Connection Management Applications for High-Speed Audio Networking

Phathisile Sibanda, Greg Foster, Richard Foss
DEPARTMENT OF COMPUTER SCIENCE
Rhodes University, Grahamstown
Email: g05x5782@campus.ru.ac.za, g.foster@ru.ac.za, r.foss@ru.ac.za
Phone: +27 46 603 8291, Fax: +27 46 636 1915

Abstract- Traditionally connection management applications, also known as patchbays, for highspeed audio networking have been predominantly developed using third generation languages such as 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 the required functionality. This has led to a need among highspeed audio networking application developers for a tool that will enable them to develop connection management applications easily and within the shortest possible time. In addition, this tool should provide them with the reliability and flexibility required by applications controlling signal routing in networks carrying realtime data. Highspeed audio networks are used for various purposes that include audio/video production and broadcasting. This paper presents an investigation evaluating the possibility of using Adobe Flash for developing connection management applications. Broadcast and Studio were considered for this investigation. Findings indicate that complex connection management applications can effectively be implemented with the Flash IDE and ActionScript.

Index Terms - Firewire, patchbay, music Local Area Network (mLAN), client/server architecture, Adobe Flash, ActionScript.

I. INTRODUCTION

The Audio Engineering Group in the Computer Science Department at Rhodes University is currently working in collaboration with Yamaha Japan in developing and deploying the music Local Area Network (mLAN). mLAN is a networking technology that is based on the Firewire standard also known as IEEE1394 networking technology. Firewire provides high data transfer rates, supports hotplugging and plugplay performance capabilities, and has provided low latency and determinism required by a network carrying realtime data [1]. The group has chosen to implement mLAN using the client/server approach as described in section 2.

Adobe Flash was originally designed to create animations for display on web pages in the 1980s [10]. The initial version of Adobe Flash (version 1.0, December 1996 by Macromedia) supported simple functionality such as enhanced bitmap integration, buttons, the Library, and the capability to tween colour changes. Over the past decade Adobe Flash has evolved from a simple graphics creation software to a powerful interactive platform, with an objectoriented typesafe dynamic scripting engine that supports ActionScript, bitmap rendering and advanced video and audio playback features. Coupled with these features is its flexible design and animation authoring IDE that is composed of many programming interfaces that include the code, the animation and the vector art editors, a compiler, a debugger and a very good help system. With these features Adobe Flash can be compared to other common IDE application such as Microsoft Visual Studio IDE.

Broadcast and Studio environments were chosen as the focal point for this investigation because of their size and complexity. It was assumed that if Adobe Flash was able to implement a usable, efficient and reliable application for these two environments, then Adobe Flash would be able to develop applications for the Hospitality environments such as hotels and convention centres which are less complex. This paper defines the mLAN client/server configuration, discusses current highspeed networking connection management applications and provides a detailed approach to the study coupled with the methods and the results obtained from the investigation.

II. MLAN CLIENT/SERVER ARCHITECTURE

Figure 1: mLAN Client/Server Configuration

Figure 1 shows the mLAN client/server configuration where patchbay applications developed for this investigation were deployed. The mLAN client/server diagram [Figure 1] shows two mLAN compatible devices connected to the mLAN Connection Management Server (mCMS) workstation using a Firewire serialbus cable (IEEE 1394 Standard). The mCMS workstation incorporates the Enabler module and each device on the mLAN network runs a vendor specific Transporter module [4]. It is a requirement that all devices on the mLAN network incorporate an IEEE 1394 Node Controller, which enables that device to
communicate with other devices and the mCMS workstation over Firewire on the mLAN network. The patchbay application communicates with the mCMS workstation through a TCP/IP communication socket using XML documents. Advantages of using the client/server approach include the decoupling of connection management between the device and the controlling workstation that runs the client application and the mCMS server [Figure 1]. Communication using generic XML documents also allows for multiple applications developed in any language to connect and communicate with the mCMS workstation. This means the actual studio network can use any medium (physical firewire cables or remote communication if the device running the patchbay has wireless capabilities) as long as the communicating devices can pass messages via TCP/IP [4].

When a user performs a connection management request (e.g. making a connection or a disconnection between two plugs) on the patchbay, a request specific XML document is created by the patchbay and sent to the mCMS server. A typical XML document will carry plug IDs, plug Type and device GUIDs data for two devices to be connected or disconnected [see Listing 1]. On the server side the Request Server module processes the information in the XML document and forwards the request to the Enabler module. The Enabler module accesses the Firewire bus and implements the request with the help of the Transporter modules on each communicating device. Responses are sent back to the client patchbay in XML documents by the mLAN server which in turn is updated appropriately.

III. RESEARCH BACKGROUND

This section discusses current patchbay applications used in high-speed audio/video networking.

A. Current Connection Management Applications Survey

A Connection Management Application (patchbay) in high-speed audio/video networking can be defined as the nerve centre for audio signal routing within the network. The patchbay facilitates connection management by allowing the user to perform many signal patching tasks on a single application [4]. This presents the user with a centralized signal control and management tool for complex networks. Tasks performed on a patchbay include making and breaking audio connections between device plugs, setting/deleting Master/Slave configurations, viewing the topology of the network, editing and maintaining network device properties such as device and plug names, saving/loading routing settings. A patchbay can also be used for general network management activities such as enabling the Network Administrator to create/delete user accounts on the audio network as well as managing network resources.

Many Connection Management Applications have been designed and implemented using third generation languages such as C++ or C for connection management in different audio environments. Patchbays can be categorised into three main groups: List-based, Grid-based and Graphic-based patchbays. List-based patchbays display devices and device plugs in a tree-like structure. List-based patchbay examples include the mLAN Version 1 Devices patchbay [Figure 2] [11] and the PathFinderPC patchbay for Axia audio systems [6].

![Figure 2: Yamaha mLAN List-based patchbay](image)

Grid-based patchbays combine the tree-like structure to display devices and plugs on the network and a Grid-matrix on which connection management tasks are performed. Grid-based patchbay examples include the Otari ND-20 patchbay [Figure 3], the CobraNet™ Manager [3] for CobraNet™ networks that provides the user with a simple real-time matrix of all connected CobraNet™ devices on the network and the Digigram’s ESControl Management Software for EtherSound networks [2].

![Figure 3: OTARI ND-20B Grid-based patchbay](image)

Lastly, there are also Graphic-based patchbays which provide a pictorial representation of the devices and plugs on the network. Graphic-based patchbay examples include the mLAN Version 2 Devices patchbay [Figure 4] [11] that shows all network devices pictorially. Connections are made or broken by simply right-clicking the picture representation of the device plug on the application, and selecting the necessary options on the submenu.
B. Research Objectives

The main objective of this study is to determine if the high-level graphic tool such as Adobe Flash using ActionScript 2.0 can be used to develop complex, interactive applications similar to those developed using third generation languages [section III, part A]. The investigation evaluates the possibility of using Adobe Flash for the development of patchbays that satisfy connection management requirements for two different sound installation environments, namely Broadcast and Studio environments.

Adobe Flash Professional 8 was chosen for this investigation because of its flexible and user friendly authoring IDE that provides the developer with powerful graphic/vector editing tools as well as object-oriented ActionScript for coding complex applications in a minimal period of time.

IV. APPROACH TO THE STUDY

A. Research Methods

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 the internet. This is also called Web-casting or Net-casting. Broadcast networks by their nature span large areas and involve the use of complex mixers and bridge units to route various kinds of audio/video data. Because of the complexity of these networks, network administrators for these networks are usually highly skilled audio engineers who know the network well and understand how the data on the network is routed. Moreover, these networks involve patching of many plugs relative to the size of the network, thus requiring an application that can reveal as many plugs as possible. Smaller networks are usually required for Studio environments. As compared to Broadcast environments which are distributed, Studio environments tend to be located in one place with only one administrator. It follows therefore that they typically deal with fewer connections. Hospitality environment networks are the least complex of the three networks and can be operated with minimal experience and training.

User feedback was integral in acquiring user requirements for the Broadcast and Studio environments. For the Broadcast environments, user requirements were documented based on the existing Otari ND-20 patchbay [Figure 3], and the Otari ND-20 user manual. Unlike Broadcast environments, Studio environments are more flexible with regards to layout and their connection management requirements.

Computer prototypes was sent out to both environment’s users along with corresponding usability questionnaires. The usability questionnaires used was derived from the Software Usability Measurement Inventory (SUMI), the Purdue Usability Testing Questionnaire (PUTQ) [7] and the Questionnaire for User Interface Satisfaction (QUIS) [8] tools. The questions utilised the criteria of learnability, efficiency of use, flexibility, system capabilities (speed, reliability, security and error handling), interactivity, subjective satisfaction, as well as and general and overall reactions. Feedback from the questionnaire assisted with the development of the interfaces.

The Use-Case diagram [Figure 5] defined the functionality included in both patchbay types, while the Object Model [Figure 6] depicts the structural relationships and dynamic interaction between objects written for the patchbays. It is evident from the object model that the applications developed were of sufficient complexity to test the capability of Adobe Flash as a serious connection management application tool. Standard software development and usability engineering principles guided the implementation of the required patchbays. The Singleton and Model-View-Controller Design patterns and Rational Unified Process (RUP -Interactive and Incremental process) formed the basis for the implementation of the software development process.

B. Patchbay Implementation

1. Broadcast Environment

Grid-Based patchbays are commonly deployed within Broadcast environments. This is because Grid-based patchbays allow for a realistic viewing of the Input and Output plugs of all devices on the network [Figure 3]. Furthermore, the connection Grid-matrix clearly displays the state of the connections (connected or disconnected) for each pair of Input and Output plugs which enhances decision-making on the part of the user when performing connection management tasks. The tree-like structure coupled with the Grid-matrix allows for the display of as many devices as possible on the network which fits well with the requirements of Broadcast environments that usually deal with hundreds of plugs at a time.

Listing 1 shows a sample XML document that the client patchbay uses to communicate with the server. This XML document creates a connection between two corresponding plugs on the Grid-based patchbay. Similar XML documents were designed for various tasks.
Listing 1: Connection XML document

Listing 1 connection XML document connects an Output plug with sourcePlugID = 1 (on a device with GUID = "0013f00400011") and is of type "audio") to a Input plug with destinationPlugID of 33 (on a device with GUID = "0013f00400000014").
Initial usability evaluations undertaken by three users at Rhodes University Audio Engineering Lab on the first Grid-based patchbay version 1 [Figure 7] indicated that the application was suited for Broadcast environments but the arrangement of the Grid-matrix boxes was found to be confusing. Columns and Rows needed to be visible inline with tree nodes on both Input and Output trees for it to be easy for the user to read and select a particular Grid box. Modifications were made to the Grid-based patchbay version 1 to incorporate these findings which resulted in the application shown in Figure 8.

A subsequent usability evaluation iteration by two Broadcast users in Canada on the Grid-based patchbay version 2 [Figure 8] revealed the following issues:

- There was a lot of wasted space, that is, features on the application had to be rearranged to enhance space usage.
- The Grid-matrix was found to have a sluggish feel.
- Labels were not consistent for all screens.
- The text input plug connection status box did not work.

The Grid-based patchbay version 2 was redesigned to include the feedback from these usability tests. The resultant application [Figure 9] users regarded as better than the Otari ND-20 patchbay [Figure 3].

2. Studio Environment

A Graphic patchbay [Figure 10] with graphic representation of the devices on the network is the norm for Studio environments. Studio networks are typically smaller than broadcast networks, and Studio Engineers are used to operating on the actual physical devices from which they can plug and unplug cables. The Graphic Patchbay therefore provides them with a virtual studio where they can connect devices in the same manner as with a real studio.
Based on the user feedback from the evaluation of the prototype sent to the users and comments by users from the analysis of the existing Yamaha Graphic patchbay [Figure 4], a Graphic-based patchbay shown in Figure 10 was developed.

ActionScript allowed for the inclusion of complex functionality that could not be implemented easily in C++ for the Yamaha mLAN Graphic patchbay [Figure 4]. Some enhancements to the mLAN Graphic-based patchbay are as listed below:

- Device blocks were implemented in a way that they can be dragged and dropped anywhere within the application space, unlike the Yamaha mLAN Graphic patchbay, where device blocks can only be moved to absolute positions, thus constraining the user.

- Individual Plug blocks were programmed using ActionScript in a way that they can be manually dragged and dropped separately from the main device blocks, to any of the four sides of the main device block, depending on the alignment of the device blocks whose plugs are to be connected [Figure 10].

- Device and plug blocks can be expanded and resized back manually to provide zooming capabilities for better visual display.

- Making and breaking a connection is made easier in the Adobe Flash patchbay compared with the Yamaha mLAN Graphic patchbay [Figure 4]. The user simply clicks the output plug to be connected and drags a connector line to the input plug just the way one would do with actual physical cables in real Studios to make a connection. Breaking a connection is easy; the user simply clicks the line connecting two plugs to be disconnected. The connector line is then automatically deleted thus disconnecting the two plugs.

V. CONCLUSION

The most complex connection management environment application (Broadcast Grid-based patchbay) was adequately implemented in Adobe Flash 8 using Action Script 2.0, leading to the conclusion that Adobe Flash 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 allow for the fast development of graphics, but through using Action Script, provided a powerful way of implementing objects and their associated methods. Features such as the dragging and dropping of device and plug blocks was easily implemented. The client/server architecture and using XML messages enabled the decoupling of the client from the server, thus ensuring language independence and increased flexibility. Adobe Flash, due to its quick graphics creation capabilities introduced the potential of a quick turnaround time for developers. This means computer prototypes can be easily designed, developed, and sent to the users for feedback. This research has indicated that Adobe Flash is an appropriate development tool for connection management application in the field of high speed audio networking.

VI. ACKNOWLEDGEMENTS

This work was undertaken in the Distributed Multimedia Centre of Excellence at Rhodes University, with financial support from Telkom SA, Business Connexion, Verso Technologies, Tellabs and StorTech THRIP, Rhodes Mellon Foundation and the National Research Foundation.

VII. REFERENCES


[Online]: http://www.audioactive.nl/technischmu/Digigram_EtherSound_1.5c.pdf


Phathisile Sibanda holds an Honours Degree in Computer Science from Rhodes University. He is currently studying towards a Masters Degree at the same institution.

Greg Foster is a lecturer at Rhodes University and his research Interests include: Rich Internet Applications, HCI, and Advanced Learning Technologies (e-Learning)

Richard Foss is a lecturer at Rhodes University and his research Interests include: Networking Computer Music Systems, Audio Control, and Multimedia.