/Response Sequences

Heading	*Description*
Use Case Name	Break Audio Connections
Actors	sound engineer, mCMS Server
Use Case Summary	The sound engineer breaks audio connections on the system
Pre-Condition	The sound engineer identifies and selects two plugs to be disconnected.
Description/Main Flow **Stimulus/Response Sequences**	• The sound engineer identifies the plugs to be disconnected. • The sound engineer right-clicks the destination *plug icon* and select the "Disconnect" menu item. • A "disconnect" request document is sent to the server to implement the request.
Post-Condition	The sound engineer successfully disconnected two plugs and audio stopped routing.
Alternate Flow(s)	
Exceptions	
Used Use Cases	**Apply Changes**

e) Updating the System

Table 4.5: "Establish Audio Connections" Use Case Description

Heading	*Description*
Use Case Name	**Update System**
Actors	sound engineer, mCMS Server
Use Case Summary	The sound engineer forces system update using latest information from the mCMS Server
Pre-Condition	
Description/Main Flow **Stimulus/Response Sequences**	• The sound engineer clicks the Update button. • The system creates and sends an Update XML document to the mCMS server. • In response the mCMS server sends a configuration XML document to the system. The configuration XML document carries current mLAN network information. • On receiving the configuration XML document, the system updates the device trees and the grid matrix appropriately.
Post-Condition	The sound engineer successfully updated the system with latest information provided by the mCMS server.

Alternate Flow(s)	

f) Appling Patchbay Changes to the Network

Table 4.6: "Apply Changes" Use Case Stimulus/Response Sequences

Heading	***Description***
Use Case Name	Apply Changes
Actors	sound engineer, mCMS Server
Use Case Summary	The sound engineer applies changes made on the system to the physical mLAN network. Example changes include changed Master/Slave synchronisation settings, new connections and disconnections.
Pre-Condition	
Description/Main Flow **Stimulus/Response Sequences**	• The sound engineer clicks the Apply button. • The system loops through the device trees checking for pending connections and disconnections. If they are found the system follows the **Establish Audio Connections** and **Break Audio Connections** use case to commit the found connections and disconnections to the mLAN network.
Post-Condition	The system applied the changes to the physical mLAN network.
Alternate Flow(s)	
Exceptions	If there is not enough bandwidth on the mLAN network, the request will fail with an error message.
Used Use Cases	**Establish Audio Connections , Break Audio Connections**

g) Setting/Clearing Master/Slave Configurations

See *Appendix A-A1: section 4 (Table 4.7).*

h) Identifying a Device on the Network

Table 4.8: "Identify Device" Use Case Stimulus/Response Sequences

Heading	***Description***
Use Case Name	Identify Device
Actors	sound engineer, mCMS Server
Use Case Summary	In the event of a large network with many devices, the sound engineer can identify a particular device on the system.
Pre-Condition	
Description/Main Flow **Stimulus/Response Sequences**	• The sound engineer right-clicks a particular node either on the Destinations or Source tree on the system. • The system presents to the user a submenu with the menu items that include, Rename Device, Clear All Device Connections, More Device Info, Change Plug Layout and Identify Device. • The sound engineer selects the "Identify Device" menu item. • The system creates and sends a identify device XML message to the mCMS server.
Post-Condition	The sound engineer successfully identified a particular device on the mLAN network through the system. The Light Emitting Diodes of the device identified flashes.
Alternate Flow(s)	
Exceptions	
Used Use Cases	

APPENDIX B: mLAN Client/Server Communication Protocol

The mLAN Client/Server Communication protocol presented in this section was directly adopted from the mCMS Client/Server Communication protocol defined by Klinkradt (2004) that utilises Extensible Markup Language (XML) messages.

1. Communication Model

The communication channel between an mLAN Connection Management Server and its clients (patchbays) is a connected TCP/IP socket, as illustrated by *Figure 1*. Extensible Markup Language (XML) is utilised above the TCP/IP layer for all communication between clients and server. The remainder of this document specifies the format of these XML documents.

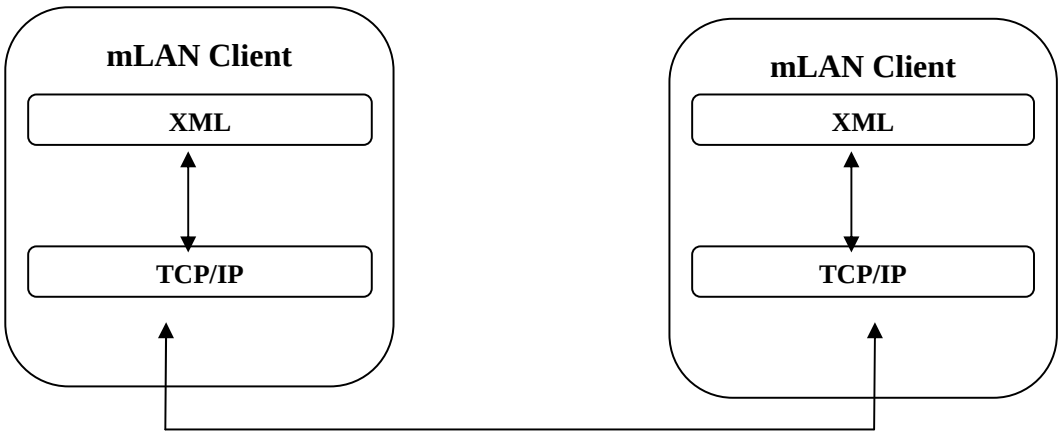

Figure 1: mCMS Client/Server Communication Model

2. Generic Client Server Document

A generic protocol format has been defined using XML to provide a platform for communication between mLAN Clients and Servers. XML was selected as the protocol of choice due to its inherent advantages of human readability and platform independence. The protocol is modelled around the underlying principle highlighted within AES-24, and which have been echoed in ACN and UPnP. This is to provide a platform independent mechanism for interoperability between networked devices and

providing a means for extensibility. It must be noted that at the inception of this protocol, it was never intended to serve any other purpose than to facilitate communication between the mLAN Server and Clients. The generic XML document, as represented in *Listing 1*, follows the Object Oriented mould of thinking. All communication must be directed at a target object within a device. A specific method of this object is to be invoked and the provided parameters to that method are applied. The assumption made is that devices are aware of the services (objects and methods) provided by other devices.

```
- <mLANCommand version="">
  - <Object namespace="" name="">
    - <Method name="">
        <Parameter name="" value="" />
      </Method>
    </Object>
  </mLANCommand>
```

Listing 1: Generic XML Document Format

The *mLANCommand* element identifies this as a mLAN specific XML document, and is flagged with a version attribute. The *Object* element provides the information required for targeting a specific object which is identified through a combination of the *namespace* and object *name* attributes. The *Method* element specifies the method that is to be invoked within the target object through the use of the method *name* attribute. Following this element is zero or more *Parameter* entities. Each of these Parameter element entries contains a *name* and *value* attribute pair. The *name* attribute identifies the targeted parameter with its value contained within the *value* attribute.

3. mLAN Client/Server Documents

This section details the documents currently utilised by the Linux based mLAN Connection Management Server and the Microsoft Windows Client. The communication between these two systems can be loosely broken down into client requests and server responses. The following objects have been identified and are currently utilised for mLAN Client Server communication:

- Useradmin

- Patch
- SampleRate
- Error

3.1 Useradmin Object

mLAN Client documents, as illustrated in *Listing 2*, which contain information regarding the useradmin objects are targeted at the useradmin object within the mLAN Server. The objects within these documents are not intended to reference instances of specific classes but refer to services that will be implemented by one or more classes within the server.

```
- <mLANCommand version="1.0">
  - <object name="useradmin" namespace="">
    - <method name="">
        <parameter name="" value="" />
      </method>
    </object>
  </mLANCommand>
```

Listing 2: Example mLAN Client Useradmin document

The useradmin object currently exposed by the mLAN Server includes the following methods:

- Logon
- Userlist
- Newuser
- Deleteuser
- Modifyuserlevel
- Modifypassword

For purposes of this investigation, only the Logon XML object was utilised since the patchbays developed only had one user level, the general user who had no rights for modifying user accounts.

3.1.1 Logon – Client Request

Listing 3 illustrates the fields defined within a useradmin logon request, as generated by a client in an attempt to logon to the server.

```
- <mLANCommand version="1.0">
  - <object name="useradmin" namespace="">
    - <method name="logon">
        <parameter name="username" value="admin" />
        <parameter name="password" value="mlan" />
      </method>
    </object>
  </mLANCommand>
```

Listing 3: Useradmin Logon XML request

The method name attribute identifies the appropriate service within the server and the parameter list consisting of name-value pair entries for username and password. The ordering of these parameters is not defined.

Logon – Server Response

If logon is successful no response is returned, else an error response document is returned providing a textual description of the logon failure.

3.2 Patch Object

The patch object is associated with the connection management services, for which the following methods have been defined:

- connect
- disconnect
- setCurrentPlugLayout
- syncsetup
- clearDanglingConnection
- pcplugs
- setnickname
- identify
- refresh

The generic patch request message is illustrated in *Listing 4*.

```
- <mLANCommand version="1.0">
  - <object name="patch" namespace="">
      <method name="" />
      <parameter name="" value="" />
    </object>
  </mLANCommand>
```

Listing 4: Generic Patch XML Request

3.2.1 Connect – Client Request

The connect patch request is utilised to create a connection between two mLAN Plugs. Multiple connections within a single request have not yet been defined. The defined parameters, as illustrated in *Listing 5*, are the GUID, plug type, and plug identifier for both source and destination plugs. It is feasible that the GUID be replaced with a nickname, but this functionality is not yet defined.

```
<?xml version="1.0" encoding="utf-16" ?>
- <mLANServerCommand version="1.0">
  - <object name="patch">
    - <method name="connect">
        <parameter name="sourceGUID" value="0013f00400400011" />
        <parameter name="sourcePlugType" value="audio" />
        <parameter name="sourcePlugID" value="1" />
        <parameter name="destinationGUID" value="0013f00400000014" />
        <parameter name="destinationPlugType" value="audio" />
        <parameter name="destinationPlugID" value="33" />
      </method>
    </object>
  </mLANServerCommand>
```

Listing 5: Patch Connect XML Request

Connect – Server Response

If the connect is successful no response is returned, else an error response document is returned providing a textual description of the failure.

3.2.2 Disconnect – Client Request

Connections between source and destination mLAN Plugs can be broken via a disconnect request [*Listing 6*]. The parameters required for this request are the GUID, plug type, and plug identifier for the destination plug. Multiple disconnections within one request are not yet defined.

```
<?xml version="1.0" encoding="utf-16" ?>
- <mLANServerCommand version="1.0">
  - <object name="patch">
    - <method name="disconnect">
        <parameter name="destinationGUID" value="0013f00400000014" />
        <parameter name="destinationPlugType" value="audio" />
        <parameter name="destinationPlugID" value="32" />
      </method>
    </object>
  </mLANServerCommand>
```

Listing 6: Patch Disconnect XML Request

Disconnect – Server Response

If the disconnect is successful no response is returned, else an error response document is returned providing a textual description of the failure.

3.2.3 SetCurrentPlugLayout – Client Request

```
<?xml version="1.0" encoding="utf-16" ?>
- <mLANServerCommand version="1.0">
  - <object name="patch">
    - <method name="setCurrentPlugLayout">
        <parameter name="GUID" value="0013f00400000014" />
        <parameter name="plugLayoutID" value="1" />
      </method>
    </object>
  </mLANServerCommand>
```

Listing 7: Patch SetCurrentPlugLayout XML Request

SetCurrentPlugLayout – Server Response

If setting the current Plug Layout is successful no response is returned, else an error response document is returned providing a textual description of the failure.

3.2.4 Syncsetup – Client Request

```
<?xml version="1.0" encoding="utf-16" ?>
- <mLANServerCommand version="1.0">
  - <object name="samplerate">
    - <method name="syncsetup">
        <parameter name="masterGUID" value="0013f00400400011" />
        <parameter name="masterWordClockOutputID" value="0" />
        <parameter name="masterCurrentSyncSourceID" value="1" />
        <parameter name="masterSampleRate" value="0000BB80" />
        <parameter name="slave" value="NODE_GUID='0013f00400000014',WORD_CLOCK_ID='0'" />
      </method>
    </object>
  </mLANServerCommand>
```

Listing 8: Patch Syncsetup XML Request

Syncsetup – Server Response

If the syncsetup is successful no response is returned, else an error response document is returned providing a textual description of the failure.

3.2.5 ClearDanglingConnection – Client Request

```
<?xml version="1.0" encoding="utf-16" ?>
- <mLANServerCommand version="1.0">
  - <object name="patch">
    - <method name="clearDanglingConnection">
        <parameter name="GUID" value="0013f00400400011" />
        <parameter name="direction" value="out" />
        <parameter name="plugID" value="15" />
        <parameter name="plugType" value="audio" />
      </method>
    </object>
  </mLANServerCommand>
```

Listing 9: Patch ClearDanglingConnection XML Request

ClearDanglingConnection – Server Response

If clearing dangling connection is successful no response is returned, else an error response document is returned providing a textual description of the failure.

3.2.6 Pcplugs – Client Request

```
<?xml version="1.0" encoding="utf-8" ?>
- <mLANServerCommand version="1.0">
  - <object name="patch">
    - <method name="pcplugs">
        <parameter name="numAudioSourcePlugs" value="2" />
        <parameter name="numAudioDestinationPlugs" value="48" />
        <parameter name="numMidiSourcePlugs" value="8" />
        <parameter name="numMidiDestinationPlugs" value="17" />
      </method>
    </object>
  </mLANServerCommand>
```

Listing 10: Patch Pcplugs XML Request

Pcplugs – Server Response

If setting pc plugs is successful no response is returned, else an error response document is returned providing a textual description of the failure.

3.2.7 SetNickname – Client Request

```
<?xml version="1.0" encoding="UTF-8" ?>
- <mLANServerCommand version="1.0">
  - <object name="control">
    - <method name="setnickname">
        <parameter name="deviceGUID" value="0013f00400400011" />
        <parameter name="nickname" value="IOne Source Device" />
      </method>
    </object>
  </mLANServerCommand>
```

Listing 11: Patch SetNickname XML Request

SetNickname – Server Response

If setting the nickname is successful no response is returned, else an error response document is returned providing a textual description of the failure.

3.2.8 Identify – Client Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<mLANServerCommand version="1.0">
- <object name="patch">
  - <method name="identify">
      <parameter name="GUID" value="0013f00400400011" />
    </method>
  </object>
</mLANServerCommand>
```

Listing 12: Patch Identify XML Request

Identify – Server Response

If identifying the device is successful no response is returned, else an error response document is returned providing a textual description of the failure.

3.2.9 Refresh – Client Request

A client can request a refresh of topology information through the use of the refresh request. No parameters are currently specified for this request.

```
- <mLANCommand version="1.0">
  - <object name="patch" namespace="">
      <method name="refresh" />
    </object>
  </mLANCommand>
```

Listing 13: Patch Refresh XML Request

The server refresh response contains a snapshot of all or a subset of the devices on the mLAN Network to which the server is connected. The format of this response is given in *Listing 14*.

Refresh – Server Response

The server refresh response contains a snapshot of all or a subset of the devices on the mLAN Network to which the server is connected. The format of this response is given in *Listing 14*.

```
- <mLANCommand>
  - <object name="patch" namespace="">
    - <method name="refresh">
      - <parameter name="configuration" value="">
          <mLANConfiguration />
        </parameter>
      </method>
    </object>
  </mLANCommand>
```

Listing 14: Patch Refresh XML Response

The value field contains a timestamp indicating when this document was generated by the server. The mLANConfiguration element contains the device specific information and is formatted as indicated in *Listing 15*.

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
- <mLANClientCommand>
  - <object name="patch">
    - <method name="refresh">
      - <parameter name="configuration" value="Tue Feb 27 11:10:43 2007">
        - <mLANConfiguration>
          - <IEEE1394Network>
            - <IEEE1394Bus bandwidthAvailable="2756" busName="3FF">
              + <IEEE1394Device GUID="0013f00400400011" firmware="DICE II OGT 0.1" model="DICE II Evaluation Board" nickname="IOne Connects-left"
                  nicknameIsWriteable="yes" numPossibleDeviceConnections="4" vendor="WaveFront">
              - <IEEE1394Device GUID="0013f00400000014" firmware="X1 OGT 070222" model="Firewire Digital Audio Snake" nickname="IOne Connects-right"
                  nicknameIsWriteable="yes" numPossibleDeviceConnections="3" vendor="I/One Connects">
                - <mLANDevice>
                  - <mLANDevicePlugs>
                      <plug direction="out" id="66" isDangling="no" nameIsWriteable="no" plugName="Analog In 1" plugType="audio" />
                      <plug direction="in" id="115" isDangling="no" nameIsWriteable="no" plugName="MIDI Out" plugType="midi" />
                    </mLANDevicePlugs>
                  - <mLANDevicePlugLayouts currentPlugLayoutID="1" numPlugLayouts="3">
                      <plugLayout id="0" nameIsWriteable="no" plugLayoutName="tx1: 8 ADAT 8 AES. tx2: 16 Anal" />
                      <plugLayout id="1" nameIsWriteable="no" plugLayoutName="tx1: 8 AES 8 Analog" />
                      <plugLayout id="2" nameIsWriteable="no" plugLayoutName="tx1: 8 Analog" />
                    </mLANDevicePlugLayouts>
                  - <mLANDeviceSyncSources numSyncSources="4">
                      <syncSource currentSampleRate="0000bb80" id="0" nameIsWriteable="no"
                        supportedSampleRates="00007d00|0000ac44|0000bb80|00015888|00017700|0002b110|0002ee00|" syncMode="1"
                        syncSourceName="SYT" />
                      <syncSource currentSampleRate="0000bb80" id="2" nameIsWriteable="no"
                        supportedSampleRates="00007d00|0000ac44|0000bb80|00015888|00017700|0002b110|0002ee00|" syncMode="3"
                        syncSourceName="AES RX" />
                    </mLANDeviceSyncSources>
                  - <mLANDeviceWordClockOutputs numWordClockOutputs="1">
                      <wordClockOutput currentSyncSourceID="0" id="0" masterGUID="0013f00400400011" masterWordClockOutputID="0" />
                    </mLANDeviceWordClockOutputs>
                  </mLANDevice>
                </IEEE1394Device>
              </IEEE1394Bus>
            </IEEE1394Network>
          - <Connections>
              <Patch destEndPointLocator="NODE_GUID='0013f00400400011',MLAN_PLUG_ID='Audio In 1'"
                srcEndPointLocator="NODE_GUID='0013f00400000014',MLAN_PLUG_ID='Audio Out 2'" />
            </Connections>
            <session end="2007-02-27 10:56:54" sessionID="0" sessionName="dummy" start="2007-02-27 10:56:54" />
          </mLANConfiguration>
        </parameter>
      </method>
    </object>
```

Listing 15: Configuration XML Document

The IEEE1394Network shown in *Listing 15* contains all of the IEEE1394Bus elements, Connections and session elements. There is a separate IEEE1394Bus element for every bus that is present on the network. Each bus is identified by a busName attribute, with the value of 3FF indicating referring to the local bus. Under each bus element are zero or more devices represented by the IEEE1394Device element. Attributes of this element include:

- GUID – The GUID of this node
- currentSampleRate – The sample rate that this device is currently utilising.
- currentSyncMode – The synchronisation modes that are supported by this device.

These are defined as indicated in *table 1*.

Synchronisation Mode	**Value**
No synchronisation supported	0
Only SYT synchronisation mode supported	1
Only internal synchronisation supported 2	2
SYT and internal synchronisation supported	3

masterGUID – The GUID of the master device that this device is currently slaved to. The value of “none” is utilised if the device is not currently a slave.

- model – The model number of this device.
- vendor – The vendor of this device.
- supportedMasterSampleRates – A delimited list of master sample rates (in hex format) that are supported by this device.
- supportedSlaveSampleRates – A delimited list of slave sample rates (in hex format) that are supported by this device.

The mLANDevice element indicates that the current device is an mLAN device and contains an mLANTransporter Element. The mLANTransporter Element is a

container for a list of mLANDevicePlug elements, which represent the plugs exposed by the mLAN device. The attributes of the mLANDevicePlug element includes:

- direction – This attribute indicates if this plug is a source or destination plug.
- plugName – A textual name that is currently assigned to this plug. This textual name is typically displayed to an end user.
- plugType – This attribute specifies the type of plug that is being defined. Currently defined types include audio and midi.

The Connections element specifies the connections that are currently present between devices on the network. An individual connection is represented by a patch element which defines the following attributes:

- destEndPointLocator – The destination of data passing via this connection
- srcEndPointLocator – The source of data for this connection.

The attributes of the destEndPointLocator and srcEndPointLocator are specified using a comma delimited list. For example: NODE_GUID=”guid”, MLAN_PLUG_ID=”textual plug name” The session element is currently not utilised, but is in place to facilitate the management of time-based sessions with the automatic allocation of devices within these sessions.

3.3 SampleRate Object

The SampleRate object provides for the setting of sample rates on targeted devices as well as allowing clients to setup master and slave device relationships. For this purpose the following methods have been defined:

- setGlobalMasterRate
- syncsetup

3.3.1 setGlobalMasterRate – Client Request

The setGlobalMasterRate document is illustrated below in *Listing 16*.

```
<?xml version="1.0" encoding="utf-8" ?>
- <mLANServerCommand version="1.0">
  - <object name="samplerate">
    - <method name="setGlobalMasterRate">
        <parameter name="masterGUID" value="0013f004007fffff" />
        <parameter name="masterSampleRate" value="0000AC44" />
      </method>
    </object>
  </mLANServerCommand>
```

Listing 16: Patch setGlobalMasterRate XML Response

The server will slave all devices to the device specified within the masterGUID attribute and at the rate specified within masterSampleRate.

3.3.2 setGlobalMasterRate – Server Response

No response is currently defined.

3.4 Notify – Server Response

The notify method is intended to notify a client user of a failed server request, an example of which is illustrated in *Listing 17*. The only defined parameter is the description parameter, which provides the error message in its value attribute.

```
- <mLANCommand>
  - <object name="error" namespace="">
    - <method name="notify">
        <parameter name="description" value="Logon failed due to invalid username or password" />
      </method>
    </object>
  </mLANCommand>
```

Listing 17: Error Notify Response

APPENDIX C: Usability Documentations

C1 User Test Profile Form

1. USER IDENTIFICATION INFORMATION

<table>
<tr><td>User Age:</td><td>
<table>
<tr><td colspan="5">Indicate with an X in the correct column</td></tr>
<tr><td><18</td><td>18-30</td><td>31-40</td><td>40-50</td><td>51<</td></tr>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
</table>
</td></tr>
<tr><td>User Occupation:</td><td></td></tr>
<tr><td>Gender:</td><td>
<table>
<tr><td colspan="2">Indicate with an X in the correct column</td></tr>
<tr><td>Male</td><td>Female</td></tr>
<tr><td></td><td></td></tr>
</table>
</td></tr>
</table>

2. USER EXPERIENCE AND PREVIOUS KNOWLEDGE

2.1. Highest education

Please mark with an X in the correct column.	
Tertiary (university)	
Trade (apprenticeship)	
Secondary (high school)	
Primary	
No formal education	

2.2. Length of time in current position

Please mark with an X in the correct column.	
Less than 6 months	
6 months – 1 year	
1 – 2 years	
Over 2 years	
other	

2.3. Length of time with this organization

Please mark with an X in the correct column.
Less than 6 months
6 months – 1 year
1 – 2 years
Over 2 years
other

2.4 Computer usage

Please mark with an X in the correct column. ***This section determines how often you use a computer system.***				
Experience:	<1 month	1-6 months	6 months- 2 years	Over 2 years
Frequency:				
Daily				
Weekly				
Monthly				
Never				

2.5. PC Features

PC Type:	
Memory Size:	
Processor Speed:	
HDD Size:	
Network Connection Speed:	
Patchbay Type:	***Please specify the type of patchbay you are using.***

Grid-Based	**Graphic-Based**	**Other**

If you chose "other" specify the type of patchbay you are familiar with:

2.6. Learning Cycle

Please mark with an X in the correct column. ***This section evaluates how you learnt using the Patchbay software you specified in section 2.5.***		
Trail and error	**Follow documentation**	**Follow documentation and Trail and error**

2.7. Notes

3. WHAT THE USER DOES

3.1. Duties

1.
2.
3.
4.
5.
6.

C2 Usability Testing Questionnaire

Please tick on the appropriate square for each software feature attribute.

1. CONSISTENCY

Allocated Scores for each software feature attribute evaluated		**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	
Testing Question	**Feature Attribute**											**Feature Attribute**
1. The naming is consistent across displays and menu options of the <<application name>>?	Strongly disagree											Strongly agree
2. The labels are located at consistent location on screens of the <<application name>>?	Strongly disagree											Strongly agree
3. The wording used is consistent with user guidance provided?	Strongly disagree											Strongly agree
4. The grouping of menu options is logical?	Strongly disagree											Strongly agree
5. The ordering of menu options is logical?	Strongly disagree											Strongly agree

2. LEARNABILITY

Allocated Scores for each software feature attribute evaluated		**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	
Testing Question	**Feature Attribute**											**Feature Attribute**
1. Learning to operate the <<application name>> system is easy?	Strongly disagree											Strongly agree
2. Remembering names and use of commands within the <<application name>> System is not challenging?	Strongly disagree											Strongly agree
3. Performing tasks is straightforward?	Strongly disagree											Strongly agree
4. Supplemental reference materials are useful?	Strongly disagree											Strongly agree
5. Command names are meaningful?	Strongly disagree											Strongly agree
6. The <<application name>> require a steep learning curve?	Strongly disagree											Strongly agree

3. TERMINOLOGY, USER GUIDANCE AND SYSTEM INFORMATION

Allocated Scores for each software feature attribute evaluated		**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	
Testing Question	**Feature Attribute**											**Feature Attribute**
1. Is the terminology used in the system related to task?	Strongly disagree											Strongly agree
3. System feedback: How helpful are error messages?	Strongly disagree											Strongly agree
4. Does the <<application name>> provide CANCEL option?	Strongly disagree											Strongly agree
5. Is HELP provided?	Strongly disagree											Strongly agree
6. Is completion of processing indicated?	Strongly disagree											Strongly agree
7. Are error messages no disruptive/ informative?	Strongly disagree											Strongly agree

4. SCREEN

Allocated Scores for each software feature attribute evaluated		**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	
Testing Question	**Feature Attribute**											**Feature Attribute**
1. Reading characters on the screen is easy?	Strongly disagree											Strongly agree
2. Organization of Information is logical and standard?	Strongly disagree											Strongly agree
3. Position of messages on screen.	Strongly disagree											Strongly agree

5. FLEXIBILITY

Allocated Scores for each software feature attribute evaluated		**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	
Testing Question	**Feature Attribute**											**Feature Attribute**
1. Does the <<application name>> have direct manipulation capability?	Strongly disagree											Strongly agree
2. Are the menu options dependent on context?	Strongly disagree											Strongly agree
3. Can the user display elements according to their needs?	Strongly disagree											Strongly agree
4. Are users allowed to customize windows?	Strongly disagree											Strongly agree
5. Can users assign command names?	Strongly disagree											Strongly agree
6. Does the system provide zooming for display expansion?	Strongly disagree											Strongly agree

6. MINIMAL ACTION

Allocated Scores for each software feature attribute evaluated		**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	
Testing Question	**Feature Attribute**											**Feature Attribute**
1. Does the <<application name>> provide default values?	Strongly disagree											Strongly agree
2. Does <<application name>> provide function keys for frequent control entries?	Strongly disagree											Strongly agree
3. Is the menu selection by pointing? primary Means of sequence control?	Strongly disagree											Strongly agree
4. Does the <<application name>> require minimal cursor positioning?	Strongly disagree											Strongly agree

7. PERCEPTUAL LIMITATION

Allocated Scores for each software feature attribute evaluated		**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	
Testing Question	**Feature Attribute**											**Feature Attribute**
1. Is the cursor distinctive?	Strongly disagree											Strongly agree
2. Are display elements distinctive?	Strongly disagree											Strongly agree
3. Does it provide easily distinguished colours?	Strongly disagree											Strongly agree
4. Is the active window indicated?	Strongly disagree											Strongly agree
5. Are menus distinct from other displayed information?	Strongly disagree											Strongly agree
6. Are groups of information demarcated?	Strongly disagree											Strongly agree
7. Is the screen density reasonable?	Strongly disagree											Strongly agree

8. SYSTEM CAPABILITIES

Allocated Scores for each software feature attribute evaluated		**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	
Testing Question	**Feature Attribute**											**Feature Attribute**
1. System speed is good enough	Strongly disagree											Strongly agree
2. System reliability is good enough	Strongly disagree											Strongly agree

9. OVERALL REATION TO THE SOFTWARE:
Product Reaction Cards (Microsoft Wording Scales)

Please tick on the appropriate square for each Key Word that best describes the overall software. The user can select as many Key Words as possible.

☐Accessible	☐Desirable	☐Gets in the way	☐Patronizing	☐Stressful
☐Appealing	☐Easy to use	☐Hard to use	☐Personal	☐Time-consuming
☐Attractive	☐Efficient	☐High quality	☐Predictable	☐Time-saving
☐Busy	☐Empowering	☐Inconsistent	☐Relevant	☐Too technical
☐Collaborative	☐Exciting	☐Intimidating	☐Reliable	☐Trustworthy
☐Complex	☐Familiar	☐Inviting	☐Rigid	☐Uncontrollable
☐Comprehensive	☐Fast	☐Motivating	☐Simplistic	☐Unconventional
☐Confusing	☐Flexible	☐Not valuable	☐Slow	☐Unpredictable
☐Connected	☐Fresh	☐Organized	☐Sophisticated	☐Usable
☐Consistent	☐Frustrating	☐Overbearing	☐Stimulating	☐Useful
☐Customizable	☐Fun	☐Overwhelming	☐Straight Forward	☐Valuable

C3 Heuristic Evaluation Checklist Form

1. Visibility of System Status

The system should always keep user informed about what is going on, through appropriate feedback wi thin reasonable time.

#	Review Checklist	Yes No N/A	Comments
1.1	Does every display begin with a title or header that describes screen contents?	O O O	
1.2	Is there a consistent icon design scheme and stylistic treatment across the system?	O O O	
1.3	Is a single, selected icon clearly visible when surrounded by unselected icons?	O O O	
1.4	Do menu instructions, prompts, and error messages appear in the same place(s) on each menu?	O O O	
1.5	In multipage data entry screens, is each page labelled to show its relation to others?	O O O	
1.6	If overtype and insert mode are both available, is there a visible indication of which one the user is in?	O O O	
1.7	If pop-up windows are used to display error messages, do they allow the user to see the field in error?	O O O	
1.8	Is there some form of system feedback for every operator action?	O O O	
1.9	After the user completes an action (or group of actions), does the feedback indicate that the next group of actions can be started?	O O O	
1.10	Is there visual feedback in menus or dialog boxes about which choices are selectable?	O O O	
1.11	Is there visual feedback in menus or dialog boxes about which choice the cursor is on now?	O O O	
1.12	If multiple options can be selected in a menu or dialog box, is there visual feedback about which options are already selected?	O O O	
1.13	Is there visual feedback when objects are selected or moved?	O O O	
1.14	Is the current status of an icon clearly indicated?	O O O	

2. Match Between System and the Real World

The system should speak the user's language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order.

#	Review Checklist	Yes No N/A	Comments
2.1	Are icons concrete and familiar?	O O O	
2.2	Are menu choices ordered in the most logical way, given the user, the item names, and the task variables?	O O O	
2.3	If there is a natural sequence to menu choices, has it been used?	O O O	
2.4	Do related and interdependent fields appear on the same screen?	O O O	
2.5	If shape is used as a visual cue, does it match cultural conventions?	O O O	
2.6	Do the selected colours correspond to common expectations about colour codes?	O O O	
2.7	When prompts imply a necessary action, are the words in the message consistent with that action?	O O O	
2.8	Do keystroke references in prompts match actual key names?	O O O	
2.9	On data entry screens, are tasks described in terminology familiar to users?	O O O	
2.10	Are field-level prompts provided for data entry screens?		
2.11	For question and answer interfaces, are questions stated in clear, simple language?	O O O	
2.12	Do menu choices fit logically into categories that have readily understood meanings?	O O O	
2.13	Are menu titles parallel grammatically?	O O O	

3. User Control and Freedom

Users should be free to select and sequence tasks (when appropriate), rather than having the system do this for them. Users often choose system functions by mistake and will need a clearly marked "emergency exit" to leave the unwanted state without having to go through an extended dialogue. Users should make their own decisions (with clear information) regarding the costs of exiting current work. The system should support undo and redo.

#	Review Checklist	Yes No N/A	Comments
3.1	If setting up windows is a low-frequency task, is it particularly easy to remember?	O O O	
3.2	In systems that use overlapping windows, is it easy for users to rearrange windows on the screen?	O O O	
3.3	In systems that use overlapping windows, is it easy for users to switch between windows?	O O O	
3.4	When a user's task is complete, does the system wait for a signal from the user before processing?	O O O	
3.5	Can users type-ahead in a system with many nested menus?	O O O	
3.6	Are users prompted to confirm commands that have drastic, destructive consequences?	O O O	
3.7	Is there an "undo" function at the level of a single action, a data entry, and a complete group of actions?	O O O	
3.8	Can users cancel out of operations in progress?	O O O	
3.9	Are character edits allowed in commands?	O O O	
3.10	Can users reduce data entry time by copying and modifying existing data?	O O O	
3.11	Are character edits allowed in data entry fields?	O O O	
3.12	If menu lists are long (more than seven items), can users select an item either by moving the cursor or by typing a mnemonic code?	O O O	
3.13	If the system uses a pointing device, do users have the option of either clicking on menu items or using a keyboard shortcut?	O O O	
3.14	Are menus broad (many items on a menu) rather than deep (many menu levels)?	O O O	
3.15	If the system has multiple menu levels, is there a mechanism that allows users to go back to previous menus?	O O O	

4. Consistency and Standards

Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.

#	Review Checklist	Yes No N/A	Comments
4.1	Have industry or company formatting standards been followed consistently in all screens within a system?	O O O	
4.2	Has a heavy use of all uppercase letters on a screen been avoided?	O O O	
4.3	Do abbreviations not include punctuation?	O O O	
4.4	Are integers right-justified and real numbers decimal-aligned?	O O O	
4.5	Are icons labelled?	O O O	
4.6	Are there no more than twelve to twenty icon types?	O O O	
4.7	Are there salient visual cues to identify the active window?	O O O	
4.8	Does each window have a title?	O O O	
4.9	Are vertical and horizontal scrolling possible in each window?	O O O	
4.10	Does the menu structure match the task structure?	O O O	
4.11	Have industry or company standards been established for menu design, and are they applied consistently on all menu screens in the system?	O O O	
4.12	Are menu choice lists presented vertically?	O O O	
4.13	If "exit" is a menu choice, does it always appear at the bottom of the list?	O O O	
4.14	Are menu titles either cantered or left-justified?	O O O	
4.15	Are menu items left-justified, with the item number or mnemonic preceding the name?	O O O	
4.16	Do embedded field-level prompts appear to the right of the field label?	O O O	
4.17	Do on-line instructions appear in a consistent location across screens?	O O O	
4.18	Are field labels and fields distinguished typographically?	O O O	
4.19	Are field labels consistent from one data entry screen to another?	O O O	
#	**Review Checklist**	**Yes No**	**Comments**

		N/A	
4.20	Do field labels appear to the left of single fields and above list fields?	O O O	
4.21	Are attention-getting techniques used with care?	O O O	
4.22	Intensity: two levels only	O O O	
4.23	Size: up to four sizes	O O O	
4.24	Font: up to three	O O O	
4.25	Blink: two to four hertz	O O O	
4.26	Colour: up to four (additional colours for occasional use only)	O O O	
4.27	Are there no more than four to seven colours, and are they far apart along the visible spectrum?	O O O	
4.28	Is a legend provided if colour codes are numerous or not obvious in meaning?	O O O	
4.28	Have pairings of high-chroma, spectrally extreme colours been avoided?	O O O	
4.30	Are saturated blues avoided for text or other small, thin line symbols?	O O O	
4.31	Is the most important information placed at the beginning of the prompt?	O O O	
4.32	Are user actions named consistently across all prompts in the system?	O O O	
4.33	Are system objects named consistently across all prompts in the system?	O O O	
4.34	Do field-level prompts provide more information than a restatement of the field name?	O O O	
4.35	Are menu choice names consistent, both within each menu and across the system, in grammatical style and terminology?	O O O	
4.36	Does the structure of menu choice names match their corresponding menu titles?	O O O	
4.43	Do abbreviations follow a simple primary rule and, if necessary, a simple secondary rule for abbreviations that otherwise would be duplicates?	O O O	

5. Help Users Recognize, Diagnose, and Recover From Errors

Error messages should be expressed in plain language (NO CODES).

#	Review Checklist	Yes No N/A	Comments
5.1	Is sound used to signal an error?	O O O	
5.2	Are prompts stated constructively, without overt or implied criticism of the user?	O O O	
5.3	Do prompts imply that the user is in control?	O O O	
5.4	Are prompts brief and unambiguous?	O O O	
5.5	Are error messages worded so that the system, not the user, takes the blame?	O O O	
5.6	If humorous error messages are used, are they appropriate and inoffensive to the user population?	O O O	
5.7	Are error messages grammatically correct?	O O O	
5.8	Do error messages avoid the use of exclamation points?	O O O	
5.9	Do error messages avoid the use of violent or hostile words?	O O O	
5.10	Do error messages avoid an anthropomorphic tone?	O O O	
5.11	Do all error messages in the system use consistent grammatical style, form, terminology, and abbreviations?	O O O	
5.12	Do messages place users in control of the system?	O O O	
5.13	If an error is detected in a data entry field, does the system place the cursor in that field or highlight the error?	O O O	
5.14	Do error messages inform the user of the error's severity?	O O O	
5.15	Do error messages suggest the cause of the problem?	O O O	
5.16	Do error messages provide appropriate semantic information?	O O O	
5.17	Do error messages provide appropriate syntactic information?	O O O	
5.18	Do error messages indicate what action the user needs to take to correct the error?	O O O	
5.19	If the system supports both novice and expert users, are multiple levels of error-message detail available?	O O O	

6. Error Prevention

Even better than good error messages is a careful design which prevents a problem from occurring in the first place.

#	Review Checklist	Yes No N/A	Comments
6.1	If the database includes groups of data, can users enter more than one group on a single screen?	O O O	
6.2	Have dots or underscores been used to indicate field length?	O O O	
6.3	Is the menu choice name on a higher-level menu used as the menu title of the lower-level menu?	O O O	
6.4	Are menu choices logical, distinctive, and mutually exclusive?	O O O	
6.5	Are data inputs case-blind whenever possible?	O O O	
6.6	If the system displays multiple windows, is navigation between windows simple and visible?	O O O	
6.7	Are the function keys that can cause the most serious consequences in hard-to-reach positions?	O O O	
6.8	Are the function keys that can cause the most serious consequences located far away from low-consequence and high-use keys?	O O O	
6.9	Has the use of qualifier keys been minimized?	O O O	
6.10	If the system uses qualifier keys, are they used consistently throughout the system?	O O O	
6.11	Does the system prevent users from making errors whenever possible?	O O O	
6.12	Does the system warn users if they are about to make a potentially serious error?	O O O	
6.13	Does the system intelligently interpret variations in user commands?	O O O	
6.14	Do data entry screens and dialog boxes indicate the number of character spaces available in a field?	O O O	
6.15	Do fields in data entry screens and dialog boxes contain default values when appropriate?	O O O	

7. Recognition Rather Than Recall

Make objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.

#	Review Checklist	Yes No N/A	Comments
7.1	For question and answer interfaces, are visual cues and white space used to distinguish questions, prompts, instructions, and user input?	O O O	
7.2	Does the data display start in the upper-left corner of the screen?	O O O	
7.3	Are multiword field labels placed horizontally (not stacked vertically)?	O O O	
7.4	Are all data a user needs on display at each step in a transaction sequence?	O O O	
7.5	Are prompts, cues, and messages placed where the eye is likely to be looking on the screen?	O O O	
7.6	Have prompts been formatted using white space, justification, and visual cues for easy scanning?	O O O	
7.7	Do text areas have "breathing space" around them?	O O O	
7.8	Is there an obvious visual distinction made between "choose one" menu and "choose many" menus?	O O O	
7.9	Have spatial relationships between soft function keys (on-screen cues) and keyboard function keys been preserved?	O O O	
7.10	Does the system gray out or delete labels of currently inactive soft function keys?	O O O	
7.11	Is white space used to create symmetry and lead the eye in the appropriate direction?	O O O	
7.12	Have items been grouped into logical zones, and have headings been used to distinguish between zones?	O O O	
7.13	Are zones no more than twelve to fourteen characters wide and six to seven lines high?	O O O	
7.14	Have zones been separated by spaces, lines, colour, letters, bold titles, rules lines, or shaded areas?	O O O	
7.15	Are field labels close to fields, but separated by at least one space?	O O O	
7.16	Are long columnar fields broken up into groups of five, separated by a blank line?	O O O	

8. Flexibility and Minimalist Design

Accelerators-unseen by the novice user-may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions. Provide alternative means of access and operation for users who differ from the "average" user (e.g., physical or cognitive ability, culture, language, etc.)

#	Review Checklist	Yes No N/A	Comments
8.1	If the system supports both novice and expert users, are multiple levels of error message detail available?	O O O	
8.2	Does the system allow novices to use a keyword grammar and experts to use a positional grammar?	O O O	
8.3	Can users define their own synonyms for commands?	O O O	
8.4	Does the system allow novice users to enter the simplest, most common form of each command, and allow expert users to add parameters?	O O O	
8.5	Do expert users have the option of entering multiple commands in a single string?	O O O	
8.6	Does the system provide function keys for high-frequency commands?	O O O	
8.7	For data entry screens with many fields or in which source documents may be incomplete, can users save a partially filled screen?	O O O	
8.8	Does the system automatically enter leading zeros?	O O O	
8.9	If menu lists are short (seven items or fewer), can users select an item by moving the cursor?	O O O	
8.10	If the system uses a type-ahead strategy, do the menu items have mnemonic codes?	O O O	

9. Aesthetic and Minimalist Design

Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.

#	Review Checklist	Yes No N/A	Comments
9.1	Is only (and all) information essential to decision making displayed on the screen?	O O O	
9.2	Are all icons in a set visually and conceptually distinct?	O O O	
9.3	Have large objects, bold lines, and simple areas been used to distinguish icons?	O O O	
9.4	Does each icon stand out from its background?	O O O	
9.5	If the system uses a standard GUI interface where menu sequence has already been specified, do menus adhere to the specification whenever possible?	O O O	
9.6	Are meaningful groups of items separated by white space?	O O O	
9.7	Does each data entry screen have a short, simple, clear, distinctive title?	O O O	
9.8	Are field labels brief, familiar, and descriptive?	O O O	
9.9	Are prompts expressed in the affirmative, and do they use the active voice?	O O O	
9.10	Is each lower-level menu choice associated with only one higher level menu?	O O O	
9.11	Are menu titles brief, yet long enough to communicate?	O O O	
9.12	Are there pop-up or pull-down menus within data entry fields that have many, but well-defined, entry options?	O O O	

10. Help and Documentation

Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large.

#	Review Checklist	Yes No N/A	Comments
10.1	If users are working from hard copy, are the parts of the hard copy that go on-line marked?	O O O	
10.2	Are on-line instructions visually distinct?	O O O	
10.3	Do the instructions follow the sequence of user actions?	O O O	
10.4	If menu choices are ambiguous, does the system provide additional explanatory information when an item is selected?	O O O	
10.5	Are data entry screens and dialog boxes supported by navigation and completion instructions?	O O O	
10.6	If menu items are ambiguous, does the system provide additional explanatory information when an item is selected?	O O O	
10.7	Are there memory aids for commands, either through on-line quick reference or prompting?	O O O	
10.8	Is the help function visible; for example, a key labelled HELP or a special menu?	O O O	
10.9	Is the help system interface (navigation, presentation, and conversation) consistent with the navigation, presentation, and conversation interfaces of the application it supports?	O O O	
10.10	Navigation: Is information easy to find?	O O O	
10.11	Presentation: Is the visual layout well designed?	O O O	
10.12	Conversation: Is the information accurate, complete, and understandable?	O O O	
10.13	Is the information relevant?	O O O	
10.14	Goal-oriented (What can I do with this program?)	O O O	
10.15	Descriptive (What is this thing for?)	O O O	
10.16	Procedural (How do I do this task?)	O O O	
10.17	Interpretive (Why did that happen?)	O O O	
10.18	Navigational (Where am I?)	O O O	
10.19	Is there context-sensitive help?	O O O	
10.20	Can the user change the level of detail available?	O O O	

11. Skills

The system should support, extend, supplement, or enhance the user's skills, background knowledge, and expertise ----not replace them.

#	Review Checklist	Yes No N/A	Comments
11.1	Can users choose between iconic and text display of information?	O O O	
11.2	Are window operations easy to learn and use?	O O O	
11.3	If users are experts, usage is frequent, or the system has a slow response time, are there fewer screens (more information per screen)?	O O O	
11.4	If users are novices, usage is infrequent, or the system has a fast response time, are there more screens (less information per screen)?	O O O	
11.5	Does the system automatically colour-code items, with little or no user effort?	O O O	
11.6	If the system supports both novice and expert users, are multiple levels of detail available.	O O O	
11.7	Are users the initiators of actions rather than the responders?	O O O	
11.8	Does the system perform data translations for users?	O O O	
11.9	Do field values avoid mixing alpha and numeric characters whenever possible?	O O O	
11.10	If the system has deep (multilevel) menus, do users have the option of typing ahead?	O O O	
11.12	When the user enters a screen or dialog box, is the cursor already positioned in the field users are most likely to need?	O O O	
11.13	Can users move forward and backward within a field?	O O O	
11.14	Is the method for moving the cursor to the next or previous field both simple and visible?	O O O	
11.15	Has auto-tabbing been avoided except when fields have fixed lengths or users are experienced?	O O O	
11.16	Do the selected input device(s) match user capabilities?	O O O	
11.17	Are cursor keys arranged in either an inverted T (best for experts) or a cross configuration (best for novices)?	O O O	

12. Pleasurable and Respectful Interaction with the User

The user's interactions with the system should enhance the quality of her or his work-life. The user should be treated with respect. The design should be aesthetically pleasing- with artistic as well as functional value.

#	Review Checklist	Yes No N/A	Comments
12.1	Is each individual icon a harmonious member of a family of icons?	O O O	
12.2	Has excessive detail in icon design been avoided?	O O O	
12.3	Has colour been used with discretion?	O O O	
12.4	Has the amount of required window housekeeping been kept to a minimum?	O O O	
12.5	If users are working from hard copy, does the screen layout match the paper form?	O O O	
12.6	Has colour been used specifically to draw attention, communicate organization, indicate status changes, and establish relationships?	O O O	
12.7	Can users turn off automatic colour coding if necessary?	O O O	
12.8	Are typing requirements minimal for question and answer interfaces?	O O O	
12.9	Do the selected input device(s) match environmental constraints?	O O O	
12.13	If the system uses multiple input devices, has hand and eye movement between input devices been minimized?	O O O	
12.14	If the system supports graphical tasks, has an alternative pointing device been provided?	O O O	
12.15	Is the numeric keypad located to the right of the alpha key area?	O O O	
12.16	Are the most frequently used function keys in the most accessible positions?	O O O	
12.17	Does the system complete unambiguous partial input on a data entry field?	O O O	

13. Privacy

The system should help the user to protect personal or private information- belonging to the user or the his/her clients**.**

#	Review Checklist	Yes No N/A	Comments
13.1	Are protected areas completely inaccessible?	O O O	
13.2	Can protected or confidential areas be accessed with certain passwords?	O O O	
13.3	Is this feature effective and successful?	O O O	

System Title: _____ Release #: _____

Evaluator: _____ Date: _____

C4 Hospitality/Convention Centre Paper Prototype Questions

a) **Establishing An Audio Connection**

i. Connect the computer in the Reception *zone* to a speaker in the Lounge *zone* and to a speaker in the Room 4 *zone*.

b) Rearranging Hotel Zone Devices

ii. Place the “CD player” at the bottom-left corner of the Reception *zone*.

iii. Place the Computer at the centre of the Reception *zone*.

c) Rearranging Hotel Zones

iv. Replace Room 2 *zone* by Room 1 *zone*.

d) Rearranging Hotel Zones

v. Replace Room 2 *zone* by Room 8 *zone*.

e) Making An Audio Disconnection

i. Disconnect the speaker in the Lounge *zone*.