Remote Lectures
Final Report

Marc Mentior

June 18, 2004

Supervisor: Ian Harries
Second Marker: Iain Phillips

Abstract

The plan was to produce a system which would allow the remote viewing of lectures and provide facilities to interact with the lecturer. Cameras, microphones and speaker systems were already in place in several of the lecture theatres and would be used.

Some systems which implement aspects of the functionality mentioned and a few closed source systems featuring this functionality already exist.

The goal has been to develop a fully featured and functional open source alternative, utilising proven technology available, such as high compression video streaming so that users with an appropriately high bandwidth connection can "attend" a lecture as effectively as if they were really there.

The project has met many targets set out in the specification and many of the problems associated with this task have been overcome. The system I have produced is functional and allows interaction with a lecturer. There are still several aspects where improvements could be made. These include finding ways of reducing the latency of the video and audio, reducing the bandwidth requirements for audio, improved error handling and robustness as a response to incorrect or unexpected data. Client side authentication and the ability to work through a firewall in a similar manner to that used by QuickTime would also be useful.

Acknowledgements

I would like to thank the staff in the DoC for their input and dedication over the course of the last four years, my supervisor (Ian Harries) for his guidance and time, my family, my friends (Jim, Jake, Darren) and my girlfriend (Christine).

I would also like to thank the authors of the following software without which this project would not have reached the level it has.

- MPEG4IP (especially mp4live and libmp4v2)
- Darwin Streaming Server
- ffmpeg (especially libavcodec)
- jffmpeg
- faad2
- mod_pam for apache

Contents

Chapter 1

Introduction

The objective of this project is to allow lectures to be recorded and streamed live so that they may be viewed both within the department and outside. Whilst being streamed live, the ability to interact with the lecturer may also be provided.

The goals of this project are to allow people to interact remotely with lectures, thereby reducing the effect of a student's illness on his degree, and to allow the lecturers to display a previous year's lecture in the event of their illness. It could also be used to help with lecturer evaluations and to show to prospective students.

Other possible features include allowing students, unable to be physically present, due to distance or travel restrictions, to 'attend' lectures. Lecture courses could be collected on CD or DVD to be sold to other institutions or students.

Several of the lecture theatres have already been equipped with cameras and microphones as well as speaker systems, reducing the amount of initial setup required.

Chapter 2

Specification

2.1 Minimum Specification

Automatic archiving of lectures. The system should record automatically any lecture in the room to the archive. The format in which it is saved should be easily streamable using the technology available.

Searchable archive for later playback. It should be possible to look through and search the archive via a webpage. The webpage should be connected to a database which has information about the lectures available at the time.

Camera control. A webpage or Java application should allow control of the camera.

Distribution (Multicast/Unicast) Upon requests, the videos should be able to be streamed to clients using unicast. For live broadcasts multicast may be used.

Administrative controls on content. Lecturers should be able to control access to their lectures, so that unauthorised students may not access them.

Webbased/Java Applet access site. A webpage should be provided to allows users to access both the search features of the archive and the archived videos themselves.

2.2 Extensions

Live broadcasting and retransmission at specific times. Lectures should be multicast to the Department of Computing (DoC) at the time of their recording and via unicast to external clients who wish to view them.

Rebroadcasts of pre-recorded lectures should also be possible, transmitted in a similar way to live lectures.

Interactivity Students should be able to interact with their lecturers remotely. Features such as the ability to ask questions should be provided.

Remote Lecture Quotas / Exemptions The ability to limit how many remote lectures a student may watch per term and exemptions from the quotas could be implemented.

Lecturers Screen forwarding It should also be possible to include the screens of the lecturer's desktops and laptops in a video stream.

In-vision watermarking Some form of DoC watermark stating the name of the client and the ownership of the video to prevent theft of the material.

Motion Tracking The camera should track the lecturer automatically. Reasonable limits on the effectiveness of this need to be imposed.

Intelligent Connection Speed Detection Increased quality or framerate of video for people with higher bandwidth connections. This is probably not feasible as the video will not be encoded at multiple qualities.

Largely automatic installation and setup of new lecture theatres The software should be able to be installed with a minimal effort to a new lecture theatre. This may have problems as kernel recompilation may be required.

2.3 User requirements

- The site should be easy to use and navigate.
- Minimum number of clicks to achieve objectives.
- Video and audio should be of maximum quality.

- Ability to ask further questions if the student requires it.

2.4 System Design Overview

Figure 2.1 on page 6 shows the overview of the system.

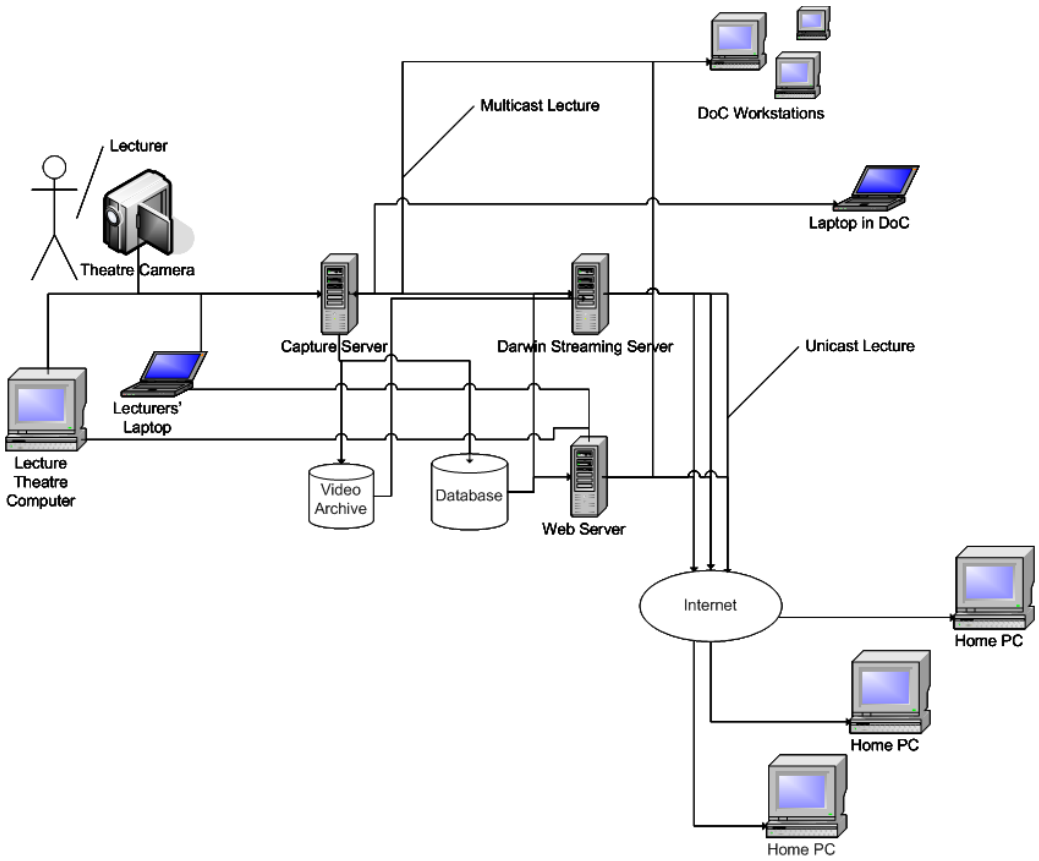


Figure 2.1: System Design Overview

2.5 Client Overview

Figure 2.2 on page 7 shows an initial sketch of the client application.

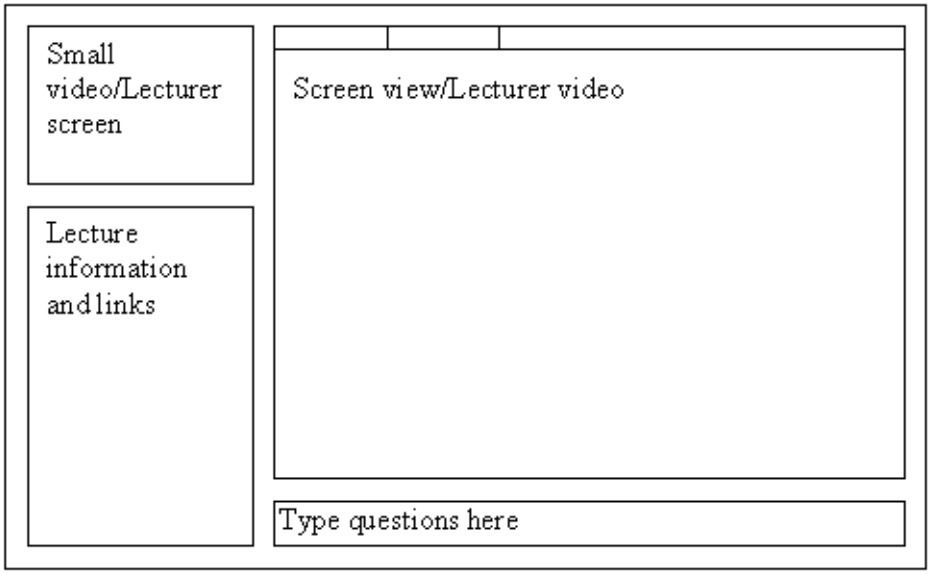


Figure 2.2: Client Overview

The client should be a multidisplay application. The user should be able to switch the Small and Large displays around to suit their preference and the lecture they are watching.

The display features a tabbed dialog allowing the user to select the source displayed in the large window.

The users may type questions for the lecturer at the bottom. When they press enter, the question will be sent.

The lecturer information and links provide the student with information about the type or title of the lecture, who is giving it and any other information the lecturer wishes to provide.

2.6 Implementation Plan

2.6.1 Automatic archiving of lectures

A scheduled job that automatically runs every hour between 9 and 6, which stores the video from that lecture theatre then archives it, giving it an appropriate name (e.g. 311/20040112-0900.mp4)
This should be done with mp4live. It may require kernel recompilation to include video4linux and bt8x8 capture card support.

It may be worth trying both the ISO MPEG-4 encoder and the XVid encoder provided with mp4live. The audio should be encoded in either AAC or CELP. Preferably CELP as it is designed specifically for speech and should require a lower bit rate (allowing more for video).

Reduced frame rates may be required to get decent picture quality at a low enough bit rate for transmission over Broadband. Audio should be sampled at 8000Hz 8-bit mono.

2.6.2 Searchable archive for later playback

Make a program to automatically add database entries for lectures which keeps information about who gave the lecture and the subject matter.

Generate a webpage that allows this information to be searched for lectures on a specific date, lectures by a specific lecturer, lectures from a specific course.

This should be keyed off the room-time-date stamp which will be unique to each lecture.

2.6.3 Camera control

Write an application to use the RS485 protocol to control the camera. Example code can be found on Ian Harries homepage[1].

Design a Java applet that can remotely control a camera using user based authentication.

2.6.4 Distribution (Unicast)

Set up a Darwin streaming server to transmit the archived footage to authorised clients. Darwin has an authorisation module and it should be possible to link this in with the department's authentication servers and the database system. Use the database system to store the list of users allowed to view each lecture. Groups of users should be allowed and two custom extra levels

of `alldoc` which let any valid DoC user and `all` which lets anybody access the stream.

Set up Darwin to allow content to be relayed from a live stream. Check the authentication applies to this. If it does not, the necessary steps to remedy this should be taken.

Update the webpage to allow users outside the DoC to authenticate and start watching the stream through a Java Applet Viewer.

If there are problems with the Java Applet Viewer find an alternative way of viewing the stream. VLC from Videolan.org is capable of viewing but would not be the ideal solution as it needs to be installed on the clients computer and can be unstable. QuickTime viewer may also work for MP4 files.

2.6.5 Distribution (Multicast)

Setup mp4live to multicast the live video around the department. Obtain information from the DoCTV group about which multicast IP should be used so that it will not interfere with their system if possible.

Design the webpage to allow users currently inside the DoC to watch the multicast lecture.

2.6.6 Administrative controls on content

Design a webpage to update the access control lists in the database referring to who may watch lectures. Create a program which will pop up on a lecturers machine when they login, informing them of the current state of the settings and allowing them to change it.

2.6.7 Live broadcasting / rebroadcasting at specific times.

Create a webpage to allow rebroadcasting of lectures to multicast ips at specific times.

2.6.8 Interactivity

Add an extension to the Java viewer applet which allows students to ask questions to the lecturer, and possibly communicate with other students who are also watching the lecture. When a question is asked a popup should appear on the lecturers' screen and produce a noise to indicate there is a

question. Questions should include information about who sent them, so misuse of the system can be traced.

2.6.9 Lecturers' Screen forwarding

Create a Java application to sit on the lecturers' desktop, which will capture the screen and transmit in high quality the pictures to the clients. This information should be archived in separate files and the database should link this to the main lecture. (Filenames like 311/20040112-0900-screen1, screen2...). The frame rate for this can be quite low - 1 frame per second or lower should suffice. Videos can be put on the lecturers' website if they wish anything faster.

2.6.10 Remote Lecture Quotas / Exemptions

Log usage of students remote lecture watching and check they have not exceeded a predefined per user/group/year quota in the database. Allow exceptions and default to be programmed via a webpage.

2.6.11 In-vision watermarking

If possible create an intermediary program which generates a watermark saying the video is the property of the Department of Computing Imperial College and moves it around the video changing colour throughout the course of the lecture.

If possible, when a client is retrieving the video via unicast add a visible moving watermark that displays the client's username on the screen.

2.6.12 Intelligent Connection Speed Detection

This could possibly be used if encoding at different bit rates proves possible, allowing the client to choose to watch a video and the server to pick the highest bandwidth the client can receive without large packetloss.

2.6.13 Largely automatic installation and setup of new lecture theatres

Package the entire system in one or more easy to install rpm or tgz files. Provide a HOWTO or INSTALL file for information regarding changes to the kernel that may be required or configuration file options that are system specific.

Chapter 3

Background

The information included in this section is designed to provide a general overview of the state of technologies and display alternatives to the chosen specification should problems arise.

Distance learning has been around for a long time. The best known example is the Open University[2] television programs. The Open University has been providing remote learning courses since 1969 via TV, radio, books and tapes. More recently, they have begun to use the Internet[3] to provide access to course material including virtual tutorials and discussion groups.

3.1 Existing Systems

There are several existing systems available on the Internet, which provide the features needed to provide an interactive remote lecture successfully. However, many of these systems require licensing, are closed source or only provide full functionality in conjunction with other systems.

The University of Geneva[4] had a fully featured system based on RealVideo[5], a closed source technology. Making extensions to this system would prove difficult or impossible. However, it does show that the theory has promise.

Another example system was found in Singapore[6] where a system had been set up using NetMeeting[7], allowing both sides to see each other. This system, however, had high bandwidth requirements and slides and work had to be e-mailed between the students and lecturers.

Interactive lecturing systems have typically required high bandwidth. With the advent of Broadband Internet for a low cost, more students are finding this a feasible way to be connected to the Internet.

Broadband connections such as cable and ADSL offer connection speeds of 10+ times faster than a traditional phone based modem.

3.2 Features which currently exist

- Two way video communication
- Virtual white boards
- Screen video capturing systems
- Broadcasting of live content to large audiences

3.3 Colourspaces

This section is to give an idea about the different colour spaces used in video encoding and manipulation on computers.

3.3.1 RGB

RGB stands for Red, Green, Blue and this is how colours in this model are represented.

RGB 24-bit representation

Each component R, G or B is represented by a value between 0 and 255. This gives possible combinations of 16,581,375 different colours. Colour monitors use red, green and blue sub-pixels to display their information, hence this seems a logical choice of format to store data in. However, this means that each pixel will take up 3 bytes of space. There are other methods of storing RGB which require less space, such as 16-bit RGB where only 5 bits (or sometimes 6 for green) are used for each pixel, or 8-bit where a palette is used to determine what each value actually represents. [8]

3.3.2 YUV

YUV is another colour space which represents a pixel in a completely different way to RGB. Y is the luminance (brightness) component of the model and U and V are chrominance (colour) components of it. YUV was designed to allow analogue broadcasts of TV to be made in both black and white and colour. In television the Y component is the black and white picture and the U and V components are the colour which can be broadcast on a separate frequency, thereby allowing both black and white and colour televisions to coexist.

YUV has an advantage other than being useful for broadcasting old TV signals. It has been shown that the human eye detects changes in intensity more readily than changes in colour between pixels. As a result of this, it is possible to sample the U and V components at a lower resolution than the Y component with very little visible degradation. [9]

YUV 4:4:4

YUV 4:4:4 takes the same amount of space as 24-bit RGB because it samples the Y and U and V components at the same, full, resolution. Hence, an image displayed using this format should appear identical to one displayed using RGB.[10] [1]

YUV 4:2:0

YUV 4:2:0 is a variant of YUV 4:4:4. However, instead of having the same number of U and V components, it has 1 U and V component per 4 pixels.

There are two ways of representing this; one is the interleaved way and the other is the planar way.

Planar version. In the planar version of YUV 4:2:0, all the Y components are given first, then the U components (1 sample per 2x2 pixels), then the V components (1 sample per 2x2 pixels). Hence there is a plane of Y, a plane of U and a plane of V, where the U and V planes are a quarter of the size of the Y plane. This format is also known as YUV12 or YV12.

An example is given below[11]:

> The layout for a 4x4 image is:
>
> Y01 Y02 Y03 Y04 Y05 Y06 Y07 Y08 Y09 Y10 Y11 Y12 Y13 Y14 Y15 Y16 U1 U2 U3 U4 V1 V2 V3 V4
>
> The components map onto the image as follows:
>
> [Y01 U1 V1] [Y02 U1 V1] [Y03 U2 V2] [Y04 U2 V2]
>
> [Y05 U1 V1] [Y06 U1 V1] [Y07 U2 V2] [Y08 U2 V2]
>
> [Y09 U3 V3] [Y10 U3 V3] [Y11 U4 V4] [Y12 U4 V4]
>
> [Y13 U3 V3] [Y13 U3 V3] [Y12 U4 V4] [Y16 U4 V4]

[1] The conversion between YUV to RGB and RGB to YUV is not perfectly lossless due to rounding of fractions.

RGB to YUV conversion

The equations for RGB to YUV conversion are given below. As demonstrated they multiply each of the R,G & B components by small fractions. Due to the way computers represent floating point numbers this will result in rounding errors, and thus, interchange between the two formats should be kept to a minimum.[12]

Y			=	0.299R	+	0.587G	+	0.114B
V	=	0.713(R - Y)	=	0.500R	-	0.419G	-	0.081B
U	=	0.564(B - V)	=	-0.169R	-	0.331G	+	0.500B

3.4 Video Formats

3.4.1 MPEG-1

The MPEG-1 (Moving Picture Experts Group) standard is a set of requirements that a compatible decoder should implement to decompress a video stream. The standard specifies ways in which algorithms may be used to compress the video stream and not how they should be implemented. It defines the format the files should take. MPEG-1 is used for the Video CD format and has a target bitrate of 1.5 Mb/s for video and audio. [13]

3.4.2 MPEG-2

MPEG-2 extends MPEG-1 by increasing the allowed bit rates and adding more algorithms which can be used to compress the video. It is backwards compatible with MPEG-1 which means that any hardware which could decode MPEG-2 can decode MPEG-1. MPEG-2 is the format used by both Digital TV and DVDs. It defines several Levels, each of which is of a different quality. It also defines several profiles which an encoder or decoder can choose to meet. A profile is defined such that it is an extension of the profiles below it. The use of different profiles can greatly simplify the creation of a decoder or encoder as it allows specific bits to be left out.[14]

3.4.3 MPEG-4

MPEG-4 is another extension of the MPEG standards. It improves upon MPEG-2 and allows much lower bit rates than MPEG-2 whilst still maintaining high quality. MPEG-4 is probably best known under the name of

DIVX, which is an MPEG-4 compatible codec, originally based on Microsoft's MPEG-4 codec designed for use with their ASF file format. [15]

The MPEG-4 specification (ISO/IEC 14496) consists of 16 parts. These are Systems, Visual, Audio, Conformance testing, Reference software, Delivery Multimedia Integration Framework (DMIF), Optimized reference software for coding of audio-visual objects, Carriage of ISO/IEC 14496 contents over IP networks, Reference hardware description, Advanced Video Coding (AVC), Scene description and application engine, ISO base media file format, Intellectual Property Management and Protection (IPMP) extensions, MP4 file format, Advanced Video Coding (AVC) file format and Animation Framework eXtension (AFX).

The parts which were of use in this project were Systems, Visual, Audio, Carriage of ISO/IEC 14496 contents over IP networks, ISO base media file format and the MP4 file format.

As with MPEG-2, different profiles are defined which specify subsets of the MPEG-4 Systems, Audio and Visual standards. Applications can then be designed to implement these. In the case of Visual, there are now currently 19 profiles, compared with about 4 in MPEG-2.

MPEG-4 scenes are composed of audiovisual media objects. These scenes are organized in a hierarchical fashion. The most primitive types in this hierarchy are objects such as:

- Still images.
- Video objects.
- Audio objects.
- Text and graphics.
- Synthetic sound.

Media objects are composed of several elements that allow the handling of the object within a scene. In coded form each media object is independent of other objects. As such, the coded representation is of that object alone. Coding of each media object is specialised to be as efficient as possible. [16]

3.4.4 dirac

δirac is a wavelet based codec from the BBC. Wavelets are a new way of encoding video which could be viewed as being similar to progressive images in JPEG, as the more information you obtain from the source the better the image quality becomes. This codec is still in the early stages but could prove

quite interesting if it is made fully functional. Unfortunately, in its current form it seems rather unstable and unable to both compress or decompress in real-time on modern home systems. [17, 18]

3.4.5 theora

Theora is the video counterpart to Vorbis, it is based on the VP3 codec. Theora is currently also still in alpha but promises to be a patent free video codec. Every other format mentioned here requires licensing and possibly royalties for its implementation and use. Once this reaches a more advanced stage it could possibly prove to be a preferable alternative to MPEG-4. See also Vorbis in section 3.5.4 on page 17 [19, 20]

3.5 General Audio

3.5.1 Audio Formats

There are several standards to compress digital audio. These include:

3.5.2 MP3

MP3s files are the result of an audio compression technique. Their full name is MPEG-1 Audio Layer III. This compression system provides approximately 10:1 compression with a low loss in quality. [21]

3.5.3 AAC (Advanced Audio Coding)

AAC was originally designed as an audio compression technique for the MPEG-2 standard. AAC was also known as NBC (Non Backward Compatible) due to the fact it was incompatible with the old MPEG-1 audio formats.

AAC audio is used in MPEG-4 files and provides the best quality currently available for MPEG-4 audio.

AAC is able to handle many more channels of audio than MP3 as it can handle 48 full audio channels and 16 low frequency enhancement channels. This is in comparison to the 5 full audio channels and 1 low frequency enhancement that MP3 provides. It also supports higher sampling frequencies than MP3. [22, 23, 24]

libfaad is an AAC decoder. It is currently available in two forms, either its new form, which is distributed in the FAAD2 package or its old form, distributed in the FAAD package. FAAD2's implementation is a large rewrite on the original and has been shown to give better performance.

3.5.4 Vorbis

The Vorbis (a.k.a Ogg Vorbis) audio codec provides very high quality sound at low bit rates and its quality is substantially higher than MP3. Ogg Vorbis can provide higher quality audio than AAC. However, it is not MPEG-4 compatible and, as such, can't be used in an MPEG-4 file or stream. Ogg Vorbis is unusual in that it is a completely free, open source and unpatented codec. [25]

3.6 Speech Specific

The codecs previously mentioned are all designed for generic audio. Generic audio is the sort of audio you would get in a film or in music. Speech specific codecs are designed specifically for the patterns of a human voice. These types of codecs are used in digital phone transmission.

3.6.1 Speex

Speex is a patent-free audio compression format available from the same people who make Ogg Vorbis. It is designed to offer high quality speech with minimal bitrates. [25]

3.6.2 Global System for Mobile telecommunication 06.10

Global System for Mobile telecommunication (GSM) 06.10 is the low bandwidth speech encoding system defined by the GSM digital mobile phone standard.[26]

3.6.3 CELP

CELP is an MPEG-4 compatible speech codec. It also supports multiple low bit rate streams in a single stream. This means that if all the data is there, a higher quality stream will be played; if not, a lower quality stream will be played. [27, 28]

This would be the preferred format for recording lectures as these would require a lower bitrate as they are usually entirely speech. However, due to a lack of decoding and encoding libraries support, it was deemed unfeasible to use this.

3.6.4 iLBC (Internet Low Bitrate Codec)

iLBC is a FREE speech codec suitable for robust voice communication over the Internet. [29]

3.7 Video Streaming

Traditionally, downloading of video clips from the Internet meant that users had to wait until they had downloaded the complete file before they could watch it. However, in the early 1990s, a new way of distributing videos started being used - streaming. Streaming videos means that the video can start being watched after only a small part of it is sent to the client, as all the information needed to decode the video & audio is included in that part. Usually, streaming video is buffered before being played to reduce the effect of network traffic on the client side. Sometimes streaming video is provided in variable rates of compression to support speeds of connection. Typically the client will have to download some software to decode the streamed video. This is usually freely available, but the server software usually costs money and does not provide the source code. [30]

3.8 Streaming Software

Currently there are 3 main video streaming solutions in use

- Microsoft Windows Media Services[31]
- Real Video / Helix Server[32]
- QuickTime / Darwin Streaming Server[33]

Each system is designed around its own file type.

- Windows Media Services are designed to stream Microsoft's ASF files.[34]
- Real Video / Helix Server is designed to stream Real Media files.[5]
- QuickTime is designed to stream Apples QuickTime MOV file format.[35]
- Darwin Streaming Server, see section 3.8.1 on page 19.

3.8.1 Darwin Streaming Server

Darwin Streaming Server is an open-source version of the QuickTime streaming server, which is designed to be run on operating systems other than MacOS. Darwin Streaming Server also supports MPEG-4 streaming.

Darwin Streaming Server is based around a core server module which loads up multiple threads. It also extensively uses modules to perform its

actions and will call these different modules when their registered roles are being performed.

When a module is initially loaded its main routine is called. This sets up its dispatch routine. Whenever the module is needed to perform a role, its dispatch routine will be called. The first role the module must perform is the register role. All modules perform the register role. The purpose of the register role is to set up all the static attributes and to call the `QTSS_AddRole` for all the roles this module wishes to support. It can also use this time to register services.

Once this has happened, the server will then call the initialise role. The purpose of this role is to set up any module specific variables or data structures.

The last role to be called is the Shutdown role. This allows a module time to clear up its data structures and free any dynamically allocated memory.

Other roles which may be called in-between include the Reread preferences role, the Authenticate role and the Authorise role. These are the main roles which were used in the access control module. There are several other roles which can be performed all related to specific tasks. A detailed overview of these can be found in QuickTime Streaming Server modules documentation.[36]

3.9 Streamable file formats

Not all types of video file can be streamed. Certain file formats have been designed with the idea of streaming in mind. Streamable formats include MPEG 4, Advanced Systems Format (ASF), OGG bitstreams, Apple MOV and RealMedia (RM). [14, 13, 15, 37, 38]

3.9.1 ASF (Advanced Systems Format)

ASF is an extensible file format primarily designed for the storing and playing of media streams over networks.

ASF is a container format for Windows Media Audio and Video-based content. ASF files can contain multiple independent or dependant streams. These include streams such as multiple audio streams, or multichannel streams or can include multiple bitrate streams for transmission over different bandwidth media. The streams can be in any compress or uncompressed format.

ASF streams can also contain things such as text, webpages, scripts or any other data.

The allowance of scripts in the ASF format has been the cause of many potential security violations and as a result is not favoured by many services. [34]

3.9.2 MP4 File Format

The MP4 file format is an extension of the ISO base media file format which was originally based on the QuickTime file format. The main difference between the QuickTime format and the ISO base media format is the use of 64-bit numbers to represent lengths. The ISO base media file format has been designed to contain media for a presentation in a flexible and extensible way. It has been designed to facilitate the presentation, interchange, management and editing of the media.

The ISO base media file format structures files in an object-oriented way. Because of this, a file can be split into its component objects very easily. The structure of objects is defined by their type. Each object is known as an atom and usually has a 4 byte code associated with it. These 4 byte codes are usually composed of lower case letters. For example, a Movie Box which contains general information about a Movie being shown is known as a 'moov' box. 'moov' is an atom which can contain sub-atoms. Each atom has a defined structure which either has specific fields in it or allows other atoms to be contained within it.

There are two base atom types from which every other atom is derived. The first atom contains a 4 byte size code followed by a 4 byte type. The size code is the number of bytes the box takes up including the size code and type field. There are two special size values 0 and 1. A value of 0 represents that the box continues to the end of the file and a value of 1 represents that the size is specified using the first 8 bytes after the type.

The other base type is an extension of the previous atom. It adds support for a version field (1 byte) and a flags field (3 bytes). Typically, a version of 0 represents that this box uses the 32-bit variant of this box and a version of 1 represents a 64-bit variant. The flags are atom dependant. Notably, not all boxes support the use of the version flag to indicate a 64-bit variant and instead a separate atom type is defined. An example of this is the 'stco' and 'co64' atom types. Both represent the chunk offsets. This is a partial offset within a file to a chunk of data. Chunks usually contain multiple access units or samples. Other atoms define which samples appear in which chunks and the samples offset within the chunk.

The file format has been designed to be independent of any type of network protocol, although it has been attempted to make it provide efficient support for network protocols in general.

The MP4 File format extends the ISO base media file format with a few extra atom types and a few specific use cases. In an MP4 file media data is stored as access units. Access units are things like an audio sample or a video frame. Access units can be joined together to form larger units.

This facilitates the process of generating hint tracks. Hint tracks are used to specify how an audiovisual track should be fragmented for presentation over a network. [39, 40]

3.10 RTP (Real-time Transport Protocol) RTCP (Real-time Transport Control Protocol)

RTP is a protocol defined in rfc1889 for the transport of real-time data. Examples of real-time data include audio and video. It can be used to provide streaming video, as well as interactive services through the use of RTCP. RTP is made up of two parts, a data part and a control part.

Each stream (e.g. video or audio track) is transmitted separately and it is the job of the decoder to combine them properly at the client side.

RTP has been designed to provide support for the detection of lost information, security, and content identification. It also can contain information to help rebuild the timing of a video or audio stream.

RTCP has been designed to monitor the quality of service and to provide information about which clients are engaging in a session

RTP does not provide any quality of service (QoS) or timeliness guarantees, it relies on lower-layer services for this. It does not provide any guarantees about the order of packet delivery, or guarantee they will be delivered. The sequence numbers in the RTP header allow the correct order of packets to be determined.

RTP is usually transmitted using the UDP/IP protocol. However, it is transport independent so it may theoretically be used over any protocol. For this project, UDP/IP should suffice as it allows for both unicast and multicasting of packets. [41, 42, 38]

3.10.1 RTP Packet

Figure 3.1 shows the layout of an RTP packet.

3.10.2 MPEG-4 Video over RTP

RFC 3016 defines a protocol for which MPEG-4 Video and appropriately encoded audio can be broadcast in a consistent way over RTP. The audio standard defined is Low-overhead MPEG-4 Audio Transport Multiplex (LATM). However, this is not the format that MP4Live or Darwin Streaming Server use

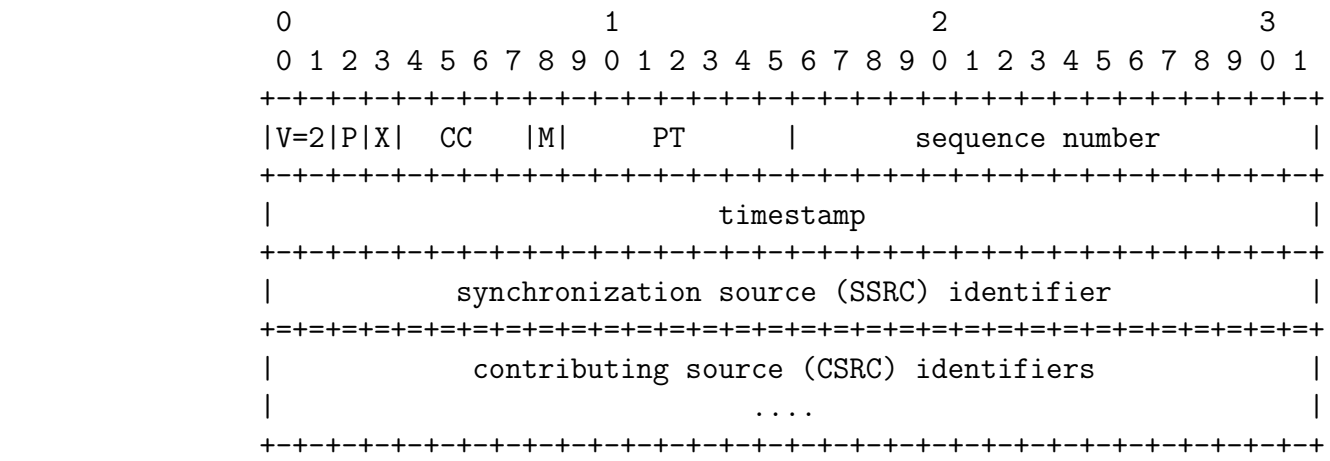


Table from RFC1889[41]

V version – 2 bits

P padding – 1 bit

X extension – 1 bit

CC CSRC count – 4 bits

M marker – 1 bit

PT payload type – 7 bits

Sequence Number sequence number – 16 bits

Timestamp timestamp – 32 bits

SSRC SSRC – 32 bits

CSRC CSRC list – 0 to 15 items, 32 bits each

Figure 3.1: RTP Packet

and, instead, the method described in RFC 3640 is used. See section 3.10.3 on page 25.

The RFC suggests that, if a frame of data is small enough to fit in an RTP packet without that packet being fragmented, then it should be. If subsequent frames can be added to that packet, without it resulting in fragmentation, they may also be added to that packet. If they would cause fragmentation, they should not be added and a new packet started. A packet that contains a fragment of a packet should not contain fragments of another packet or even a complete packet. If a packet requires fragmentation, it should be split into sequential packets, all with the same timestamp. A packet which contains only complete frames or the last frame of a fragmented packet should have the RTP marker bit set.

Timestamps and sequence numbers should start at a random offset for security reasons. [43].

3.10.3 MPEG-4 Audio over RTP

RFC3640 defined a way to transmit MPEG-4 elementary streams over RTP. Although it describes video as well, the method described earlier in section 3.10.2 on page 23 is normally used. However, the default configuration of parameters allows for RFC3640 to be interchangeable with the format described in RFC3016.

RFC3640 defines several specific modes of transmission. Generic, constant bit-rate CELP, variable bit-rate CELP, gigh bit-rate AAC and low bit-rate AAC.

Note: Audio is broadcast in its generic mode by Darwin Streaming Server.

This format splits the stream up into Access Units (samples in the case of audio). It will then follow similar guidelines to video with regard to packet splitting, but it also may include access unit headers to describe information about the data such as the size, the decoding time and the order of the samples within the packet. The settings that determine which data is contained within the headers is transmitted using `out-of-band` means. In the case of Darwin Streaming Server, this is included in the SDP in the `fmtp` attribute of the appropriate track. This allows for variable size headers and variable presence of the attributes on a per stream basis. This also means that excessive headers need not be broadcast, as, without the use of `out-of-band` means, the headers would have to be included always. [38]

3.11 RTSP (Real-Time Streaming Protocol)

RTSP is designed to be a control system for remote streaming sources. For example, it is possible to use RTSP to control the stream to an individual client, allowing the client to seek to a point in the stream or pause/stop the stream. This, of course, only works where the source is able to be fast forwarded or stopped, unlike a live broadcast. RTSP relies on RTP to deliver the content to the user. [44]

3.12 SDP (Session Description Protocol)

RFC 2327 is used as a way of describing a presentation that can be sent over RTSP. SDP is a protocol for the description and advertisement of multimedia conferences. It is used to specify which tracks are available and any related parameters. It also specifies general information about the sources. [45]

3.13 SMIL (Synchronized Multimedia Integration Language)

SMIL is a markup language for laying out video presentations. SMIL can be used for things such as interactive video, video on demand and online training. SMIL can be used to create dynamic multimedia presentations with multiple synchronised sources of media and adjust their layout. [46, 47]

3.14 Multicast, Broadcast and Unicast

Unicast is a system whereby an IP packet sent from a machine is sent to a specific machine's IP. (e.g. 192.168.0.3). Figure 3.2 on page 26 gives an example of this.

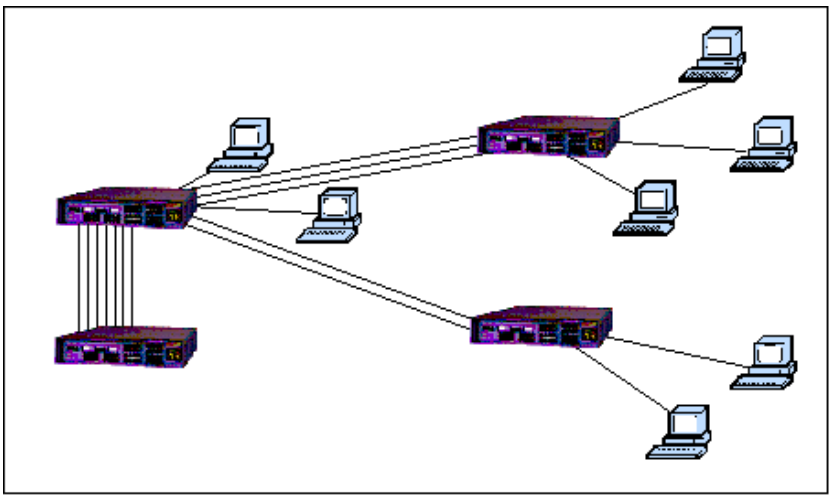


Figure 3.2: Basic Unicast Service [48]

Broadcast is a system whereby an IP packet sent from a machine is sent to a specific group address which all machines listen out for (e.g. 192.168.0.255 is sent to all machines with an IP address of 192.168.0.1-254). Multicast is more efficient than unicast or broadcast as it sends to a group of machines that choose whether they wish to receive the packet or not (e.g. 239.1.2.3 would send to any machine that was listening for it). Multicast is very suitable for broadcasting live content as all clients will be at the same point at the same time. Figure 3.3 on page 27 gives an example of this.

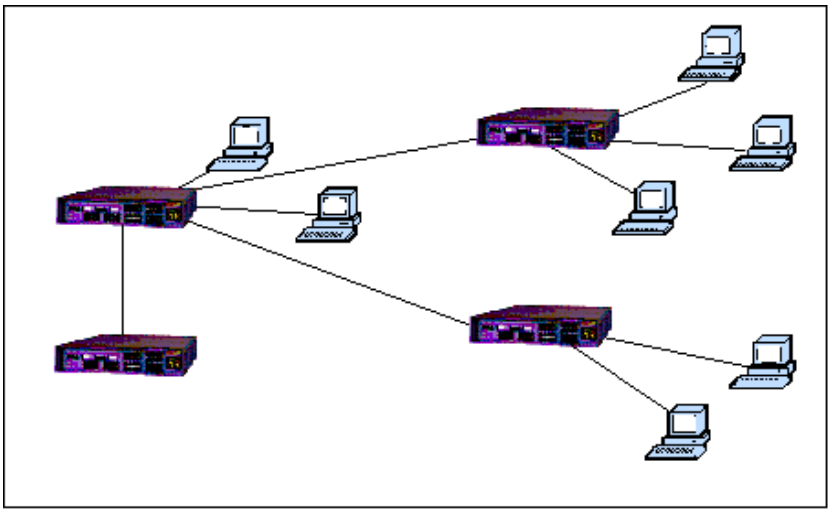


Figure 3.3: Multicast Transport Service [48]

3.15 Camera Control

The camera in the labs is a WV-CSR400 made by Panasonic and is controlled via an RS-485 interface.

3.15.1 RS-485

RS-485 has similarities with RS-232 (which is used by COM/serial ports on PCs). The most significant is the way they actually transmit data which gives RS-485 a range of about 1.2km compared with about 30m for RS-232. RS-485 uses twisted pairs of wires. The pairs carry opposite voltages, one positive the other negative. The signal is inactive in this state. When the polarities are reversed the signal is deemed active. This allows the camera to be physically far away from the machine which is recording and controlling the video.[49]

More information on this can be found in Appendix C on page 83

3.15.2 Protocol

The camera was sent commands over the RS-485 connection. All the commands had the following format

STX command ETX

where STX and ETX are 0x02 and 0x03 respectively. "*command*" is one of several formats. The most common format is GCx:*bytes* where x represents the number of bytes following. Separate bytes are designated by the inclusion of a colon between them. Upon receipt of a command an ACK response is sent. ACK is 0x06. On newer cameras a confirmation of the command may also be sent. Other commands should perform specific functions which are not directly related to camera control. A more detailed description is given in the Panasonic Protocol Information document, although it should be noted that not all commands apply to this model of camera. [50]

3.15.3 Motion Tracking

There are several techniques for performing motion tracking. The most common techniques are based on block matching of frames assuming a 2 dimensional translational model for each block. A distortion function is used to calculate how similar the source and target blocks are. Example functions include mean absolute difference (MAD) and mean square difference (MSD). The Equations for these are shown below.

$$MSD(dx, dy) = \frac{1}{mn} \sum_{p=1}^{m} \sum_{q=1}^{n} [A(p, q) - B(p + dx, q + dy)]^2 \quad (3.1)$$

$$MAD(dx, dy) = \frac{1}{mn} \sum_{p=1}^{m} \sum_{q=1}^{n} |A(p, q) - B(p + dx, q + dy)| \quad (3.2)$$

A and B are the source and target blocks and (dx, dy) is the motion vector. MSD has been proven to be more accurate than MAD.

Other matching systems include Pel Difference Classification (PDC) and Integral Projection (IP) The equations for these are shown below.

$$PDC(dx, dy) = \sum_{p=1}^{m} \sum_{q=1}^{n} [|A(p, q) - B(p + dx, q + dy)| \leq t?1 : 0] \quad (3.3)$$

$$IP(dx, dy) = \sum_{p=1}^{m} \left| \sum_{q=1}^{n} A(p, q) - \sum_{q=1}^{n} B(p + dx, q + dy) \right| + \sum_{q=1}^{n} \left| \sum_{p=1}^{m} A(p, q) - \sum_{p=1}^{m} B(p + dx, q + dy) \right| \quad (3.4)$$

Searching through all possible motion vector candidates would take an extremely long time, so there are several faster but sub-optimal search strategies which may be employed. These include the Orthogonal Search Algorithm (OSA) given below

Guang-Zhong Yang's[51] multimedia notes give this as: "Given a center [cx, cy], test 3 points: [cx, cy], [cx-s, cy], [cx+s, cy]. Let [a, b] be the best match, test 3 points: [a, b], [a, b+s], [a, b-s]. Take best match as new center, set s = s/2, repeat."

Not all motion vectors for an object agree due to mismatches from similar segments of an image. For this reason hierarchical strategies can be used, which involve finding the motion vectors of low resolution images and then propagating these to higher resolutions.

3.16 WinTV PCI Capture Card

The WinTV PCI Capture Card is a Bt848 based card. These cards are supported under most operating systems including Linux. This will be used as the source for the video as it will be connected to the camera. [52]

3.17 3rd Party Software

3.17.1 MPEG4IP

MPEG4IP is a set of tools designed for encoding MPEG-4 files under Linux. A useful tool in this set will be mp4live which allows a video source to be streamed to a file or over a network or both. [53]

MP4Live

MP4Live was a tool which was included within MPEG4IP which was designed for the encoding and broadcasting of video and audio across a network. It was written in C++ and allowed both capture to a file and simultaneous streaming across a network. It made extensive use of interfaces, allowing extra modules to be easily added. However, no plugin code was used and modules need to be explicitly called. Configuration data is also centralised and any extra items need to be added to a central configuration module.

3.17.2 Java Media Framework (JMF)

JMF is an API designed by Sun for use in Java. It enables applications and applets to use audio and video and other forms of media relatively easily. [54, 55]

Sun and IBM jointly provided a reference implementation for JMF, which allowed it to be easily extended by writing classes which implement their Codec interface. Other types of extension, such as different players and data sources are also possible.

JMF features an application which allows you to register your own namespaces. Once registered, the JMF Player will attempt to search these namespaces for plugins compatible with the type of media it is trying to decode.

The Java Media Framework is split into 3 main areas: input, data processing and output. Input consists of getting data from a source, for example, reading from a file, receiving over a network or capturing from a live source. Processing usually involves converting between formats, compressing or decompressing and apply effects to the media. Finally output, usually involves

displaying the media (on the screen, through speakers), saving to a file or sending over a network.

Several types of processor exist. The most common ones are Demultiplexers and multiplexers, Depacketisers and packetisers and codecs.

Demultiplexers and multiplexers either extract individual tracks from a combined source or combine multiple tracks into a single output. An example of a demultiplexer is something that reads a file and outputs separate audio and video tracks from it.

Depacketisers and packetisers retrieve information, for example, from RTP and decompose this information into separate access units or frames. These are then sent on to the codec for decompression. A packetiser performs the reverse process of a depacketiser and takes encoded samples and splits them up into packets for transmission.

Codecs decode or encode information from one format into another. For example, a decoder could convert from MPEG-4 video into YUV 4:2:0 planar format.

IBM provides a demonstration version of an MPEG-4 decoder on their website, but this suffers several limitations. Firstly it a 30 day trial and presents a logo over the image. Secondly the source code for this is not available and thus it cannot easily be extended.

JMF also provides support for RTP and RTSP. The RTSP implementation is quite new and is still not fully developed. The RTP implementation is closed source making debugging any problems that occur within it difficult. However, the RTP implementation is quite well documented and well tested as it has been a part of JMF for some time. The RTSP implementation, however, suffers from a lack of documentation for anything other than its basic usage.

3.17.3 QuickTime

Quicktime proved to be an invaluable tool in the testing of the system. Its support for MPEG-4 video and audio, RTSP authentication and RTP over RTSP all were extremely useful in testing the system and tracking down errors.

QTJava

QTJava is a Java wrapper for QuickTime that allows it to be embedded in Java applications. However, it is only available for Windows and Mac, as it relies on native libraries.

3.17.4 FFmpeg

FFmpeg is a tool which provides format conversion between many different video and audio formats. Within this project is a library called libavcodec. Libavcodec provides FFmpeg with its functionality. [56]

JFFmpeg

JFFmpeg is a JNI wrapper for FFmpeg. It was originally designed to support H.263 which is a subset of the MPEG-4 specifically for low bandwidth video conferencing. Its use of FFmpeg made it ideal for a wrapper for MPEG-4. [57]

3.17.5 PAM (Pluggable Authentication Modules)

PAM provides an API for programs wishing to authenticate users. It relies on the use of modules to perform its authentication. These modules are specified in a configuration file that is program specific. This allows relatively easy addition of authentication to programs via its callback interface.

3.17.6 libpq

Libpq is Postgresql's C wrapper. It allows an application to connect to a database to execute queries and to perform actions based on the results. It has a few relatively straight forward commands.

`PQconnectdb` - This function accepts a connection string and will return a `PGconn` resource on success.

`PQstatus` - Will return `ConnStatusType` information about supplied connection.

`PQexec` - Will execute a query on a given connection and return a `PGresult` pointer.

`PQclear` - Used to clear up the memory used by a given `PGresult`.

`PQresultStatus` - Returns a `ExecStatusType` for a given `PGresult`. This allows you to determine the success or failure of a query and whether or not any results were returned.

`PQescapeString` - Returns an escaped version of a string such that user input cannot break a query.

More information can be found on the postgresql website [58]

3.17.7 gSOAP

gSOAP is a C++ library and compiler for SOAP based web services. It will produce the WSDL file needed from a C header file and provides libraries to easily implement a fully featured SOAP client and server.

SOAP (Simple Object Access Protocol)

SOAP is a protocol for exchanging information. It is based on an XML based protocol. It defines a framework for describing what is in a message and how to process it. It defines a set of datatypes, encoding rules for representing user defined information and a convention for representing remote procedure calls and responses.

It is typically used in combination with HTTP to produce a web service. This is where a server provides a defined interface to which clients connect. This web service can be used for things such as remote processing of data, retrieval of data from the service or executing a predefined function on a server. [59]

3.17.8 STunnel

STunnel is an application which can act as an SSL wrapper for any program. It will listen on a port for an SSL connection. It will then accept and decrypt this connection, forwarding the data on to a local or remote port. STunnel can also be used in the opposite direction to connect a client which does not support SSL to connect to an SSL enabled server.

Used in combination, a SSL Tunnel can be created between two programs, neither of which support SSL.

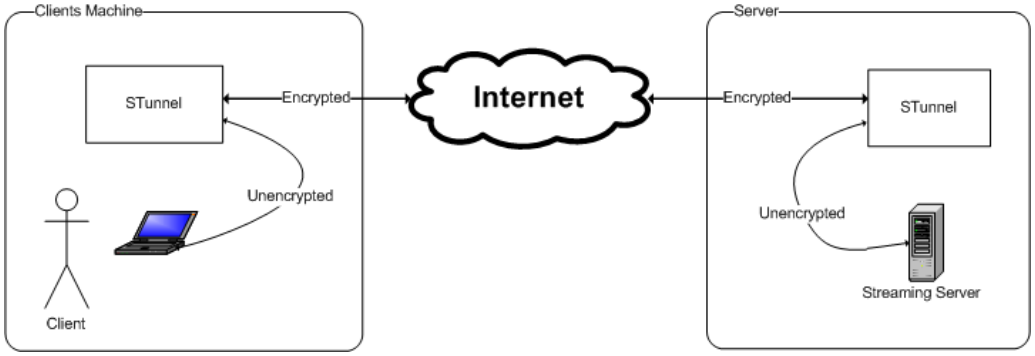


Figure 3.4: STunnel End-to-End Encryption

Hence, password authentication is more secure. A small download with a script to start the appropriate STunnels on a user's machine could be made or possibly this could be part of the client application. A small problem with this may be the way it rewrites the "from addresses" at the server end. This would mean that the only method of transmission was using RTSP over RTP.[60]

Chapter 4

Testing

Each stage of the project should be checked as it is completed, and when other stages are completed they should be checked to see that they have not had unwanted side effects on any other stage.

An unfamiliar user should be given the system to test its ease of use. Feedback from the user about design issues should be taken and used to improve the project.

The authentication system should be checked for security and to ensure it does not result in plaintext passwords being transmitted over the internet nor allow ways around it by choosing specific pages.

The system should be tested both from inside and outside of college to ensure it is working properly.

The system should be able to be left operating without manual intervention for extended periods of time. Therefore, the code needs a high degree of stability. Possible checks include checking that the memory usage of the program does not steadily increase throughout the day.

The system should be stress tested to determine the maximum operational limit. Once this has been determined, a limit should be placed so that it cannot be exceeded (e.g. Maximum number of clients that can access archive before the server cannot seek on the hard disk fast enough).

4.1 Specific Tests

Test	Expected Result
Requesting a lecture from the archive.	The lecture is displayed on the client's screen. The lecture is the one that the client requested.
See which lectures have been archived.	The correct lectures have been archived. There should be no missing lectures unless some known system problem has interfered with it. If there is a system problem, it should be corrected.
Search the archive for specific lectures.	The search function should return accurate results.
Check all the aspects of the camera can be controlled.	The camera responds correctly to the controls.
Add controls to the lecture and test they restrict the access to that which they define.	Only those with authorisation should be able to access a lecture.
Check distribution of lectures is correct.	Multicast should be available on local machines. Unicast should be available on remote machines.
Student asks a question during the live lecture.	The question should arrive at the lecturer's screen and inform the lecturer.
Check quotas	Users should not be able to view more lectures than their quota allows. Exemptions should allow users to watch more than their quota would otherwise allow.
Lecturer's screen.	The lecturer's screen is accurately forwarded and screens from different lectures are not mixed up.
In-vision watermarking.	Displays the appropriate username if applicable.
Connection speed detection.	The connection speed should be detected correctly and an appropriate quality video transmitted.
Automatic Installation.	The software should be installed correctly and function as expected from the other tests.

4.2 Questionnaire

User satisfaction questionnaire

1. Did the program serve your requirements?
2. Did the program meet your expectations?
3. Did you have any problems accessing the program?
4. On a scale of 1 to 5 where 1 is very poor and 5 is excellent how would you rate the ease with which you found the lecture you were looking for?
5. How would you rate the user interface of the client application from an ease of use perspective?
6. How would you rate the user interface of the client application from a functionality perspective?
7. On a scale of 1 to 5, how would you rate the visual quality of the broadcast?
8. On a scale of 1 to 5, how would you rate the audio quality of the broadcast?
9. If you used the question facility, was it correctly sent to the lecturer?
10. On a scale of 1 to 5, how would you rate the efficiency of the camera control system?
11. If you have an appropriate suggestion to offer to improve the effectiveness of the remote lecture program please email it to Marc Mentior.

Chapter 5

Implementation

5.1 Server

5.1.1 MPEG4IP

Video Quality

The video resolution for lectures has been decided to be 352x288 at 25 frames per second with a bit rate of approximately 350 kilobits per second. There is an inherent relationship between $FrameRate \times FrameSize \times Quality \propto Bitrate$ in video codecs. Quality, in this context, refers to the visual quality perceived by a viewer for a given resolution. This includes things such as crispness, smoothness and pixel bit depth. To increase one value, while keeping some other aspect the same, it is necessary to change at least one of the remaining aspects.

The use of a higher frame rate and lower resolution may reduce latency in the video and increase the chance of a frame being successfully decoded.

The reason for the reduced latency is that less data needs to be used to represent a frame and will take less time to send. The increased chance of decoding is caused by the reduced number of packets. The likelihood of a packet being lost is similar in a large sample set when the network conditions are the same. Therefore, if 1 frame is split across 20 packets and one packet is lost, it may not be possible to decode the whole frame.

If we have 2 frames split over 20 packets the loss of 1 packet may not make the video unplayable, but it is likely this lost packet will still produce decoding errors which may be noticed as visual artefacts, as there is likely to be a dependence on previous or subsequent frames. Figure 5.1 on page 40 is an example of this.

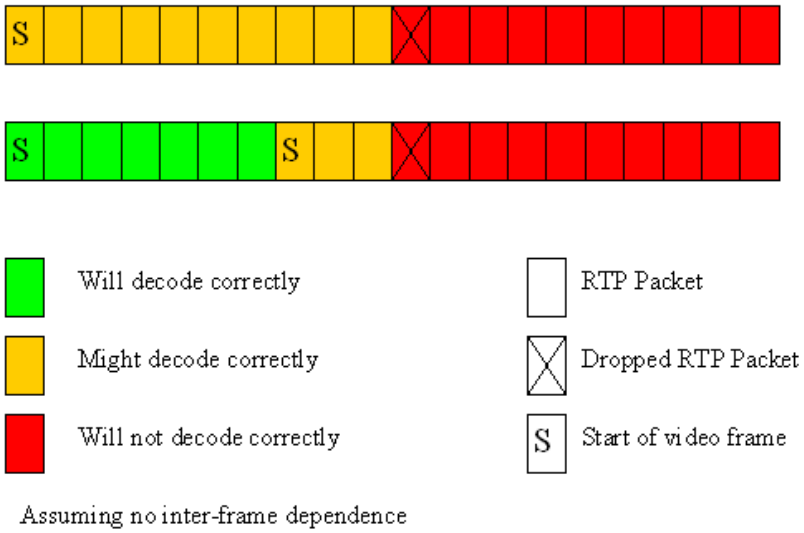


Figure 5.1: Packet Example

Watermarking

Watermarking is performed by loading in a watermark file which is in BGR (RGB with the B and R values swapped) format in the same dimensions as the video being captured. This RGB image is then converted into a YUV 4:2:0 planar format. Watermarking is then implemented by comparing the Y component with 0 (ie black). If it is not, it replaces the image at that point with the value from the overlay and replaces the appropriate U and V values as well. Black is the transparency value in the overlay image. The watermark image is a largely black image with a white Imperial College logo in the corner. See figure 5.2 on page 41.

Several config file options were added to support watermarking. These include:

`videoWatermark` - which, when set to 1, enables watermarking.

`videoWatermarkFile` - specifies the path to the watermark file. This file is stored as a RAW 24-bit BGR image.

`videoWatermarkWidth` - the width of the watermark. Currently, this is used only to load in the file, but it could be used in future to allow the watermark to be resized.

`videoWatermarkHeight` - the height of the watermark. This is also only to load in the file.

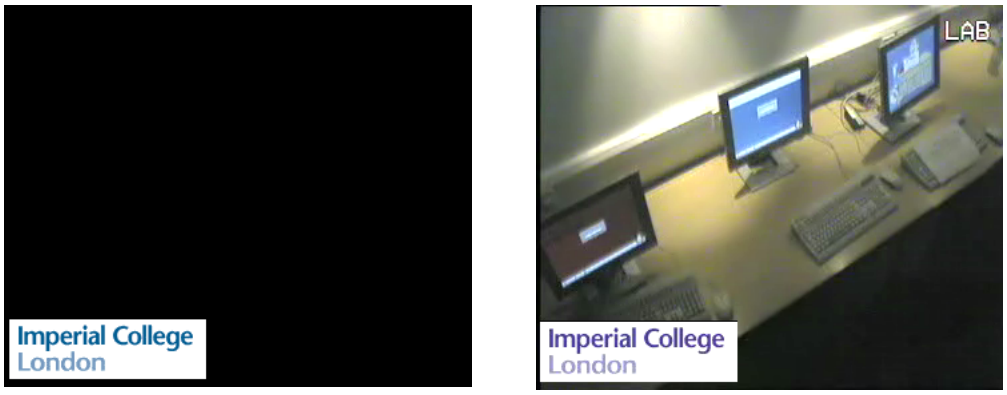


Figure 5.2: Watermark overlay and the result of an overlay

File Switching

Adjusted MPEG4IP to support file name patterns and added an extra duration parameter (`totalDuration`). The behaviour was then changed so that at duration intervals the file was switched to a new file specified by the file name pattern. After the `totalDuration` time had passed the program would terminate.

This allowed the automatic recording of lectures to be started at the beginning of the day and remove any gaps caused by terminating and restarting the program every hour.

`totalDuration` - The length of time mp4live should stay loaded.

`totalDurationUnits` - The unit of time for `totalDuration`, given in seconds.

`duration` - Recording time of an individual file.

`durationUnits` - The unit of time for `duration`, given in seconds.

`recordMp4FileSpec` - The file name specification for naming a new file at the switch over point. Parameters are described in table 5.1 on page 42.

`room` - This specifies in which room the session is recording. This is used in the file name specification and the updating of the database.

%Y	Current year
%M	Current month
%D	Current day
%h	Current hour
%m	Current minute
%s	Current second
%%	The % sign
%R	The room (see the `room` parameter)

Table 5.1: List of replacements

Database Connectivity

The database connection is used to inform the website when a new file has been recorded. Every lecture that is recorded is added to the database along with the file name it was given. This allows the web interface to correctly predict the RTSP URL for the lecture.

`useDB` - Specifies whether or not to use the database. A value of 1 indicates the database should be used.

`dbHost` - The server on which the database is located.

`dbPort` - The port on which to connect to the database. The default is 5432.

`dbName` - The name of the database to use.

`dbUser` - The username with which to connect to the database.

`dbPass` - The password with which to connect to the database.

`selectLectureQry` - This is the query used to determine which lecture we are currently recording. This allows the query to be updated to take account of new information without requiring the program to be recompiled. It is parsed for the same parameters as those in `recordMp4FileSpec`. For the moment this should be left as

```
SELECT find_lecture_room(%R)
```

`updateLectureFilesQry` - This parameter is currently not used, due to its dependence on the results returned from the select query.

5.1.2 Darwin Streaming Server

Darwin Streaming Server allows the use of modules to perform functions. To perform a specific function a module must register for the *role* it wishes to perform. There were several complications to adding the two parts of this module (Authentication and Authorisation). The first problem was that Darwin specifically only supported the use of 1 authentication/authorisation module. This meant the old module definitely had to go. The second problem was the structure of the first module made it very hard to implement in a similar manner.

Instead, it was decided that rewriting large sections of it and losing compatibility with the old `qtaccess` file system would be preferable.

PAM Authentication Module

A module was created to utilise PAM for authentication - thus allowing kerberos or other authentication types to be easily used. The use of STunnel provided a sensible requirement, as transmitting passwords in plain-text over the internet was not a very secure way of doing it. Due to the lack of support in both Darwin Streaming Server and JMF, it seemed sensible to use STunnel's ability to act as a proxy rather than implement full SSL support on both sides.

The initial plan for implementation proved infeasible due to Darwin Streaming Server's choice of allowing, first, only one Authentication Module. This meant that the old one which checked the password file needed to be removed. The second problem was caused by the way Apple had chosen to implement their authentication. Instead of providing the password to the authentication module to be checked, the authentication module returned the encrypted version of password which had been obtained from the file. Eventually, it was discovered that the plain-text password that the client had sent could be obtained and verified using PAM. Once verified by PAM, a dummy encrypted version was created and returned that the normal verification routine would match with the plain-text password that was sent. A full description of this can be found in Appendix E.

Database based file access configuration

The part of the module that deals with authorisation assumes that the user has already been authenticated. It looks up that users profile object. The user's profile object contains that groups the user is a member of. The user profile, machine and the filename are then sent using libpq to a stored

procedure. This then determines the lecture from the machine and filename and then determines the access level for the given user.

5.1.3 Camera Control Service

A camera control service was written that allowed an appropriately privileged user to remotely control and administer the camera. Authentication was again done with PAM and the same database tables were used for access control. An access level greater than 1 allows the user to control the camera. No further distinctions were made in this implementation.

The current implementation for the provided camera implements several features. These include variable speed panning and tilting, time limiting on the zoom and focus commands and access to the cameras menu system. The time limiting on zooming and focusing was due to the cameras implementation of performing the command for 2.2 seconds. This was circumvented by sending a stop command after a short delay to the camera. This delay is controllable from the client application.

5.1.4 Database Schemas

When designing the schemas for the database it was attempted to reduce the unnecessary duplication of data, while at the same time reducing the number of cross table joins required. PostgreSQL was chosen as the database system as this was the department's supported system. However, its support for SERIAL data types and functions (stored procedures) as well as its C library were also taken into consideration.

Producing a database independent version was considered however differences in SQL statements between different databases, combined with differences in functionality (such as lack of support for stored procedures), gave rise to the decision that abstracting to a sufficient level to overcome these differences would consume too much time.

Attributes of a User

These were not part of the database schema as these could be obtained from the /etc/passwd and /etc/group files. See table 5.2 on page 46 for a description of the attributes.

groups[]	groups a user is in
name	the users name

Table 5.2: User Attributes

Attributes of a Lecturer

See table 5.3 on page 47 for a description of the attributes.

Field	Type	Description
Lecturer	variable-length string(6)	Lecturers username, e.g. ih
Name	variable-length string(60)	The lecturers name, e.g. Ian Harries
Room	variable-length string(8)	The lecturers room, e.g. H360

Table 5.3: Lecturer Attributes

Attributes of a Lecture Course

See table 5.4 on page 47 for a description of the attributes.

Field	Type	Description
Code	variable-length string(6)	The course code, e.g. C419
Title	variable-length string(60)	The course title, e.g. Advanced Multimedia

Table 5.4: Lecture Course Attributes

Attributes of a Lecture

An individual lecture should have a unique identity. Hence, a unique number is assigned to every lecture. This number can be obtained from the room, date and time attributes as these are unique to a lecture.

See table 5.5 on page 47 for a description of the attributes.

Field	Type	Description
Lecture	SERIAL	A unique number for the lecture
Date	timestamp without time zone	The date and time the lecture will occur
Room	variable-length string(8)	The room within which the lecture occurs
Lecturer	variable-length string(8)	The username of the lecturer
Course	variable-length string(6)	The course code of this lecture

Table 5.5: Lecture Attributes

Access Control

The access control table was designed as a configurable and overridable system. It is based on a system known as a driver set. In this system, a set of weighted drivers are used to determine the appropriate rule. In this case, the access level that should be granted. The driver set chosen consists of the users group, the user, the lecturer, the lecture course and a specfic lecture. These were weighted as follows:

- group = 1
- user = 2
- course = 4
- lecturer = 8
- lecture = 16

Hence, a rule which is specific to the lecture will override any rules which do not implicitly mention that lecture. Typically, it would be expected that the lecturers set a default rule for themselves for a particular course. This allows them to override a particular lecture or user/groups settings if they wish. A value of * indicates that this field has not been set. The weight of a particular rule is simply the sum of all the set fields weights.

To select an access rule, all fields should be provided to a stored procedure which will then select the maximum weighted rule and return its access level. In the case of the user being in multiple groups, all the groups should be provided as a comma separated list.

In the case of two rules with the same weighting being found, for example, when a user is in two groups which have different access levels defined, then the result is undefined. Future implementations may wish to implement either a minimum or a maximum access level system, or give priority to the first group alphabetically.

The access level is an integer where 0 is equivalent to deny access. In this implementation, 1 represents access to view a lecture and 2 represents access to control a camera. Access level 3 represents that the user should be presented also with the camera menu control buttons, although there is no functional difference between levels 2 and 3 with the ability to control the camera (i.e. level 2 grants you full access to the camera, but not to the GUI).

See table 5.6 on page 49 for a description of the attributes.

Note: Lecture is represented as a character field so the value '' can be used*

Field	Type	Description
Group	variable-length string(15)	A group
User	variable-length string(8)	A username
Lecturer	variable-length string(8)	A lecturers username
LectureCourse	variable-length string(6)	A course code
Lecture	fixed-length string(11)	The unique identifier for this lecture
Weight	integer	The weight of this rule
AccessLevel	integer	The access level this rule defines

Table 5.6: Access Control Attributes

Lecture Resources

Each resource which a lecture references can be added to the Resources table. The value of the resource field can be HTML code representing a resource. For example it could be an HTML link:

```
<A HREF="http://www.doc.ic.ac.uk">Department Homepage</A>
```

A page is dynamically generated containing all the resources for a lecture. This is then displayed in the client.

See table 5.7 on page 49 for a description of the attributes.

Field	Type	Description
Lecture	integer	A lecture
Resource	text	HTML code representing the resource

Table 5.7: Lecture Resources

File Locations

These tables were designed to locate where a file referenced by a lecture was stored. See table 5.8 on page 49 for a description of the attributes.

Field	Type	Description
Lecture	integer	A lecture
File	text	A parameter made of ‘machine name : full file name’

Table 5.8: Lecture File Locations

Live Lectures

Each entry represents a different feed for the client application. MulticastCount allows audio and video to be combined into a single entry. The base port represents the lowest port numbered stream. 2 is added to the base port to obtain the next stream. Hence, if a feed had MulticastIp 239.251.1.1, MulticastCount was set to 2 and MulticastBasePort was 35000, a stream would be broadcast on 239.251.1.1:35000 and a second stream would be broadcast on 239.251.1.1:35002. These would have been grouped together to produce a feed.

See table 5.9 on page 50 for a description of the attributes.

Field	Type	Description
Lecture	integer	A lecture
SDP	text	The location of the SDP file
MulticastCount	integer	Number of streams to this feed
MulticastIP	inet	Multicast IP address of this feed
MulticastBasePort	integer	Base port for multicast feed
Type	variable-length string(20)	The type of feed, e.g. Camera

Table 5.9: Live Lecture Attributes

Server Parameters

This is used to represent values which although largely unchanging may be of use to applications. For example, it may contain a value which says 'rtspPort' for the room 'lab' is '55801'. Other uses include a value for the root of the streaming server 'movies' directory so that dynamic RTSP URLs can be created. It also includes things such as the question server for a given room.

See table 5.10 on page 50 for a description of the attributes.

Field	Type	Description
Room	variable-length string(8)	A room
Parameter	text	Some room specific parameter name
Value	text	The value for this room specific parameter

Table 5.10: Server Parameter Attributes

5.2 Administrative Client

The administrative client was split into two parts. The first part allows the lecturer to control access to a lecture and add additional resources. This takes the form of a set of webpages. The second part takes the form of an extension to the client applet. This allows the lecturer or a user with the appropriate access to control the camera.

5.2.1 Web Page

Access Control

To add rules the lecturer is presented with a webpage based wizard. The lecturer may select whether he wishes this rule to apply to a specific group, or '*'. The lecturer then may choose a specific user or '*'. The process continues with the lecture courses presented by the lecturer and '*'. Finally individual lectures and '*'. The '*' option as described in the lecture access control option represents that any value is valid for this rule.

Several items should be noted:

- The lecturer is not presented with an option to select his/her name or '*' to prevent interfere with the default rules.
- The list of lectures presented will only display lectures relevant to the course selected (or all lectures if no course has been selected).
- Lectures are displayed in 'Room Date Time' format.

Finally the lecturer is presented with an option to pick an access level.

Once the lecturer clicks add this rule will be added to the database.

A list of rules, along with an edit and delete button, is available should the user wish to change or delete a rule.

Overview of Extra Resources

Extra resources are implemented using a similar system to Access Control. Three options are available; add, edit and delete. Add presents the lecturer with a selection of lectures in 'Room Date Time' format and a text box for the user to type in HTML. Once add is pressed, this is added to the database. Edit and Delete present a list of lectures initially. Once a lecture has been selected, a list of resources for that lecture is presented. The lecturers may then delete or edit these as they wish.

A link to the resource display page for a lecture is also provided so that a lecturer may check the way the resources will be displayed.

5.2.2 Java Applets

Camera Control

Camera control is performed by a Java applet, which presents the user with a *live* feed from the video along with controls to zoom, pan and tilt the camera. When the user clicks one of these buttons the applet sends a command to the remote camera control service which is connected to the camera. The camera control application has a defined interface which means that a different application can be substituted to enable other cameras to be supported.

Communication is performed using SOAP messages. These allowed both authentication of the user and the ability to return the success or failure of the command (and the reasons for failure).

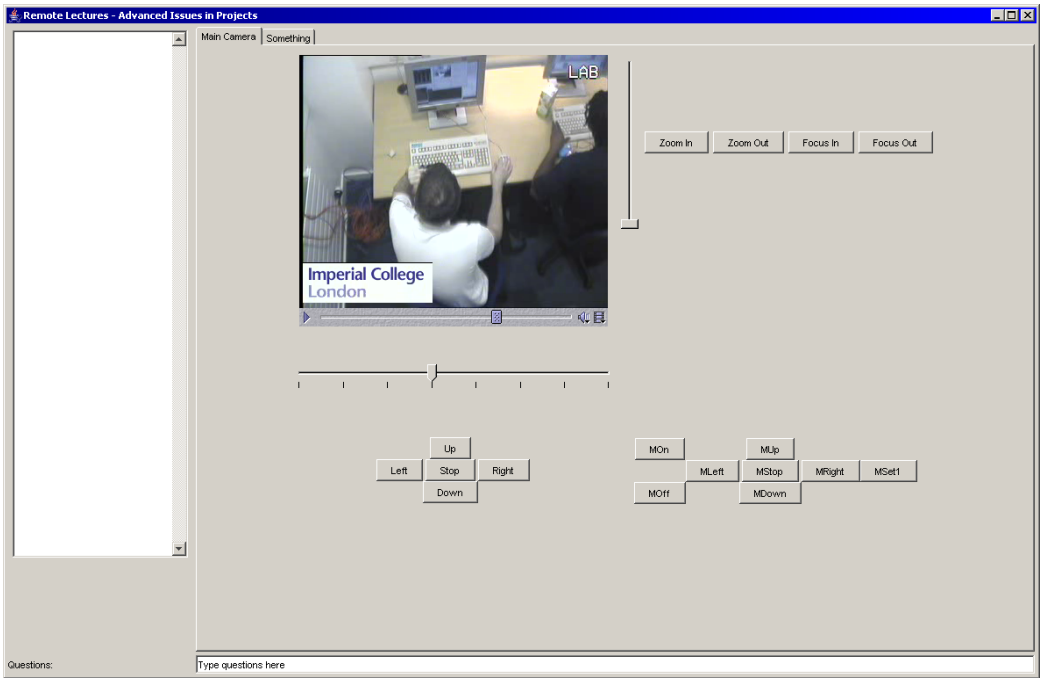


Figure 5.3: Example of the Camera Control Interface

The interface is defined in Appendix B on page 81.

5.2.3 Desktop Java Application

Screen capture

A JMF plugin was written to allow Screen Capture. This is started by visiting a webpage with the Java applet. This then captures the screen, compresses it and sends it over RTP. A pregenerated SDP file for this is then added to the Live Lectures table in the database. The applet compresses the screen captures and sends them over RTP to a multicast address which it obtains from the database. Upon the applet being closed, the line in the database is removed.

Question service

When the lecturer logs in, a SOAP service is started on the lecturer's machine. This then presents the lecturer with the option of allowing, denying or holding questions. These options cause questions to be either displayed immediately, refused or delayed until a time at which the lecturer clicks accept.

When a question arrives and the service is in Allow mode, a message box will appear asking if the lecturer wishes to display the question on the screen. If the lecturer clicks Yes, the question is displayed. If the lecturer clicks No the question is held until the next question arrives. If a question arrives whilst a question box is already being displayed another one is displayed.

5.3 User Client

5.3.1 Structure

Java Applet

A Java applet was chosen as the interface to the system due to its easy configurability through a webpage using applet parameters. It was also chosen for its cross platform support and ability to integrate with the Java Media Framework (JMF).

Despite the ability to download the applet automatically, some client-side installation is required. Firstly of Java, but also of JMF (which was modified during the course of this project) and several plugins which were written. It also needs some libraries for SOAP, XML and JAX-RPC. Details of these can be found in Appendix F.

These all need to be installed in the client's `CLASSPATH` to work correctly.

The applet needed to be signed as it tries to access a remote machine. Details on how the applet was signed can be found in Appendix D on page 87. Using an applet also meant the client could be upgraded simply by changing the applet on the webserver and the clients would automatically redownload it.

Both Java and the JMF need to be installed on the client's machine as well as the custom plugins that have been written for it to work.

JMF

5.3.2 JMF additions

To get the video displayed in a Java applet that could be used on any system some Plugins needed to be written for JMF. These plugins are described in the following section.

Changes to JMF's RTSP implementation

A minor change, which had major repercussions, was required to make Sun's implementation of RTSP work correctly. This change was embedded deep within the implementation and required the recompilation of the entire JMF library. This change took a long time to find due to the multi-threaded nature of the library and the fact the exception that was generated was being caught and not acted upon. Although the change was minor, it proved rather difficult to get JMF to recompile. Recompilation appeared to be necessary as the change of a field from an int to a double appeared not to be binary

compatible. As it was used in other classes, replacing just the individual class could not be done. The JMF source comes in two parts; the source code which is distributed under Sun's Community Source Licensing Program and the binary only parts for which the source is not available. Both these packages need to be installed before attempting compilation. Also, under Windows, some classes from Microsofts Java implementation are required as well as some source which did not appear to be included in either the source or binary distribution. This was obtained within the JAR of the original JMF. Visual C++ also needed to be installed to compile the C libraries under Windows and needed to be in the path. However, if this was added to the path initially, certain things would not compile initially and therefore needed to be added after they had compiled and an error had been generated saying VC++ could not be found. Once all these criteria had been satisfied, the JMF source code would compile.

Some other changes to the RTSP implementation were considered. The passing of the SDP information to the RTP Demultiplexers would have been useful, as this contained the video size information and would not have required the video size to be changed.

Support for RTP via RTSP encapsulation would also have been useful, as this would have reduced the effect of firewalls on clients wishing to view Remote Lectures.

MP4 File Format Demultiplexer

This plugin was based on Sun's QuickTime demultiplexer as the MP4 file format is based on QuickTime's format. Several changes needed to be made to the plugin. Changes were made to support the 64-bit variants of fields defined by the MP4 file format which were necessary due to their use in the MP4 files being produced. Some new atom types needed to be added due to important information being stored within them (such as the video format description). Changes were needed to fix problems with the expected layout of the files. Certain atoms which initially appeared to be supported (given their presence as functions) turned out not to have been implemented at all and just were skipped.

Several problems were encountered due to the fact that Java does not have unsigned types, but the MP4 specification specifically uses these in a large number of places. This was mostly overcome by the use of casting to a wider type and bitwise anding them with the maximum value of the previous type. This was not overcome in the case of longs and consequently only 63-bits are effective. This should be unlikely to cause a problem as this represents numbers in the range of 0 to 9,223,372,036,854,775,808. As a comparison

the number of seconds since 1970 is about 1,087,306,000. This leaves 9 extra digits, even if we offer millisecond resolution that is still 6 more digits to fill up. It is unlikely that any presentation will have use for this many bits let alone the extra one bit.

The change that needed to be made to allow the video format description to be parsed also uncovered a problem, as this information was not being sent to the decoder. This resulted in a change being made so that in the case of the first packet being sent to the decoder, a header, including this information was sent along with the rest of the packet.

The original plan had been to write an MP4 file parser from scratch using the object oriented nature of the file format, but this was not finished as this turned out to be a much larger undertaking than had been envisaged. So, instead, the decision was taken to keep the modified QuickTime demultiplexer.

There are still atom types which are not supported; these either do not occur or do not provide information that is useful to this project.

Changes also needed to be made to the way audio samples were sent to the audio decoder. The original implementation only catered for a few audio formats which used constant sample sizes. This needed to be changed to look up the appropriate size for each sample. It was decided that the best approach would be to scrap the old audio extraction part and rewrite it based on the video extraction part which had been determined to be working properly.

MPEG-4 Decoder

Several adjustments to JFFmpeg to support the decoding of MPEG-4 Video were made. These originally only included just the addition of registering the types, however later required changes to the workings of the code to support the changes in the resolution caused by the Depacketiser not having information about the resolution of the video before it had started to receive the video.

JFFmpeg initially featured code to support the reception of H.263 over RTP. This was removed due to its incompatibility with the current version of FFmpeg and a separate Java based depacketiser was written.

The JFFmpeg code also suffered severe problems decoding the bitstreams produced by the xvid codec, but, by using ffdshow (a Direct Show wrapper for FFmpeg), it was eventually determined that a bug in the current implementation of FFmpeg was causing this problem. An upgrade to the latest version a few weeks later solved this problem.

Some visual artefacts still occur in the initial frames, but this usually lasts

less than a second and was not deemed to be a major problem. It is believe that this is caused by the decoder still not being initialised correctly when the first frame is received.

MPEG-4 Video RTP Depacketiser

An MPEG-4 video RTP depacketiser was written to support the receiving of MPEG-4 video over the internet. Initially, the plan had been to use the MPEG-1/2 RTP depacketiser as a base for this, but it soon became apparent that the implementation was thoroughly different and resulted in it being almost completely rewritten.

The code will detect the type of packet it is currently dealing with and, if it is of an appropriate type, attempt to decode the data contained within. From this information, it was possible to correctly set the video resolution. This, however, would have been substantially easier if the RTSP implementation in JMF passed the `fmtp` attribute from the SDP information received at the initiation of the connection.

AAC Decoder

A JMF plugin was written and worked as a JNI wrapper for libfaad to decode the audio. Several weeks were spent attempting to get this working as it produced errors which were either meaningless or wrong.

A step by step comparison of a working decoder utilising an older version of the same library revealed no visible differences. A lack of comments in the libfaad source made understanding what each function was supposed to do difficult.

It was eventually found, after an exhaustive stepping through the libfaad code, that having detected the sampling rate of the audio correctly, it would double it. The reasons for this behaviour are still unclear although this is why the audio did not work. Sampling at a higher rate (32,000Hz instead of 16,000Hz) produced the correct audio.

Problems with the audio were not helped by the microphone attached to my machine which began to function incorrectly. This was eventually tracked down to a slightly malfunctioning microphone input on the case.

MPEG-4 Audio RTP Depacketiser

Using the information given in RFC 3640 a depacketiser for the MPEG-4 generic profile was written. It was, however, designed only with audio in mind and, as a result, does not support outputting of any other kind of

media. The depacketisers output was fed into the AAC decoder plugin, which in turn produced raw audio which could be sent to the audio render.

The depacketiser does not implement all functionality of the RFC. This is mostly due to time constraints and the lack of a need for these features. For example, it does not support the use of Decoding Time Stamps or Composition Time Stamps and instead bases its timing on the timestamp of the RTP message combined with the AU index deltas and sample rate.

Whilst producing this, several pieces of information were needed from the SDP. As JMFs implementation does not support the passing of this information lots of pieces had to be hard coded. This hard coding does not present difficulties though, as both Darwin Streaming Server and MP4Live are consistent in their transmission of audio data.

Chapter 6

Evaluation

6.1 Targets of Specification

Outlined below is a comparison between the specification and the actual achievements. A tick means I view the functionality as successfully working, a grey tick means that work on the function was performed, but it was either not finished or not successfully integrated with the rest of the system. A cross represents a function that was not implemented due to time constraints. A more detailed description of the workings is given in the implementation chapter.

6.1.1 Minimum Specification

Automatic archiving of lectures. ✓ This was implemented in a way that allows all lectures to be automatically archived. A lecture should be defined before it occurs so that the appropriate file can be associated with it. Java code to enable the playing back of these files was generated. A database and changes to MP4live were made to automatically update the database.

A comparison with the planned implementation shows that it was largely accurate, with the only difference being that the quality of audio suggested was much lower than was required. This was mostly to do with issues relating to libfaad and its decoding of low sample rate audio.

Searchable archive for later playback. ✓ The archive was implemented to allow several search criteria. Lectures need to be added to the database beforehand so that they may be properly associated with any

files automatically. Several tools allow changes to these tables so that updates can be performed.

The plan shows that, although small differences such as the primary key in the database was changed, this section was also followed well.

Camera control. ✓ The camera can be controlled by a user with appropriate access, using a Java Applet which can be loaded from the webpage. A camera control service is run on a machine defined in a database table. This table can be updated if the camera control service is moved. Authentication of users is provided by an interface to PAM, and authorisation is provided through a connection to the database.

This section again stayed largely within my plan.

Distribution (Multicast/Unicast) ✓ Lectures are distributed to individual clients upon request using unicast. Live lectures are distributed via multicast within the department. The website allows users to select whether they are inside the department or not. The users are then presented with the appropriate settings for unicast or multicast connections.

This section followed with my plan. However, it did require more software to be installed on the client's machine than I had envisaged. Automatic detection of whether a user was in the department was not performed as it may be possible that they are within the department but unable to receive the multicast feed due to their router/switch not supporting it.

Administrative controls on content. ✓ A configurable Access Control system which is a database table is used. The use of stored procedures (Postgresql Functions) allows for client applications to use these access control features easily. There is currently no RTSP authentication in JMF, so although Darwin Streaming Server was adjusted to support it, it is not used. However, QuickTime does support authentication and can be used to connect to the resource instead. In the authentication section there is no ability to specify that anyone may access the lectures, hence the `all` group in the specification is not available. However, the presence of a `qtaccess` file in the directory determines whether authentication is required or not, so it would be possible to move certain lectures to a directory with no access restrictions hence allowing anyone to watch them. The `alldoc` option is, however, provided by granting an access without specifying a user or group, as users are authenticated before they are authorised.

Webbased/Java Applet access site. ✓ A website allows the user to search the archive and play videos. It also allows lecturers administrative control over their lectures. An admin page has also been designed to allow a `super-user` to control other aspects of the system.

6.1.2 Extensions

Live broadcasting and retransmission at specific times. ✓ Lectures are currently broadcast using multicast as they are recorded. They are also relayed using Darwin Streaming server and unicast. There is no rebroadcasting facility currently available, although this is not to say that a lecturer could not direct the students to a video in the archive.

Interactivity ✓ The students can type questions into their client. These are then sent as XML based SOAP messages to the SOAP service running on the lecturer's machine. This then either displays the message or suppresses it until the end of the lecture. It is then up to the lecturer to decide how best to deal with the question.

Remote Lecture Quotas / Exemptions ✗ Remote lecture quotas and exemptions were not implemented. It would also appear infeasible given the way in which applications like QuickTime produce connection requests. Several authentication and authorisation requests are produced for every connection. It might, however, be possible to implement this as a Darwin Streaming Server module if the quotas were not placed on the amount of times an individual lecture was watched but the number of different lectures. Alternatively, some kind of time or connection based system could be used to decide if a user had decided to watch the same lecture again or if his client was just performing some other command.

Lecturers Screen forwarding ✓ Lecturers' screens are currently not archived. However, in the case of slides, a lecturer could place these on the web in the traditional way. An application capable of capturing and encoding the screen was made but there was not enough time to implement an MP4 File Multiplexer. For this reason the Screen Forwarding could not be saved in a format that could be streamed again using the current streaming system. However, clients watching live lectures could view this through the generation of an SDP file. This SDP file was then placed in the streaming servers "movies" directory, in much the same way as for the video. A row would then be added to the live lectures table in the database.

In-vision watermarking ✓ Watermarking was implemented in MP4Live. This occurs before the image is compressed so it appears both on local multicast and remote unicast material as well as recorded material. Watermarking currently relies on a raw BGR format image, with a pre-

defined size. This could be extended to support PNG images and to allow resizing of the overlay image to the same size as the video.

Motion Tracking ✓ Although motion estimation and therefore detection were implemented, it proved infeasible to accurately track the lecturer as the motion estimation would produce some false matches due to the use of non-optimal heuristics to save on time whilst looking for motion. These heuristics needed to be used to keep the processing speed at a real time rate. Other problems, such as noise in the video, introduced the false impression of motion. The camera also seemed to suffer from a slight swaying effect, similar to that of a mirage. There appeared to be no direct cause of this and large periods of time could occur without this effect appearing. Although the use of several different techniques was experimented with, none proved able to give reliable enough results.

Intelligent Connection Speed Detection ✗ This was not implemented, but my research and has led me to believe that, with several changes to the mp4live code, a second file writer and RTP sink could be created to generate a different file and RTP output with different quality settings. These need not be at the same resolution as the original, but extensive changes would need to be made to retrieve the signal and process several different resolution images for the purpose of encoding. The problems of deciding which quality would be most appropriate for a given user and automatic switching between version was not addressed.

Largely automatic installation and setup of new lecture theatres ✗ Work on this was not completed. A list of packages which needed to be installed for functionality was compiled, but required manual intervention to install properly. The extra modules for the kernel which were required could be compiled in a way such as to allow machines other than the original machine to be used.

6.2 Performance

This section represents the evaluation of the software and tries to quantify its performance so that comparisons with between similar solutions may be drawn.

6.2.1 Latency

How much of a delay is there between the lecturer saying something and the student seeing it?

This varied because of several factors. This included network congestion, client side buffering. The buffering caused by Darwin streaming servers relay and the time taken to encode the video in mp4live all cause an delay in reception of the video.

During test from a home broadband connection the delay would vary between about 1 second and 15 seconds. The usual delay was around 8 seconds.

6.2.2 Resource Usage

How much CPU usage does the video encoding take?

On a Pentium IV 2.8GHz approximately 38% CPU usage occurs when encoding video, varying between about 35% and 41%. As a reference when idle the machine maintained a CPU usage level between 0 and 1%.

How much CPU usage does the streaming of recorded video take?

Streaming a prerecorded video to 1 client produced no noticeable difference in CPU usage. On an otherwise idle system the CPU usage stayed between 0 and 1%.

With 5 clients viewing separate lectures CPU usage still did not consistently reach even 1%.

This reason for this is that most of the processing incurred by sending streaming video is incurred in the first few seconds as it needs to read and process the MP4 file. After this only disk accesses are needed and hence CPU processing is minimal.

How much IO/Memory does video encoding take?

MP4Live used about 27mb of memory when video recording was started. After 10 minutes this was still at 27mb. After an hour this was 27mb. After

a full day of being loaded this was 27mb. These results seem to indicate a lack of any memory leaks in this program. Memory usage did increase about 40 minutes into each hour, however this was only 1mb. The increase is most probably caused by mp4live requiring extra memory to store the header information that is has not encoded into the MP4 file.

How much IO/Memory does streaming take?

Darwin Streaming Server apparently only took 4mb of memory when streaming 1 video. I find this very hard to believe due to the code size. Loading up a second stream increased this to 6mb. Opening a further 3 video windows increased the usage to 9mb. Giving an approximate value of 1.3mb per video stream. Notably it took Darwin Streaming Server rather longer to return its memory usage down after these streams had been quit.

6.2.3 Motion Tracking

How well does it work in normal use?

Unfortunately motion tracking does not work very well. It proved easy to confuse even when nothing appeared to be moving.

6.2.4 Stress and Load

How many concurrent clients can watch a live broadcast?

Based on my calculations in the previous section a very large number should be able to successfully watch a live broadcast. However, the network bandwidth to the machine may produce a limiting effect before that point. Over 250 clients should be able to connect before bandwidth becomes a problem. However, this assumes perfect network performance on both sides of the connection. A more real world value would probably be about 150-200. However, Darwin streaming server has the ability to act as a relay for another Darwin streaming server. Using this, it should be possible to distribute the load allowing substantially more clients to connect.

How many concurrent clients can watch different recorded lectures?

15 clients successfully watched different lectures at the same time. Larger tests were not accomplished due to time restrictions.

How many concurrent clients can watch the same lecture?

15 clients successfully watch the same lecture at the same time. These were all started at similar times so caching of the files may have occurred. More clients could not be tested due to time restrictions.

6.3 Testing

Requesting a lecture from the archive. This test succeeded, with the correct lecture being sent to the user.

See which lectures have been archived. The correct lectures were archived, all problems involving missing lectures were resolved.

Search the archive for specific lectures. The search function returns accurate results. Combinations search criteria could be added to improve the lookup process.

Check all the aspects of the camera can be controlled. The camera responds correctly to the controls. A delay of up to 15 seconds between the command being issued and visual confirmation of the command being performed may be noticed, depending on network conditions and settings.

Add controls to the lecture and test they restrict access as defined. When authorisation is enabled only authorised users allowed in the database rules are granted access. Database rules may initially appear confusing to lectures and may result in their creating incorrect rules which allow access to resources.

Check distribution of lectures is correct. Multicast is available on most DoC machines. Unicast is available on both local and remote machines.

Student asks a question during the live lecture. The question should arrive at the lecturer's screen and, depending on the lecturer's settings will inform the lecturer. The service may be unable to connect and will produce an error on the clients side.

Check quotas This feature was not implemented.

Lecturer's screen. The lecturer's screen is accurately forwarded and screens from different lectures are not mixed up.

In-vision watermarking. Usernames were not displayed in the watermark but the Imperial College logo appears.

Connection speed detection. This was not implemented.

Automatic Installation. This was not implemented. Manual installation is required.

6.4 Improvements of existing system

The following is a list of changes to code (such as completely rewriting them) that I would like to apply. However, due to time constraints, I was unable to do these.

- A rewrite of the MP4 demultiplexer to an object-oriented fashion. This would make extending it and debugging substantially easier.
- Rewrite the watermarking code to use a common image format instead of a raw BGR image.
- Instead of using the database functions directly, produce a wrapper class to automatically manage the memory of the result objects produced.
- Increase the ease with which the camera control service can be configured.
- Rewrite the GUI for the camera control interface. This is currently not very resizeable as there are a lot of buttons.
- Allow more information to be returned from the JFFmpeg and AAC decoder native C routines. Currently only a single value is returned specifying success or failure. Other useful values could be returned such as the length of the output packet.

Chapter 7

Conclusion

From my research and attempts to implement parts of the MPEG-4 standard I have found a wealth of options and features which not only provide very fine control over media but produce such a set of options that a complete implementation of them may take several years. There also appears to be no implementation available currently that supports all aspects of even one part of the specification. MPEG-4 extends and improves upon the work done with MPEG-1 and MPEG-2 increasing its scope such that it is capable of performing a wide variety of tasks.

During the course of my work I used C, C++ and Java as well as PHP and SQL. Switching between so many languages which share similarities in syntax as well as differences presented an interesting challenge. For example, after programming with Java, a reliance on the garbage collection for memory management makes it strange to explicitly free memory when returning to C. The syntax differences between creating objects in C++ and Java is also worthy of note as objects are automatically created in C++ by declaring them. In Java the objects must be `new`ed. Other differences such as the automatic freeing of local variables in C++ at the end of a function, thereby calling an objects destructor, allows for a more determinable memory usage pattern. Darwin Streaming Server uses C++ objects to automatically deallocate memory which seems to be a rather easy and clean way to clear up memory. An example of this is:

```
char *someString = (char *)calloc(20, sizeof(
   char));
OSCharArrayDeleter someStringDeleter(someString)
   ;
if (....) {
        return 0;
} else {
```

```
                return 1;
        }
```

In the above example, the string someString is automatically freed by the destructor of OSCharArrayDeleter. This is useful as it removes the requirement for the lines `free someString` to be present before both return statements. In a more complicated example, this can be quite an effective memory management tool. Of course, if the value of someString needs to be used outside of the function, after this function has terminated, this can be very problematic.

Overall, the project has laid thick foundations for any future projects in the area. Although the specific implementation ideas may wish to be reorganised, the majority of the code should be reusable in other similar projects. It is noted, though, that some of the code contains hard coding of variables that could not be parametrised in the project's current form. It also contains at least 1 memory leak in the decoding of video or audio. There are some synchronisation problems between the audio and video when being broadcast over the Internet. The source of these has yet to be determined and leaves the way open for further research.

Chapter 8

Possible Extensions

With further time and resources, I can see definite extensions to this project being created. Some of these are relatively minor and would really only be of use if there was an interest in marketing this product. However, any attempted marketing of this product would require the payment of patent royalties to the appropriate parties in respect to MPEG-4 related technology. A list of possible extensions is now given.

- Implement Support for Other Cameras.
- Reduce the latency experience when viewing the video.
- More error tolerance over RTP.
- Support for multiple compression rates.
- Add support for Secure RTSP to JMF.
- Support for RTSP Authentication in JMF.
- Add support for RTP over RTSP connection like QuickTime to get around firewalls.
- Add extra support for getting information from the SDP information.
- Better error handling for AAC and MPEG4.
- Better parsing of the timetable information.
- Support for resizing of a watermark.

- Adjusting JMF to pass Decoder Specific Data. For example in the case of RTSP based connections this could be the SDP information. This could then be parsed by the RTP Depacketiser, which in turn could produce specific initialisation data for the audio or video decoder.

Appendix A

User Manual

A.1 Server

A cron job automatically starts the server in the morning. This calls a script which takes a room name as a parameter. This will then start MP4Live with the room specific config file.

Currently config files reside in `/data/movies/mam00/` and follow the naming convension of `room_mp4live_rc`.

Several configuration variables may be of interest. Those which are room dependant are listed below.

`room -` This is the name of the room.

`sdpFile -` This is the name of the SDP file which will be generated.

`recordMp4FileSpec -` This is the file name specification for the recorded lecture files.

Darwin Streaming Server configuration can be found in: `/etc/streaming/streamingserver.xml`.

The presence of a `qtaccess` file in the root of the streaming server 'movies' directory indicates that authentication and authorisation using PAM should be enabled. Authentication is performed using PAM, which is setup through files contained in /etc/pam.d/. Authorisation is based off the access control database tables.

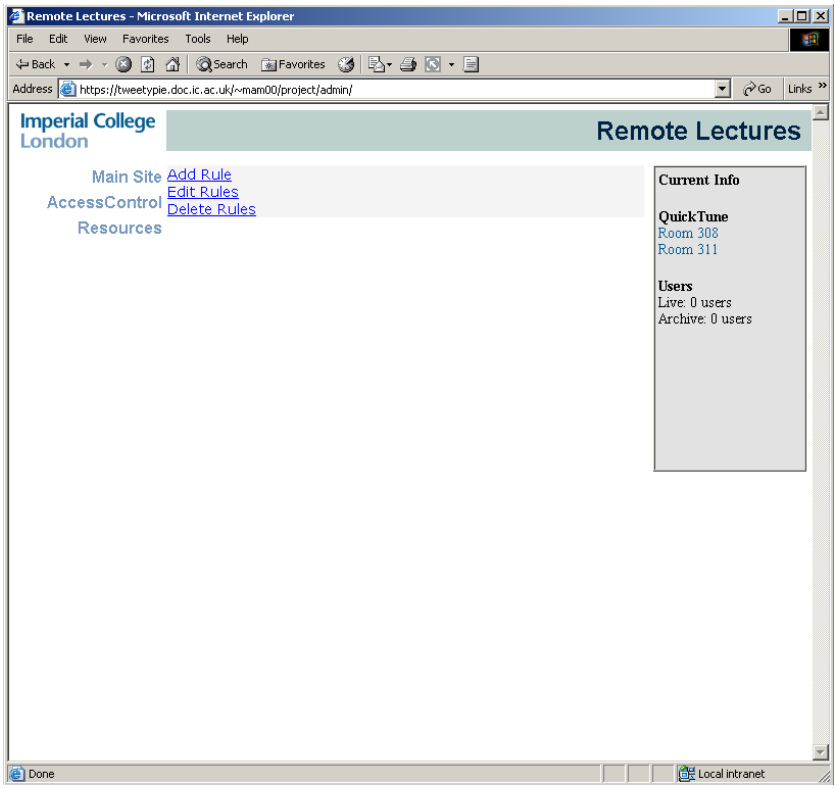
A.2 Administrative Client

A.2.1 Web Page

This can be accessed through the website address https://www.doc.ic.ac.uk/~mam00/project/admin/[1]

Access Control

To change the Access Controls for a lecture, select the Access Control link. You can then choose to Add, Edit or Delete existing Access Controls.
If you wish to Add an Access Control line, a step by step wizard guides you through the process.
If you wish to Edit or Delete an Access Control line, a page displaying all access control lines that are managed by you are displayed. You may then choose to delete these. If you wish to edit them a wizard, similar to that displayed when adding a line, will be displayed.



[1] www may need to be changed for `tweetypie` due to security restrictions.

Resources

If you wish to Add, Edit or Delete resources to a lecture click the Resources link.
The first step is to select the lecture you wish. A list of date time room combinations will be displayed and you should select the appropriate one.
You may then continue. A list of existing entries will be displayed.
If you wish to edit one of these, highlight it, and then click Edit.
If you wish to delete an entry, highlight it, and click Delete.
If you wish to add a new entry fill in the appropriate box and click Add.
Note: Reordering of entries is not supported.

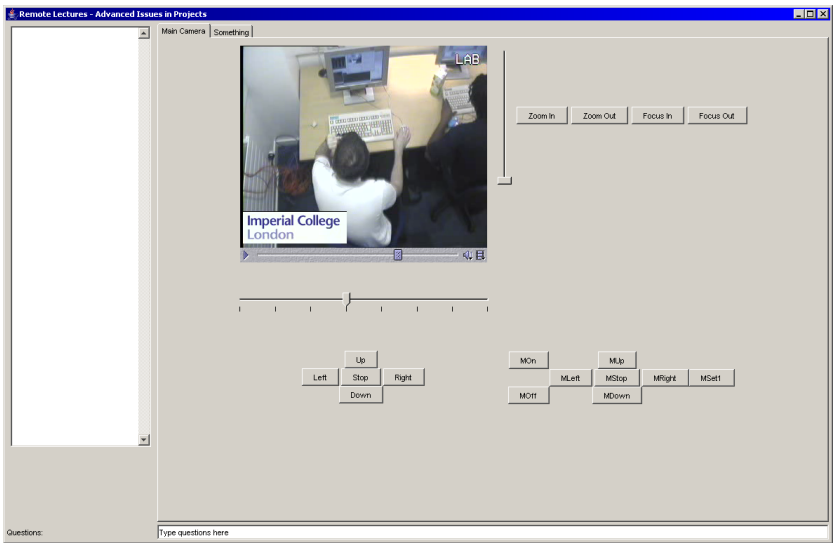
A.2.2 Java Applets

Camera Control

The camera control interface is obtained by going to http://www.doc.ic.ac.uk/~mam00/project/ and selecting the appropriate lecture. Provided the user has been granted access to control the camera, the appropriate controls should appear automatically.

If an access level of 2 or above has been granted, buttons such as Zoom, Left and Right, Up and Down and Focus will be displayed. A speed slider should be noted above the directional controls and a time slider should be noted beside the Zoom and Focus controls.

If an access level of 3 or above has been granted, an extra set of buttons, which allow control of the camera internal menu system, will be displayed.



A.2.3 Desktop Java Applications

Screen Capture

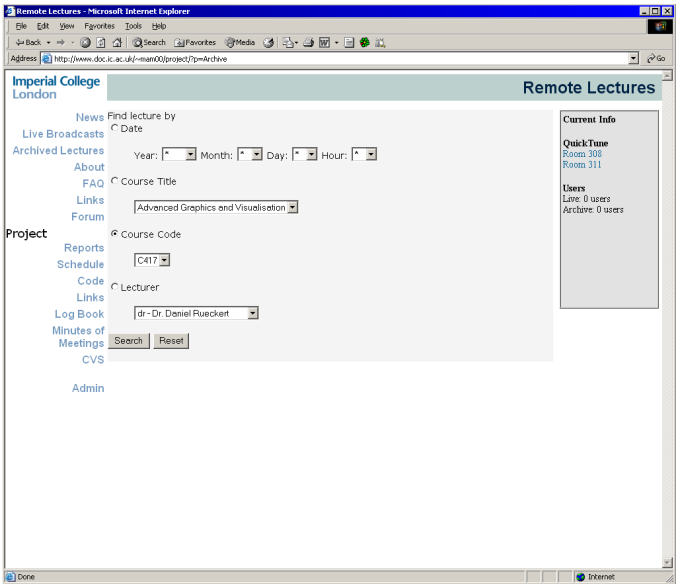
Once loaded, this should automatically broadcast a compressed version of the lecturer's screen to the clients. A line should be added to the appropriate database table so all clients connecting after this point will be able to view the screen.

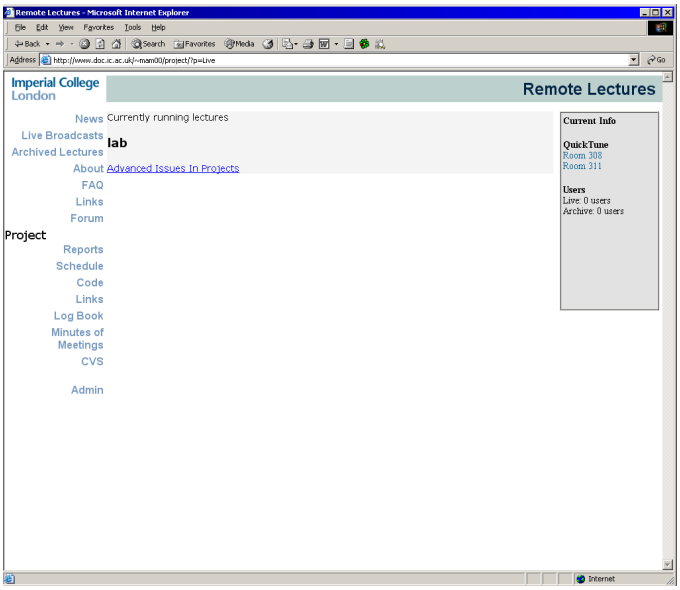
Question Server

Once loaded, this will display a set of options allowing the lecturer to select whether they wish to have questions displayed when they arrive, or at a later point in time. When a question arrives, it will display a popup on the lecturer's screen if they have chosen this option. The lecturer may then choose to view the question, or delay it until later. If they have chosen to view all questions at a later time the question will be added to a queue until the lecturer chooses to review the questions.

A.3 User Client

This should be loaded by visiting http://www.doc.ic.ac.uk/~mam00/project/ and selecting either to view a live broadcast or an archived broadcast. If the user chooses a live broadcast they will be presented with the option asking if they are currently within the department or not. If they select they are the client will be loaded using the multicast connection. If they select they are not the unicast system will be used.





A.4 Known Issues

A.4.1 Video

The first frame may not appear to be fully decoded when using the RTP transport. This is due to the RTSP implementation not providing a way of obtaining the `fmtp` attribute from the SDP information. The `fmtp` attribute contains the initial decoder parameters.

A.4.2 Audio

The audio may be corrupt or delayed in relation to the video. This is a known problem with the audio decoder. The solution has not been found.

A.4.3 Interactivity

Questions do not reach the lecturer. This may happen for several reason. These include the port that was chosen for the lecturer's question server not being allowed through the college firewall. Intermittent network conditions. The server being too busy to accept any more requests.

A.4.4 Camera Control

Sometimes, the camera will ignore all commands sent to it. There are 2 known reasons this may happen.

1. You have connected the camera to the machine after the camera and machine have been turned on. Turn the camera on and off.

2. You have not set the serial baud rate correctly. Try setting it to 115200 using the `setserial` command as `root`. See Appendix C for more information about this.

3. The cable has become disconnected or otherwise broken.

Appendix B

Camera Control Interface

The Camera Control Interface was implemented as a C++ SOAP server. It was created using the gSOAP compiler. The interface was based off the following command `int control(char *command, int *result)`. This allows the user to send a text command to the camera control service, and receive an integer result as a return value. The command and result values are dependant on specific implementation. However the following rules should be applied.

- A negative or 0 result indicates some sort of failure.
- A positive result indicates success.

The following commands should be implemented where possible.

`left speed` The camera turns left at the given speed. Speed is dependant on the camera. If no speed is given a default should be chosen. Speed values between 0 and 7 have been implemented in the client giving 8 different speed settings.

`right speed` The camera turns right at the given speed.

`up speed` The camera looks up at the given speed.

`down speed` The camera looks down at the given speed.

`zoomIn time` Zoom further in, stopping the zoom command after time has passed

`zoomOut time` Zoom further out, stopping the zoom command after time has passed

`focusIn time` Focus further in, stopping the focus command after time has passed

`focusOut time` Focus further out, stopping the focus command after time has passed

`stop` Stop moving/zooming the camera

`focus mode` Where mode is one of (auto|manual)

`menu command` Where command is one of (on|off|up|down|left|right|set1|set2)

Currently the service expects a username and password to be sent along with the request. Authentication of this is not currently performed and no access control is implemented.

Implementation should be performed by extending the `CameraControl` class and overriding `start`,`stop` and `performCommand` methods. You should also adjust `CameraControl.cpp`'s `getCameraControl` method, extending it so that it knows about the new type of camera.

Appendix C

RS-485

C.1 Kernel Changes[1]

Edit the file **drivers/pci/pci.ids**, adding the following (substitute for the PCI IDs of another card if necessary).

```
135a  Brain Boxes
        0261  Velocity RS422/485
```

Then run `drivers/pci/gen-devlist`. This will generate the appropriate .c & .h files. `lspci` should now display the Brain Boxes device by name (subject to kernel recompilation and reinstallation)

Next edit **include/linux/pci_ids.h**, adding the appropriate PCI IDs again as DEFINE statements this time.

```
#define PCI_VENDOR_ID_BRAINBOXES        0x135a
#define PCI_DEVICE_ID_VEL_1PORT_422_2   0x0261
```

Finally edit **drivers/serial/8250_pci.c**, adding the new device in the place shown. An explanation of the parameters is given below.

```
static struct pci_device_id serial_pci_tbl[] = {
               PCI_VENDOR_ID_BRAINBOXES, PCI_DEVICE_ID_VEL_1PORT_422_2,
                PCI_ANY_ID, PCI_ANY_ID, 0, 0,
                pbn_b2_1_921600 ,
        {       PCI_VENDOR_ID_V3, PCI_DEVICE_ID_V3_V960,
                PCI_SUBVENDOR_ID_CONNECT_TECH,
                PCI_SUBDEVICE_ID_CONNECT_TECH_BH8_232, 0, 0,
                pbn_b1_8_1382400 },
```

[1]These changes were designed for a 2.6.2 kernel. They should, however, be applicable to any 2.6.x Kernel and any 2.4.x Kernel

There are 7 parameters.

1. The PCI ID of the Vendor.
2. The PCI ID of the Device.
3. The Sub-Vendor ID. (Use PCI_ANY_ID if it is not known).
4. The Sub-Vendor Device ID. (Again, use PCI_ANY_ID if this is not known).
5. Always 0.
6. Always 0.
7. Device Settings. The description given in the 8250_pci.c is as follows. The makeup of these names are: `pbn_bn_bt_n_baud`, bn = PCI BAR number, bt = Index using PCI BARs, n = number of serial ports, baud = baud rate.

C.2 setserial

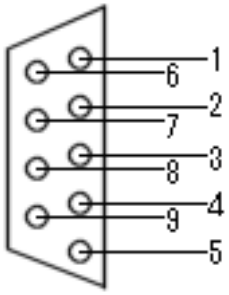
Although the default settings should function correctly, if they do not the `setserial` command can be used to configure the port. In the example given below, text in italic should be replaced with the value at the appropriate points.

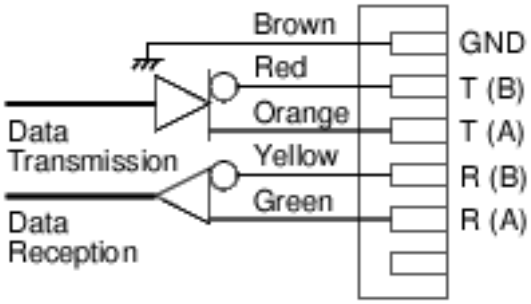
```
>lspci -v
...
02:0b.0 Serial controller: Brain Boxes: Unknown device 0261 (rev 01)
 (prog-if 00 [8250])
        Subsystem: Brain Boxes: Unknown device 0403
        Flags: medium devsel, IRQ  23
        Memory at feaff400 (32-bit, non-prefetchable) [size=128]
        I/O ports at d480 [size=128]
        I/O ports at  dfe0 [size=8]
        I/O ports at dfac [size=4]
...

>setserial  /dev/ttyS4 uart 16750 port  0xdfe0 irq  23
  baud_base 115200 spd_normal skip_test
```

C.3 Cable Specifications

RS485 Card **Pin Assignments** **Camera**





[61]

Pin	Purpose
1	TXD-
2	TXD+
3	RTS-
4	RTS+
5	GND-
6	RXD-
7	RXD+
8	CTS-
9	CTS+

Pin 1 ↔ Pin Red
Pin 2 ↔ Pin Orange
Pin 6 ↔ Pin Yellow
Pin 7 ↔ Pin Green
Pin 5 ↔ Pin Brown

Wire Colour	Purpose
Red	TXD-
Orange	TXD+
Yellow	RXD-
Green	RXD+
Brown	GND

Table C.1: RS-485 Pin assignments

An explanation of the abbreviations above is now given.

TXD Transmitted Data

RTS Request to Send (not used)

GND Ground

RXD Received Data

CTS Clear to Send (not used)

Appendix D

JAR Signing

D.1 Why Applets need to be signed

By default, applets are not granted access to system resources which are outside of the directory from which they were launched. In the case of applets launched from a webbrowser, it is not granted access to the local system and may only communicate with the webserver on which it is hosted. A signed applet can access local system resources. This is, however, limited by the local system's security policy.

A signed applet merely guarantees that the code which was signed for has not been altered since it was generated. It does not guarantee the code is not malicious.

D.2 How to sign applets

This guide tells how to self sign an applet. The process of self signing an applet allows a user to choose whether they trust you and to accept your certificate. This is needed to gain access to features such as network communication to hosts other than your own and the webserver. This is based off the webpage created by Simon Tokumine. [62]

1. Generating a private key in a keystore. Use a command similar to this
   `keytool -genkey -keystore myKeyStore -alias me`
   Fill out the requested information.

2. Next, generate a self signed certificate. Use a command similar to this
   `keytool -selfcert -keystore myKeyStore -alias me`

3. Finally sign the JAR file using the jarsigner program using a command like this
   `jarsigner -keystore myKeyStore jarfile.jar me`
   You will need to make sure the paths for myKeyStore are correct and the name of the jar file matches yours.

Appendix E

Darwin PAM Access Module

The initial plan for implementation of the Darwin PAM Authentication and Authorisation module proved infeasible due to Darwin Streaming Server's implementation.

The first reason is that Darwin allows only one Authentication Module to be loaded. This is explicitly coded in a number of places. This is presumably the reason the documentation for writing an Access Module is almost non-existent in comparison to the 200 pages devoted to other modules. The result of this limitation was that the old module was removed, as it would be impossible to convert the several hundred users' accounts within the departments to Darwin's password file format. It would also be impossible to generate the digest passwords which were made up of the MD5 of "user:password:realm", where realm was a Darwin specific authentication realm.

Instead it was decided not to support digest authentication as there would be no way to support this using PAM. This is because PAM requires both the plaintext username and password. These are then passed to the authentication module specified in the PAM configuration file for Darwin streaming server. PAM then verifies them using an arbitrary authentication method. In this case it was Kerberos. PAM would then return success or failure for the authentication.

The second problem with Darwin was that, in basic authentication mode, the authentication module was not expected to return whether the password was correct or not. Rather, it was expected to return an object containing the encrypted version of the users password. This encrypted version was then checked by the main server against the version of the plaintext password which it encrypted there. It was discovered that the RTSP parameters that were passed to the Authentication module contained the plaintext password was contained. Using this, plaintext password authentication with PAM could proceed. This still left the problem of returning an encrypted version of

the password. It was discovered that the encryption routine for basic Authentication was the library function `char *crypt(const char *key, const char *salt)`. A look at the man page for this function revealed that it required two parameters. These parameters were a key and a salt. The function then encrypted a predefined string (usually a string consisting of all '0's). The key was the users password and the salt was a random two-character string from the set [a-zA-Z0-9./]. The value returned was the encrypted password Darwin required, prefixed with the two character salt. The inclusion of the salt is the important factor. The salt means that only with the correct salt and the correct password could the correctly encrypted value be produced. ' The function in Darwin to compare the users passwords was based around the following:

```
if (strcmp(cryptedPassword, crypt(plainTextPassword,
   cryptedPassword)) == 0) {
  // grant access
}
```

Because the first two characters of the encrypted password are the salt, it becomes possible to trick Darwin into authenticating a user by deliberately encrypting a password with any given salt and returning this as the user's encrypted password.

Appendix F

Java Applet Libraries

The Java applet required the use of several libraries. These are listed below along with their function.

xerces - This is for XML parsing.

JAXB - Java Architecture for XML Binding.

JAXP - Java API for XML Processing.

JAXR - Java API for XML Registries.

JAX-RPC - Java API for XML-based RPC.

SAAJ - SOAP with Attachments API for Java.

JMF (modified) - Java Media Framework for decoding and playing video.

jffmpeg (modified for MPEG-4 video) - FFmpeg wrapper.

AACDec (created to decode AAC audio) - libfaad decoding wrapper.

MP4 - created to depacketise MPEG-4 video and view .mp4 files.

The XML and SOAP based libraries are used for communication with the camera and question servers.

Bibliography

[1] Ian Harries homepage, useful information about several aspects of the project.
http://www.doc.ic.ac.uk/~ih/.

[2] Open University main site.
http://www.open.ac.uk.

[3] International Centre for Distance Learning at the Open University.
http://www-icdl.open.ac.uk/.

[4] University of Genevas remote lecture system.
http://www.unige.ch/e-cours/e-cours_eng.pdf.

[5] Real Video and Audio.
http://www.real.com/.

[6] Remote lectures over Singapore ONE.
http://www.cdtl.nus.edu.sg/link/jul1998/tech1.htm.

[7] Mircosoft's Netmeeting.
http://www.microsoft.com/windows/netmeeting/.

[8] Wikipedia - RGB color model.
http://en.wikipedia.org/wiki/RGB.

[9] Wikipedia - YUV.
http://en.wikipedia.org/wiki/YUV.

[10] Wikipedia - YUV 4:4:4.
http://en.wikipedia.org/wiki/YUV_4:4:4.

[11] Wikipedia - YUV 4:2:0.
http://en.wikipedia.org/wiki/YUV_4:4:4.

[12] FourCC.org - RGB ↔ YUV conversion.
http://www.fourcc.org/fccyvrgb.php.

[13] MPEG-1 Information.
http://icsl.ee.washington.edu/~woobin/ti/overview.html.

[14] MPEG-2 Information.
http://www.erg.abdn.ac.uk/research/future-net/digital-video/mpeg2.html.

[15] MPEG-4 Information.
http://www.extremetech.com/article2/0,3973,838500,00.asp.

[16] Overview of the MPEG-4 Standard.
http://www.chiariglione.org/mpeg/standards/mpeg-4/mpeg-4.htm.

[17] BBCs Overview of Dirac project.
http://www.bbc.co.uk/rd/projects/dirac/.

[18] SourceForge site for dirac.
http://sourceforge.net/projects/dirac/.

[19] Theora Homepage.
http://www.theora.org/.

[20] VP3 Codec Homepage.
http://www.on2.com/.

[21] MP3 Information.
http://hotwired.lycos.com/webmonkey/00/31/index3a.html?tw=multimedia.

[22] AAC Information.
http://www.mpeg.org/MPEG/aac.html.

[23] AAC Information.
http://www.mp3-tech.org/aac.html.

[24] International Organization for Standardization. *Information technology – Coding of audio-visual objects – Part 3: Audio*, volume ISO/IEC 14496-3. International Organization for Standardization, Geneva, Switzerland, 2001.

[25] Ogg Vorbis/Speex.
http://www.vorbis.org/.

[26] GSM 06.10 Audio Codec Information.
http://kbs.cs.tu-berlin.de/~jutta/toast.html.

[27] Comparison of MPEG-4 CELP to other codecs.
http://www.tnt.uni-hannover.de/project/mpeg/audio/public/w2424.html.

[28] CELP information.
http://www.vialicensing.com/products/mpeg4celp/standard.html.

[29] iLBC codec information.
http://www.ilbcfreeware.org/.

[30] General video streaming information.
http://www.streamingvideos.com/streamingvideos/strmhome.html.

[31] Windows Media Services.
http://www.microsoft.com/windows/windowsmedia/9series/server.aspx.

[32] Helix Server - Real Media streaming server.
https://helix-server.helixcommunity.org/.

[33] Darwin Streaming Server - OpenSource Quicktime streaming server.
http://developer.apple.com/darwin/projects/streaming/.

[34] ASF format information.
http://msdn.microsoft.com/library/en-us/wmform/htm/overviewoftheasfformat.asp.

[35] Apples QuickTime.
http://www.apple.com/quicktime/.

[36] QuickTime Streaming Servers Module documentation.
http://developer.apple.com/documentation/QuickTime/QTSS/QTSS.pdf.

[37] OGG Bitstream Information.
http://www.xiph.org/ogg/vorbis/doc/oggstream.html.

[38] RFC 3640 - RTP Payload Format for Transport of MPEG-4 Elementary Streams.
http://www.ietf.org/rfc/rfc3640.txt.

[39] International Organization for Standardization. *Information technology – Coding of audio-visual objects – Part 12: ISO base media file format*, volume ISO/IEC 14496-12. International Organization for Standardization, Geneva, Switzerland, 2004.

[40] International Organization for Standardization. *Information technology – Coding of audio-visual objects – Part 14: MP4 file format*, volume ISO/IEC 14496-14. International Organization for Standardization, Geneva, Switzerland, 2004.

[41] RFC 1889 - Real-time Transport Protocol specification.
http://www.ietf.org/rfc/rfc1889.txt.

[42] RTP Information.
http://www.cs.columbia.edu/~hgs/rtp/overview.html.

[43] RFC 3016 - RTP Payload Format for MPEG-4 Audio/Visual Streams.
http://www.ietf.org/rfc/rfc3016.txt.

[44] Overview of RTSP.
http://www.tml.hut.fi/Studies/Tik-110.300/1998/Essays/rtsp.html#RTSP-OVER.

[45] RFC 2327 - SDP: Session Description Protocol.
http://www.ietf.org/rfc/rfc2327.txt.

[46] SMIL Overview.
http://www.fluition.com/whatissmil.html.

[47] SMIL Information.
http://www.empirenet.com/~joseram/.

[48] Digital Television via IP Multicast - Final Report.
http://www.doc.ic.ac.uk/project/2003/362/g0336215M/site/docrep/finalreport.doc.

[49] RS485 Information.
http://www.hw.cz/english/docs/rs485/rs485.html.

[50] Panasonic Protocol Information for WV-CS850(A)(B) and WV-CS860A.
http://www.panasonic.fr/cctv/hl/paop/combip/Protoc_CS850_860.pdf.

[51] Professor Guang-Zhong Yangs Notes on Motion Tracking.
https://www.doc.ic.ac.uk/~gzy/teaching/multimedia/notes/mm-notes-6.pdf.

[52] Hauppauge, manufacturers of the WinTV card.
http://www.hauppauge.co.uk/.

[53] MPEG4IP: mp4live and other useful MPEG-4 encoding / decoding programs.
http://mpeg4ip.sourceforge.net/index.php.

[54] Java Media Framework Homepage.
http://java.sun.com/products/java-media/jmf/.

[55] MPEG-4 video for JMF.
http://www.alphaworks.ibm.com/tech/mpeg-4.

[56] FFmpeg encoding and decoding libraries for MPEG-4 and others.
http://ffmpeg.sourceforge.net/.

[57] JFFmpeg - JMF Wrapper for FFmpeg.
http://sourceforge.net/projects/jffmpeg/.

[58] PostgreSQL Website.
http://www.postgresql.org.

[59] SOAP specification.
http://www.w3.org/TR/2000/NOTE-SOAP-20000508/.

[60] Stunnel – Universal SSL Wrapper.
http://www.stunnel.org/.

[61] Camera Pin Assignments.
http://www.hofland.nl/download/PANASONIC%20850_P.PDF.

[62] JAR Signing.
http://www.doc.ic.ac.uk/~dcw/signing.html.

[63] General video encoding and transcoding information.
http://www.doom9.org/.

[64] Panasonic, contains information about similar camera models.
http://www.panasonic.co.uk/.

[65] Linux Kernels and updates.
http://www.kernels.org/.

[66] Video Lan, open source streaming server and client.
http://www.videolan.org/.

[67] ffdshow - Windows DirectShow version of ffmpeg.
http://ffdshow.sourceforge.net/.

[68] FAAC - AAC audio encoder and decoder.
http://faac.sourceforge.net/.

[69] Information about the Real Time Streaming Protocol.
http://www.rtsp.org.

[70] Streaming Video Information.
http://www.videorelay.com/streaming_qa.html.

[71] Hardware H.264 (MPEG-4 extension) encoder.
http://www.provideo.com.tw/PV250.htm.

[72] Hardware MPEG-4 encoder.
http://www.icpamerica.com/IVC_4200.php.

[73] Multicast, unicast information.
http://ntrg.cs.tcd.ie/undergrad/4ba2/multicast/.

[74] International Organization for Standardization. *Information technology – Coding of audio-visual objects – Part 1: Systems*, volume ISO/IEC 14496-1. International Organization for Standardization, Geneva, Switzerland, 2004.

[75] International Organization for Standardization. *Information technology – Coding of audio-visual objects – Part 2: Video*, volume ISO/IEC 14496-2. International Organization for Standardization, Geneva, Switzerland, 2004.

[76] Motion Estimation.
http://www.cs.cf.ac.uk/Dave/Multimedia/node259.html.

[77] RogueWave SourcePro DB.
http://www.roguewave.com/products/sourcepro/db/.