

DESKTOP CONFERENCING SYSTEM

By

MAZLIZAN FITRI BIN MOKHTAR

**Dissertation submitted in partial fulfillment of
the requirements for the
Bachelor of Technology (Hons)
(Information System)**

DECEMBER 16, 2004

Universiti Teknologi PETRONAS
Bandar Seri Iskandar
31750 Tronoh
Perak Darul Ridzuan

t

TK

5105.2

.M475

2004

- 1) Image transmission
- 2) Data compression (Telecommunication)
- 3) IT / IS -- theory

CERTIFICATION OF APPROVAL

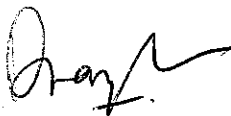
Desktop Conferencing System

by

Mazlizan Fitri Bin Mokhtar

A project dissertation submitted to the
Information System Programme
Universiti Teknologi PETRONAS
in partial fulfillment of the requirement for the
BACHELOR OF TECHNOLOGY (Hons)
(INFORMATION SYSTEM)

Approved by,

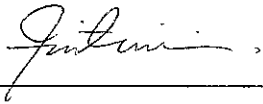


(Mr. Anang Hudaya Muhamad Amin)

UNIVERSITI TEKNOLOGI PETRONAS
TRONOH, PERAK
DEC 2004

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.



MAZLIZAN FITRI BIN MOKHTAR

ABSTRACT

Up until recently the whole area of video conferencing has proved to be an expensive and tricky technology to be working with. Most of the video conferencing technologies can be found in a large room video conferencing system with sophisticated and expensive conferencing equipments. And in other hand, teaching and learning process is still limited by physical boundaries. The main idea of this project is to improve communication and correlation among students, lecturers and tutors. The methodology chosen for the development of this project is Prototyping system development methodology. It consists of Requirement Analysis, Design Prototype, Evaluate Prototype and Project completion. In Requirement Analysis Phase, the requirements of the application and the functional specification are determined followed by Design Prototype Phase where all the critical part of this project is developed. These include the Graphical User Interface (GUI) development and the coding of this application. The third phase is Evaluate Prototype Phase where the testing phase took place. Each sub-component is tested to make sure it met all requirements. Once all components of the application is tested and all requirements are satisfied, the last phase, that is Project Completion Phase are considered completed whereby final documentation are to be developed before the final presentation. As a conclusion, this project aims to improve current communication style. It consumes communication technology effectively whereby the processing power of desktop computers has almost reached a level to become comfortable with processing the multimedia data. In addition, advances in the bandwidth availability on the internet and on LAN's/ WAN's has given the networks the ability to handle the real time streaming media data.

ACKNOWLEDGEMENT

Bismillah ar-Rahmani Ar-Raheem

In the Name of Allah, The Most Compassionate, the Most Merciful

In order to complete this report, I have been doing researches over internet and reading materials over video conferencing technologies. In this Desktop Video Conferencing project, I would like to thank:

1. **Mr. Anang Hudaya bin Muhamad Amin**, my supervisor (for giving me the guidelines and ways in producing a good output and full support in terms of knowledge input along this internship)
2. **The Backbone of FYP Committee** – Mr. Mohd Nor Ibrahim and Ms Vivien, and all IT/IS lecturers, (for giving full commitment in term of providing info about the final year project)
3. **Universiti Teknologi PETRONAS** – all UTP staff (for the full cooperation and providing me very convenient places to complete the project with the provided utilities)
4. **My parents**, Mr. Mokhtar bin Murad and Mrs Uzaiyah binti Yakob and **family** who supports me financially and mentally
5. **Friends**, for their support and information over my Final Year Project
6. Those who involved directly and indirectly towards the project.

TABLE OF CONTENTS

ABSTRACT	i
ACKNOWLEDGEMENT	ii
TABLE OF CONTENT	iii
LIST OF FIGURES	v
ABBREVIATIONS AND NOMENCLATURES	vi
CHAPTER 1: INTRODUCTION	1
1.1 Background of Study.....	1
1.2 Problem Statement.....	2
1.2.1 Sophisticated and Expensive Conferencing Equipment.....	2
1.2.2 Teaching and Learning Process Limited by Physical Boundaries	2
1.2.3 Rapid Technology Evolvement.....	3
1.3 Objective.....	3
1.4 Scope of Study.....	4
CHAPTER 2: LITERATURE REVIEW AND THEORY	5
2.1 Video Conferencing Technologies.....	5
2.1.1 Types of Video Conferencing.....	5
2.1.2 The H.323 Video Conferencing Standard...	7
2.2 Media Capture and Compression Technology.....	10
2.2.1 Codec - compression/decompression.....	10
2.2.2 Audio Capture.....	12
2.2.3 Audio Quantizing.....	12
2.2.4 Audio Compression.....	13
2.2.5 Video Capture.....	14
2.2.6 Video Compression Techniques.....	15

2.2.7	Video Codec's.....	16
CHAPTER 3:	METHODOLOGY AND PROJECT WORK.....	18
3.1	User Interface Design.....	18
3.2	Project Methodology.....	21
3.3	Tools Required.....	25
3.3.1	Hardware Requirements.....	25
3.3.2	Software Requirements.....	25
CHAPTER 4:	RESULTS AND DISCUSSION.....	26
4.1	Functional Specification.....	26
4.1.1	Functional Specification for Client.....	26
4.1.2	Functional Specification for Server.....	37
4.2	Discussion on Port Control Issues.....	39
CHAPTER 5:	CONCLUSION AND RECOMMENDATION.....	45
5.1	Conclusion.....	45
5.2	Recommendation.....	46
REFERENCES.....		48
APPENDICES.....		49

LIST OF FIGURES

- Figure 2.1: H.323 Video Conferencing
- Figure 3.1: Graphical Presentation of Prototyping Methodology
- Figure 3.2: Initial prototype design
- Figure 3.3: Actual prototype design
- Figure 4.1: Actions available after starting the client application
- Figure 4.2: Options available after capturing media
- Figure 4.3: Functions available after logging into the server
- Figure 4.4: Functions available while in a group conference
- Figure 4.5: Functions available while chatting
- Figure 4.6: Functions available after starting the server application
- Figure 4.7: Administration of ports
- Figure 4.8: Scenario on port control issues

ABBREVIATIONS AND NOMENCLATURES

- GUI : Graphical User Interface
- LAN : Local Area Network
- WAN : Wide Area Network
- MCU : Multipoint Control Unit
- VC : Video Conferencing
- PCM : Pulse Code Modulation
- LPC : Linear Predictive Coding
- GSM : Group Special Mobile
- RTP : Real-Time Transport Protocol
- RTCP : Real- Time Control Protocol
- JMF : Java Media Framework
- RGB : Red, green, blue
- NIG : Numbers in group

CHAPTER 1

INTRODUCTION

1. INTRODUCTION

Chapter 1 explains the fundamental information of the project, which consists of background of study, problem statement, objective and scope of the project. A brief explanation of video conferencing also included in this section.

1.1 Background of Study

This report is written as a pre-requisite for undergraduate students to complete the studies Desktop video conferencing system is chosen for my title for the final year project as it could help me to enhance and in-depth with all the theories I have learnt during my years of studies.

Desktop video conferencing began over a decade ago with the introduction of expensive group of video conferencing systems. It was designed to send and receive compressed audio over dedicated links that could pledge predictable service. Standards on how the conferences were to be organized and how the video/audio streams were to travel over the network eventually evolved, but initially the systems were not very interoperable. These systems evolved and a common set of video conferencing standards, H320, emerged. However these standards limited users to the expensive equipment and facilities required for these standards. The user had the option to purchase the expensive equipment or to travel to special video conferencing enabled meeting rooms to communicate. Although these large specialist systems are still prevalent, the evolution of networks, standards and desktop computing power has allowed mid level or low level desktop systems to enter the arena and compete to provide simple and useful desktop video conferencing solutions.

1.2 Problem Statement

Up until recently the whole area of video conferencing has proved to be an expensive and tricky technology to be working with. It was not always worth to bother or spend money to implement such solutions when other communicating and collaborating options appeared more beneficial. Until now, most videoconferencing was done with room videoconferencing systems, which use very sophisticated and expensive equipment to provide high-quality sound and video. As a solution, Desktop Video Conferencing was introduced which is less expensive and use personal computer.

1.2.1 Sophisticated and Expensive Conferencing Equipment

Most videoconferencing technologies were done with large room video conferencing systems, which use very sophisticated and expensive equipment to provide high quality sound and video. The desktop video conferencing experience is often of much lower quality. Because of technical restrictions in sending video from one computer to another, the image in desktop videoconferencing is typically not the size of your whole screen, and the image quality won't be comparable to television. Although lacking in the area of quality, the novelty value of video accompanying voice and text has been proved popular on the internet. There are now many chat rooms on the internet with option to include your video as well as voice. This can be very slow and jumpy if the end to end connection has slow spots.

1.2.2 Teaching and Learning Process Limited by Physical Boundaries

Video conferencing systems are often used for distance learning in which the participants are not geographically near each other or for some other reason, are physically unable to meet. For example;

- Group members for projects/assignments need to share their ideas and work on their project at the same time.
- Communication becomes a concern when team members stay in separate places.

- Universities/ colleges team up with businesses organizations to offer employee training or certification.
- Lecturer team- communicates between remote lecturers, sharing subject matter expertise or a unique approach to a topic.
- Students meet with tutors for enrichment, remediation, or a helpful bit of personal attention.
- A librarian offers an introduction to library services and library tour for students.

1.2.3 Rapid Technology Evolvement

Technology evolved from time to time and it changed rapidly. Nowadays, most of mobile phone products have camera technology. It is not possible that we can have video conferencing between remote users just by using a mobile phone. Meetings through video conferences are being investigated by many organizations as an alternative to traveling long distances. As stated in a press release [7], 3G mobile phones with video conferencing facilities must be the most exciting advance technology.

1.3 Objectives

This project is done in two phases. In the first phase, research has been done to understand the concept of video conferencing technologies. In second phase, the design and development of the desktop video conferencing have to be done. Objectives of the project to be achieved are as follow;

- 1) Main objective is to develop a desktop vide conferencing application which enable students, lecturers, and tutors to meet and share information.
- 2) Develop a desktop video conferencing application which is less expensive and use personal computer.

1.4 Scope of Study

The scope of study is on how video conferencing technologies can enhance learning and teaching process. It is known that a versatile and stimulating multimedia environment can help to support a range of learning experiences with access to remote users within an organization network. The programming scope of the project encompassed study of the Java Media Framework, Java networking, and Java swing, the tools that would be used to build the application. Later in the development phase, some other tools may be added that can help to achieve the objectives. In order to plan and design the application, research has to be done in order to complete the task. Meanwhile, the user interfaces design for the client and the server will prioritize the application usability. Mainly, the scopes of this project are;

- Enhance learning and teaching process by using video conferencing technologies and facilities.
- Development of a desktop video conferencing system in pure Java using Java Media Framework (JMF) that can help to increase learning efficiency.
- Effective communication between individuals in the organization using desktop conferencing system whereby it enables the transmission of audio, images and text.
- Development of user interfaces design that prioritizes the application usability.

CHAPTER 2

LITERATURE REVIEW AND THEORY

2. LITERATURE REVIEW

This chapter contains the acknowledged findings on this field, consisting of relevant theories, hypothesis, facts and data which are relevant to the objective and the research of this project.

2.1 Video Conferencing Technologies

Video conferencing is basically the transmission of video and audio between separate physical locations. This is achieved through the use of capture devices, cameras and microphones, video conferencing software, processors, storage devices, network devices and video display units. In addition, facilities such as chat, shared whiteboard and document sharing generally considered the norm as part of the video conference experience. Current day video conferencing systems provide a rich and useful array of functions designed to solve the most complex of communication and collaboration problems.

2.1.1 Types of Video Conferencing

From researching video conferencing vendors on the internet there appears to be a huge variety of vendors with many various different solutions to the video conference requirement. With interoperability a key factor in designing such systems we can develop systems to provide solutions to almost any requirement using these vendors' products. Generally speaking there are three types of video conferencing systems as stated in an internet source [2];

- Desktop Unit

Desktop conferencing unit generally consist of an application with a GUI making use of the simple capture devices and software codec's on the desktop pc to communicate with other clients via high speed network connection. The media may be broadcasted to the other clients, unicast or multicast. There is generally no extra hardware involved other than the capture devices which may consist of a basic web cam and a microphone or headset. The software consists of the video conferencing application itself, the web-cam software and the codec's. The processing power of the computer and the bandwidth available are the limiting factors of desktop video conferencing.

- Roll – Abouts

Roll-about systems are complete video conferencing packages probably housed within a wheeled cabinet or stored entirely in a cupboard. They often have one or two monitors sitting on top of a cabinet, a minimum of one camera, an audio system, which may include an echo canceller to handle audio feedback. In addition an audio suppresser unit to deal with background noise may be included. These systems also contain hardware codec's probably built into a custom capture card that the cameras are plugged into. The use of a central multipoint control unit (MCU) to control the conferences would also be commonplace for these systems.

- Studio / Room – sized Systems

The most expensive video conferencing options are housed in a room or studio specifically designed for video conferencing. The whole room, from the color of the walls to the room acoustics will be tuned to suit the conferencing system. There may be many cameras and microphones for input. There may be several monitors, projectors, audio devices and document collaboration facilities. These systems are generally implemented by large multinationals or educational institutions. Sometimes these institutions rent these systems for public bookings

possibly to make financial sense of the investment. Gateways, multipoint control units and high bandwidth networks are some of the main hardware components used in such systems.

2.1.2 The H.323 Video Conferencing Standard

Source from internet [3] indicates that the H320 standard was originally developed to allow video conferencing over networks such as ISDN or dedicated T1 circuits. This allowed video conferencing systems to be developed to work within these guidelines. The hi-expense for the use of these network connections proved undesirable and standards for conferencing over local area networks or the internet evolved. TCP/IP, as used on the Internet, is now being called upon to provide less expensive and more flexible connections. In conjunction with this, a new ITU (International Telecommunications Union) standard has emerged for supporting audio/video conferencing over IP. This new standard is called H.323 and was first approved by the ITU in 1996. Since then, the standard has evolved through additional versions and also been implemented in multiple vendors' products. H323 standards based video conferencing was engineered for conferencing over packet based networks. As time has passed the standard have evolved and are now more flexible, the equipment is now less expensive, and the option for real desktop video conferencing has emerged.

A source from internet [1] indicates that H323 is an international telecommunications union standard for video conferencing over IP and specifies compulsory and optional requirements in several areas in order for a conference to be initiated, conducted, maintained and terminated. The standard defines the major components that may be part of the conferencing system, terminals, gateways, gatekeepers, and multi-point control units. The standard has essentially promoted interoperability between different vendors of video conferencing systems in such a way that some of these vendors have formed strategic alliances to ensure interoperability between their products.

The following is a basic diagram of the H.323 video conferencing standard structure.

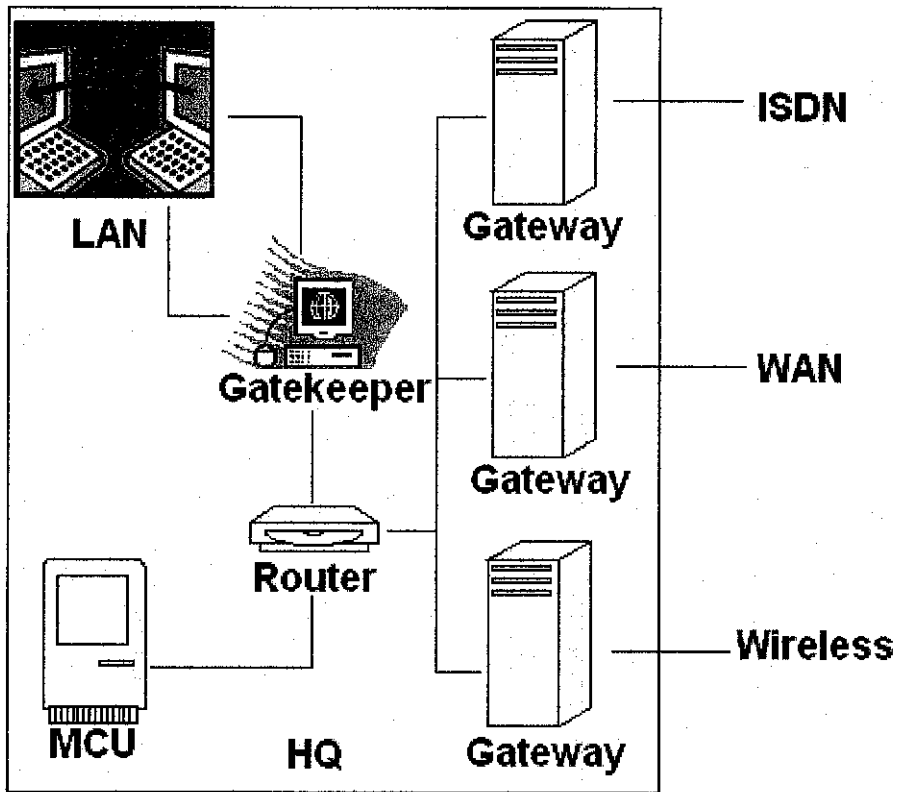


Figure 2.1

The following is the description of some of the components involved.

- Multipoint Control Unit (MCU)

MCU's allow for conferencing functions between three or more clients. In addition data sharing can be managed by the MCU. An MCU normally contains two parts:

- Multipoint controller (MC) that handles the signaling and control messages necessary to setup and manage conferences.
- Multipoint processor (MP) that accepts streams from endpoints, replicates them, and forwards them to the correct participating endpoints. This process can be controlled in different 'modes'. Generally the audio stream is multiplexed and served to all clients but the video can be controlled differently.

The stream to all clients can be switched onto the client that is currently speaking or someone can have chairman control, I.e. one chosen client can select who is on view by everyone. Another option is where all incoming video is combined into a single split screen video and sent to all clients, so everyone can see everyone else.

An MCU can implement both MC and MP functions, in which case it is referred to as a centralized MCU. Alternatively, a decentralized MCU handles only the MC functions, leaving the multipoint processor function to the endpoints. Each endpoint serves its media to each other clients leaving the MCU to handle the administrative side of the conferencing. This is basically the method I have used for my application.

Arguments for and against centralized versus decentralized multipoint conferencing are not unlike those surrounding the debate of centralized server-based computing versus peer-to-peer computing.

- Gatekeeper

The gatekeeper is an optional H.323 component. It provides several important services and will likely be a part of most H.323 networks. A gatekeeper must act as a security guard for the conferencing zone. Gatekeepers introduce the concept of zone, a zone being a collection of all video conference clients, gateways and MCU managed by the single gatekeeper. The gatekeeper provides services like admission control, bandwidth management and call authorization within its zone.

- Gateways

Gateways are devices which connect one standard protocol to another. Typically, they have been for computer data protocols, but gateways are now being designed to interconnect different videoconferencing standards. A videoconferencing gateway usually sits at the boundary between a Local Area Network (LAN) and the outside world. Current gateways are devoted to converting H.323 on the LAN to H.320 on the wide area network, or H.323 on the LAN to analogue voice lines on the wide area network. A variation on this is

a gateway between H.323 on the Internet and local voice service; this uses the Internet to cover most of the distance, and the local phone network to get the wire that last mile to the phone.

Gateways are a necessity when a conference is to be held between two or more clients using different protocols. For example, if one client uses only H.323 protocol and the other client uses H.320 (ISDN) protocol, then a gateway is needed at one end to handle the conversion. Also, as mentioned above, gateways are also a practical way of bringing in calls from regular telephone systems (POTS). Another usage example is if someone is on the road on their wireless phone, they can still participate in a conference via a gateway which bridges them into the conference.

2.2 Media Capture and Compression Technology

The analogue audio and video being taken into the capture devices on desktop go through numerous processes, using software and hardware, using many different protocols in the path to the other user's output devices. The following sections describe some of the technology involved in sending media along this path.

2.2.1 Codec - compression/decompression

A source from internet [11] indicates that Codec compresses/decompresses the video or audio data sent over the network. Uncompressed real-time media data is generally too large to be sent over the network and needs to be compressed into a smaller bits-per-second rate for transmission. The codec uses a particular algorithm to compress the data to a certain compressed transmittable format (for each format there is a particular codec that compresses raw media data to that format using its specific algorithm).

The procedure goes as follows. The raw media data is captured by the capture device using its software. The video conferencing (VC) software uses a codec to compress the stream of data into a small enough size so as it fits onto the network to

provide the endpoint with an acceptable real-time stream. The VC software at the other end uses another codec to convert the media from the compressed format to a presentable (raw media) to be output to the screen or speakers. The efficiency of the codec's combined with the speed of the network really determines how 'real-time' and smooth the media will be presented to the user.

The word codec is just a general word for the compression/decompression process. The process can be carried out in special hardware devices that take the raw media input (the capture devices would be plugged into such devices) and output the compressed stream (to be used by the VC software). Hardware codec's are generally faster in completing their compression/decompression task, making near real-time communication more likely, and hardware codec's normally carry their own processing power "on-board" such that they do not rely on the resources of the underlying system. For instance, in the case of a desktop system, using a hardware codec may mean that there is no need to use the PC's processor, or that it will be able to run other applications on PC while simultaneously participating in a video conference. On the other hand, software codec's are generally less expensive and easier to install (no special hardware required), but they tend to produce lower quality conferencing with low frame rates. In H.323 desktop video conferencing systems, the codec typically resides on an interface board or in a software application. In the videoconferencing world, a codec converts analogue video and audio signals from a capture device to digital signals for transmission over digital circuits, and then converts the digital signals back to analogue signals for display. The format of the data is to be compressed so it can have varying levels of quality and size. Some information is sacrificed in the process of compression, which may result in diminished picture and sound quality. Codec's compress the data using different algorithms that might include losing frames or reducing redundant data in sequential frames.

2.2.2 Audio Capture

K. Pohlmann, (1985) [8] indicates that digital audio data is usually described using the following three parameters: sampling rate, bits per sample, and number of channels. The sampling rate is the number of samples taken per second. 'Bits per sample' is the number of bits used to represent each sample value. Number of channels is one for mono, two for stereo, etc.

- Sampling

The frequency or amplitude of the sound waves entering a microphone is measured in hertz which means cycles per second. The human ear can typically hear sound waves at frequencies between 20 Hz and 20 kHz. Speech is typically between 40 Hz and 4 kHz and while speaking the amplitude continuously varies between these extreme values. To capture this signal digitally the amplitude value of the analogue wave needs to be measured at regular intervals called sampling. According to the Nyquist theory of signal processing, to faithfully represent a signal of a certain frequency, the sampling rate must be at least twice that of the highest frequency present in the signal. Using Nyquist's theory, 8 kHz is a sufficient sampling rate to capture the range of human voice, and 40 kHz is a sufficient sampling rate to capture the range of human hearing. In practice typical audio rates sampled by desktop audio capture devices range from 8 kHz to 48 kHz.

2.2.3 Audio Quantizing

Sampled values of the sound waves amplitude can be picked from a wave height divided into a number of levels. The number of levels depends on the amount of bits used to represent the sample. Using desktop audio capture devices can typically choose a capture format that specifies 8 bit or 16 bit sampling (my application supposed has a default value of 8bit sampling but can be changed to any supported format provided by the capture device). 8bit sampling gives 256 different values possible to sample from

(16bit samples gives 65536, but double bandwidth required, for very little audible difference in on a desktop Video conferencing system)

2.2.4 Audio Compression

Although definitely a child when compared its big brother Video, the audio stream can require a considerable amount of bandwidth to transmit. There are many techniques used to compress digital audio. Typically these are techniques that can achieve real-time compression and decompression in software or hardware (although if encoding using software there is the CPU usage overhead to consider). Some techniques apply to general audio signals and some are designed specifically for speech signals. Some of the techniques commonly used by audio codec's for desktop video conferencing systems are described below. The followings are findings from D. Pan, Digital Technical Journal, Vol. 5 No. 2. [9].

- **PCM (Pulse Code Modulation)**

With PCM encoding methods, each sample is represented by a code word. Uniform PCM uses uniform quantizer step spacing. By performing a transformation, the quantizer step spacing can be changed to be logarithmic allowing a larger range of values to be covered with the same number of bits. There are two commonly used transformations: mu-law and A-law. These transformations allow 8 bits per sample to represent the same range of values that would be achieved with 14 bits per sample uniform PCM. This translates into a compression ratio of 1.75:1 (original amount of information: compressed amount of information). Because of the logarithmic nature of the transform, low amplitude samples are encoded with greater accuracy than high amplitude samples.

- **LPC (Linear Predictive Coding)**

Linear Predictive Coding is one of the encoding methods designed specifically for speech. By using models of the characteristics of speech signals, these

encoding methods achieve good results for speech data. However, these methods usually do not work well for non-speech audio signals. A LPC encoder fits speech signals to a simple analytic model of the vocal tract. GSM (Group Special Mobile) encoding uses a variation of LPC called RPE-LPC (Regular Pulse Excited - Linear Predictive Coder with a Long Term Predictor Loop).

- **GSM (Group Special Mobile)**

GSM began as a European cellular phone speech encoding standard. It compresses 160 13-bit samples (2080 bits) to 260 bits which is an 8:1 compression ratio. For 8 kHz sampling, this means GSM encoded speech requires bandwidth of 13 kbps.

2.2.5 Video Capture

Video is a sequence of still images. When presented fast enough it gives the appearance of fluid motion. An internet source [10] indicates that ordinary television uses a display system called Phase Alternating Line, (PAL) which is streamed at a rate of 25 frames per second and has a frame of size 576x720 pixels. Desktop video conferencing applications use video as input which involves capturing video from capture devices such as web cams.

Each frame of video is 'grabbed' by the web-cam software as a description of the pixels making up the image. The following is a quick overview of how these pixels are represented in terms of their color and brightness.

- **Color Schemes**

The human retina has three types of color photoreceptor cone cells, receptive to the 3 primary colors Red, Green and Blue. Because there are exactly three types of color photoreceptor, three numerical components are necessary and sufficient to describe a color, although in printing it is convenient to use a fourth (black) component. Different color schemes are used for different purposes but generally for desktop video the color schemes used are either RGB or YUV.

➤ Color schemes - RGB (red, green, blue)

The RGB color space is a three dimensional coordinate system where each axis representing the brightness of each of the three primary colors. The RGB color scheme contains, for each color, information about the color and the luminance.

➤ Color schemes - YUV (Y=luma, U=Red-luma, V=Blue-luma)

The YUV color scheme is derived from the above RGB scheme but separates the common luminance from the color saturation itself. It was originally devised as a scheme to allow color movies be watched on black and white televisions. Brightness and color information are treated differently by the human visual system. Humans are more sensitive to changes in brightness than changes in color. Because of this, a special component is used to represent brightness information. This component is called the luma and is denoted by the symbol Y. The luma is what black and white TVs use to present their picture. The U and V channels subtract the luma values from Red (U) and Blue (V) to reduce the color information. All these values, Y, U and V, can then be reassembled to determine the mix of Red, Green and Blue. One of the main advantages of the YUV format over the RGB format is that uses less bandwidth to produce the video stream because the luminance information is being transmitted only once rather than three times with each color.

2.2.6 Video Compression Techniques

There is too much data involved in describing raw streaming video for it to be transmitted over desktop level communication systems at an effective frame rate. The video stream needs to be compressed. Video compression is typically lossy in that data is lost in the compression process. There are different encoding algorithms available to

perform this process that can compress to varying percentages of original size. The information lost in this compression is generally redundant, and can be omitted causing only minimum loss of viewing quality for the user.

An internet source [11] indicates that the different schemes use some common techniques to 'loose' the information including Color Space Sampling and Interframe Reduction.

- **Color Space Sampling**

Color Space Sampling involves reducing the amount of color values needing to be encoded in representing each frame. If the image is captured in the YUV format, the U and the V variables can be sub-sampled, because the human eye is less sensitive to changes in the color that changes in the light (U luminance value).

- **Interframe Reduction**

Further compression is achieved by calculating differences between successive frames and transmitting only those differences. Further savings in bandwidth can be made by predicting such changes and transmitting only the information which varies the predictions. This technique uses motion vectors to predict changes due to motion in the picture. In both these cases there is the need to periodically send full key frames (intra-frames). These frames provide a new point of reference for the succeeding difference frames (inter-frames). This process works best when the user is limiting his/her movement on camera and is perhaps not wearing bright colored clothes in a room without complex patterns on the wallpaper. The less movement and complicated patterns in the frames the less inter- frame data to be transmitted.

2.2.7 Video Codec's

The compression process may be performed in hardware or in software, but in the arena of desktop video conferencing, codec's are normally software processes which are quite

intensive on CPU usage, whereas the large systems normally provide hardware accelerated compression/decompression.

The following are a couple of the more common video compression schemes. Compression to these formats on the desktop is normally achieved in software. The bandwidth required varies as does the CPU usage in actually processing the media stream to these formats.

- Motion JPEG

This encoding format involves losing data in techniques called Quantization and Variable Length Encoding. The techniques can be complicated, this is a huge subject area and can be read in publications like *'Video Compression'*, Peter Symes, ISBN: 0-07-063344-4 Published by McGraw-Hill April 1998. In general the JPEG encoding scheme is best used in the area of still pictures or video surveillance (one way stream, low frame rate, good picture) rather than real time conferencing as it does not result in high enough compression to suit low bandwidth applications, although the image quality is better if the high bandwidth is available. One of the advantages of the JPEG format is that the level of compression or quality can be set by the user.

- H263

H.263 is the standard desktop video conferencing codec. As such, it is optimized for low data rates and relatively low motion. H.263 is an advancement of the H.261 standard and it was mainly used as a starting point for the development of MPEG. H.263 works hard to achieve the high levels of compression, it is fairly CPU intensive and it might slow down mid range machines. It has a strong inter-frame compression component indicating it is most effective when encoding low movement video streams.

CHAPTER 3

METHODOLOGY AND PROJECT WORK

3. METHODOLOGY

Chapter 3 features the detailed description of methodology and procedure of completing this project. This methodology is implemented in order to ensure that the project is running as required. An overview of the network is also described in this chapter.

3.1 User Interface Design

For most people the User interface is the Application. What users want is the developer to build the system to be easy to use, and for it to achieve its task. A well designed user interface is one that should be intuitive to the user and by using it, users should understand the problem and be able to work with the system without any user manuals or training.

In the business sense the better the user interface the lower the cost. The easier it is to use, the less the training and support costs, and the better it is, the more satisfaction users will get from working with the application.

There are some issues whereby users assumed that the application will meet their needs in terms of functionality. This is taken as an assumption. There is no flexibility in the functionality, whereas when it comes to the design of the user interface a little flexibility comes into play. If the programmer or analyst knows the subject domain well or has worked with the system, he can use this flexibility to add the icing to a well baked cake. The user interface can be pleasing, intuitive, and efficient for the user to perform the task, the project will meet its objectives and everyone is happy. If the user interface makes it difficult to use the system, it won't be used; it's as simple as that. People will meander back to the traditional system or old ways of completing the task.

Although it may be of critical importance to ensure an application meets its functional needs, it can be equally important to address the way in which the user can use it to meet these needs.

For this application there is a base functionality that needs to be met, but after having lots of flexibility in designing any extras, and in creating a most intuitive and useful user interface. To get this some similar applications have been looked, and done some research in the area of user interface design, especially in the design of desktop video conference applications. The following is some areas according to Theo Mandel, (1997) that have been found interesting (and taken on board during development) and particularly for the application.

- **Consistency.**

There is a need to be consistent in the actions required to carry out the functions. e.g. to join a group or to chat with a member, requires highlighting from the list and then clicking the button. The user is always asked to confirm the function. For all the main functions there are 3 ways of activating; by button, by toolbar item or by mnemonic key-strokes. This consistency helps to get used to the application quickly, users can build a mental model of the system. A mental model of the system is the user's natural approach to the system. After using a system for a while the user can become used to the user interface, and if designed consistently, users will be able to navigate intuitively through the various functions.

- **Color**

The color scheme used in the application is designed to blend with java's default colors. Button, icons, and graphics are chosen appropriately to look consistent with the whole system. Because of the graphical nature of well chosen colors and icons really compliment the look and feel. Especially during a conference when all the video components are on a screen.

- **Font**

Every typed word on screen is of the same font type, size. The font color only changes upon selection.

- **Permissions**

When a function has been carried out or is not currently available, it is 'greyed' out. This consistency aids the user in building the mental model of the application.

- **Screen layout**

The split-pane object containing the desktop object for the internal frames really proves useful here in this application. It allows all the functions to be tucked away nicely to a vertical panel on the left and have a whole blank desktop on the right to work with. It is probably not desirable to allow resizing of the main window or split-pane-barrier. The ability to resize the main window may not suit the design of this application and has not been included.

- **Monitor**

The use of a monitor screen, presenting the users own video, is the standard in desktop video conferencing applications. The ability to show and hide this monitor also appears useful. I have made the monitor unresizable, so it is always the same size as it is captured to ensure that the user sees on his screen exactly what other clients is receiving. In addition, the frames can be positioned where the user wishes.

- **Frame positioning**

Although all the internal frames on the desktop are moveable they initially appear in their own appropriate position. Each new client video window pops up at the top of the screen from left to right. This is to encourage the user to look as close to the web-cam as possible to aid the perception of eye contact with the clients. The chat windows initially appear on the bottom left of the screen as close to the keyboard as possible, and the audio windows initially appear minimized. This is partly due to space on the desktop (minimize clutter) and partly to the less importance of their function (compared to the video).

3.2 Project Methodology

Methodology chosen for the development of this project is Prototyping. Prototyping is an iterative analysis technique in which users are actively involved in the mocking-up of screens and reports. The purpose of a prototype is to show people the possible design(s) for the user interface of an application. In designing user interface, 4 steps of this iterative process allowed me to define these. Diagram shown in the Figure 3.1 will describe these 4 processes in detailed.

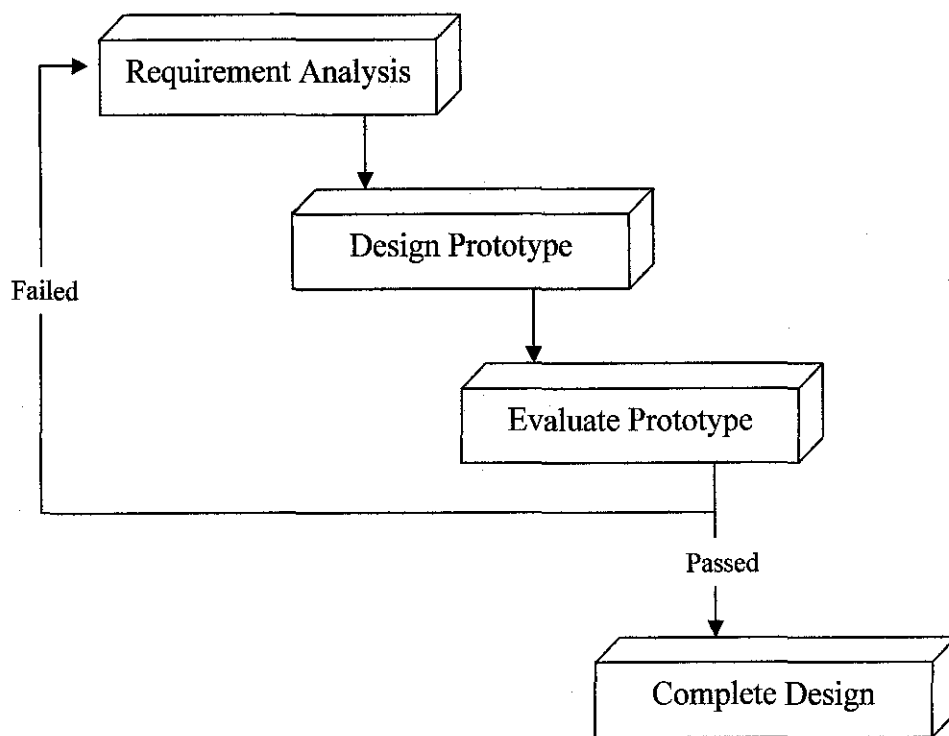


Figure 3.1 Graphical Presentation of the Methodology Used

- **Requirement Analysis**

Needs and requirements for the user interface were determined. From there, objects, functions, menus, messages have been recognize. From there, the objects that need to be present to the user were decided. The input and output for the application was also identified in order to move to the next phase.

- **Design Prototype**

The initial prototypes involved basically playing around with the technology, and getting a feel for the code. In this phase, extra time may be needed since this is a challenging task. At this point, the application was able to get the JMF capturing media and sending a stream to another IP address. In this phase, prototype has been built and the prototype was done as shown in Figure 3.2.

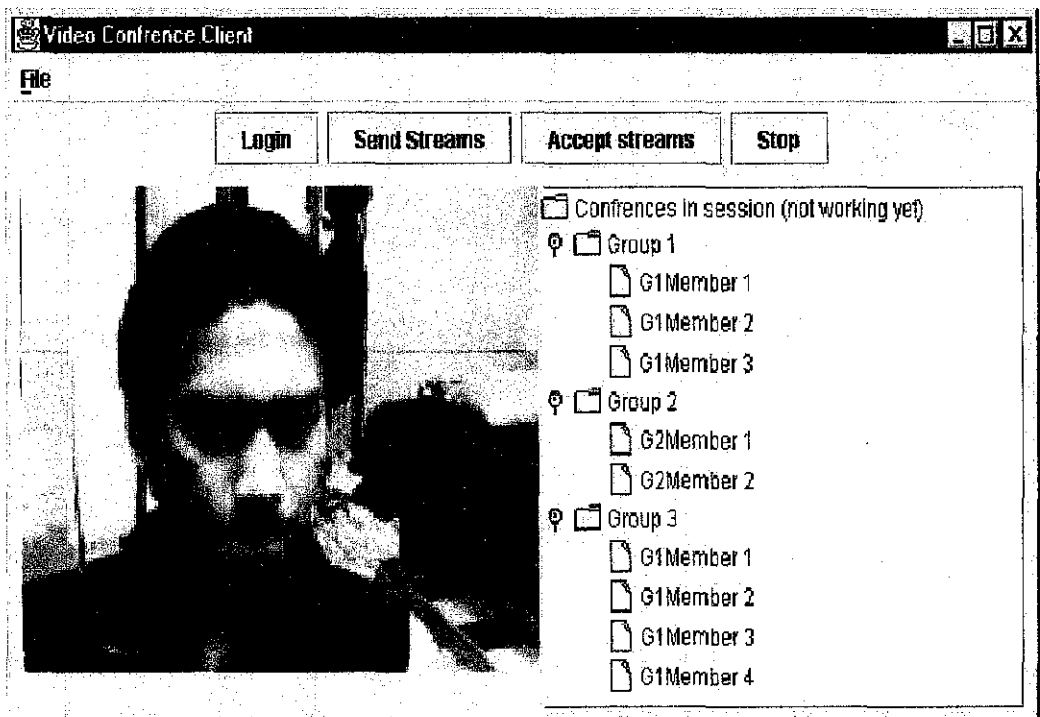


Figure 3.2

- **Evaluate Prototype**

After a prototype was built there is a need to verify that it met the needs as specified in the functional specification. Then, take a look at 3 useful questions. What was good about the prototype? What was bad about the prototype? What's missing from the prototype? After answering these questions, the parts that need to be modified can be determined.

- What was good about the prototype?
 - ✓ The prototype met most of the functional specification requirements.
 - ✓ The prototype was intuitive, helpful and pleasing to the eye.
- What was bad about the prototype?
 - ✓ The prototype limited the number of clients on the screen.
 - ✓ This prototype placed each client's video/audio into a panel that was a static size.
 - ✓ Users have to select from the list of conference members and the image window will only display the selected conference member.
 - ✓ Limited functions
- What was bad about the prototype?
 - ✓ The prototype limited the number of clients on the screen.
 - ✓ This prototype placed each client's video/audio into a panel that was a static size.
 - ✓ Users have to select from the list of conference members and the image window will only display the selected conference member.
 - ✓ Limited functions
- What was missing from the prototype?
 - ✓ Button panel for GUI.
 - ✓ Chat facility
 - ✓ Limited image windows.
 - ✓ Client status on title bar (i.e., capturing, logged- in, in-group or chatting)
 - ✓ Unicast streaming.

- **Complete Design**

Prototyping process can be stopped after the evaluation process is free from error, no longer generating any new requirements, or is generating a small number of not-so-important requirements. After implementing the above amendments, new additions and deletions, the application was in presentable state – Figure 3.3 – but there are still lots of further improvements can be done in future.

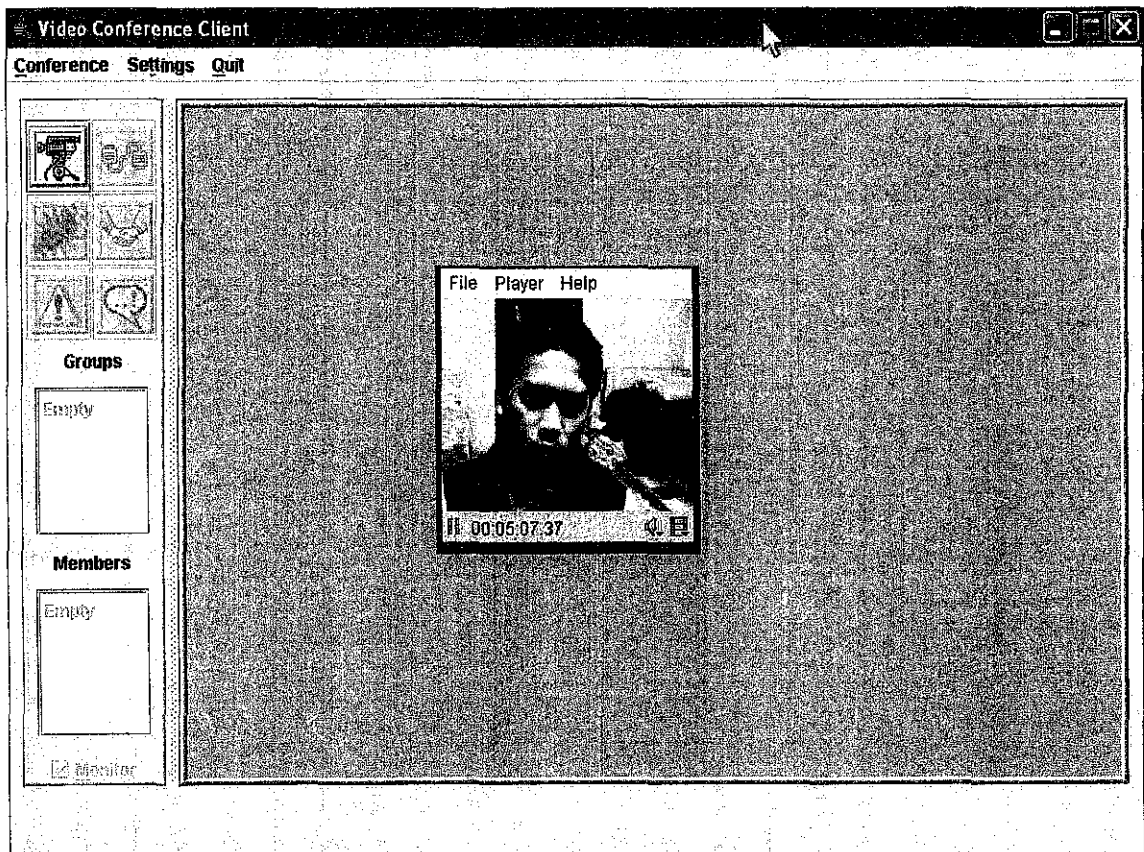


Figure 3.3

3.3 Tools Required

3.3.1 Hardware Requirements

Suggested minimum requirement for development of this project will be:

- Platform – based application, which is Windows 2000
- 800MHz speed processor
- 256MB RAM
- 10.2 GB Hard Disk
- 40X CD-ROM
- 128k connection
- 16MB of VGA
- Video and audio capture devices
- Networked computers (also with video and audio devices)

3.3.2 Software Requirements

These are several software needed to develop the application, any addition depends on situation.

- Java Media Framework
- Java 2 SDK
- Swing
- Client/ server programming
- Adobe Photoshop 6.0

CHAPTER 4

RESULTS AND DISCUSSION

4. RESULTS AND DISCUSSION

This chapter presents the finding or outcome of the project work. By following the methodology as stated in chapter 3, the outcome and results will be presented in this chapter. Later in this chapter, there will be discussion on some raised issues.

4.1 Functional Specification

In the Requirement Analysis phase, the functional specifications for the application have been drawn up. Normally a functional specification is defined to meet a set of user needs. i.e. a specification of exactly how a function is to be performed as specified by the user. In this case I have used the objective outlined in chapter 1, some best practices in GUI design and some guidance from my colleague to design the system.

4.1.1 Functional Specification for Client

C1 Main Frame

The main GUI for the client should be presented upon running the application. The GUI should be presented in a maximized frame with no-resize, minimize and close options. The frame should contain a menu-bar, and a split-pane of button/lists panel and a desktop to contain the internal frames.

C2 Menu – bar

The menu-bar should contain the following options. Conference, settings, quit. Each of the functions in each of the headings on the menu-bar should be reachable through the use of 'hot-key' mnemonics.

C2.1 Conference

This item should contain the options to capture, login, start- group, join-group and chat.

C2.2 Settings

This item should contain options to set the server-IP address, the server port, the chat port and the beginning local port (the port to listen for the first incoming stream).

C2.3 Quit

The quit menu- item should contain options to quit the group or quit the application. The quit application option should perform the same function as the quit button on the frame.

C3 Button/Lists panel

This panel should be a slim panel of vertical orientation on the left hand side of the application window.

C4 Application functions

Not all of the functions of the application are to be included on the button panel. Only the main 'action' functions are to be set included here. Functions like settings or quit application are only to be reached from the menu-bar. Any function included on the button-panel should perform exactly the same function as its corresponding function of the same name, reachable from the menu-bar. –Figure 4.1- shows the actions available after starting the client.

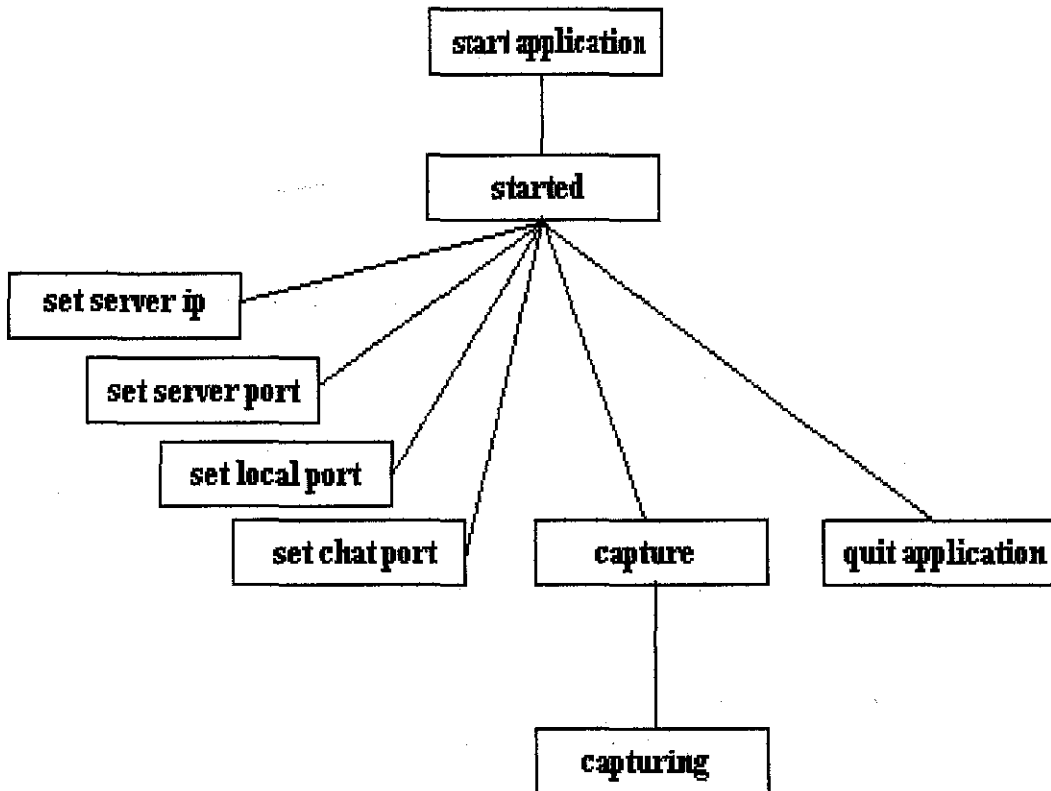


Figure 4.1

C4.1 Capture media [accessible through Button panel and menu-bar]

This function is essential to the conferencing application and is to be carried out as the first operation upon starting the application. It is assumed that the user has registered the capture devices with the JMF registry. The following procedure will take place upon activating this function.

- **Capture device selection:** The application will automatically select the first audio and video capture device from the registry's list. (this is the norm, as there is normally only one capture device available for each media type)
- **Capture format selection:** The user should be presented with the video capture formats available by the specified capture device. The pop up menu should contain the Options ok and cancel. The formats should be in a drop-down list.

The user should be able to select a capture device using the mouse or keyboard and confirm selection by clicking ok, or to renege on selection by clicking cancel.

- **Encoding format selection:** Upon selection of the preferred capture format the user should be offered an option to select an encoding format for the video. The most efficient (minimum bit-rate) audio capture and encoding formats that are JMF compatible are to be automatically selected and the user should not be offered an option to choose. This decision was made on the basis that the audio is generally of good quality with the lowest bit-rate encoding and that sometimes the JMF processor cannot process all the capture formats for the audio capture devices.

Again the user should be offered these video encoding options in a drop down list with the options to confirm selection or cancel. And again the list should be accessible through the keyboard or the mouse.

During selection of the capture and encoding formats and until the actual capture monitor is displayed (this process can take a few moments), the user should be kept informed of the progress of the capture process by small pop up messages. These should contain text like "Setting capture format..." or "setting encoding format...".

When the capture monitor is displayed it should be in the bottom right hand corner of the frame (out of the way), and of the same size as requested during the capture process. The monitor should be un-resizable (I thought this desirable so as the user always sees the size of the video that is being transmitted during a conference). The monitor frame should be movable and contain standard Java Media Framework controls at the bottom. These controls contain useful functions like start and stop the processor or a function to adjust the JPEG encoding quality of the media to be transmitted.

At the bottom of the button panel there should be a check-off box to hide/show the monitor. When the application is not capturing this should be deactivated/greyed

out. Upon starting to capture this should be activated and indicating that the monitor is on display. Upon clicking, the monitor should hide, but still be capturing.

C4.2 Login [accessible through button panel and menu-bar]

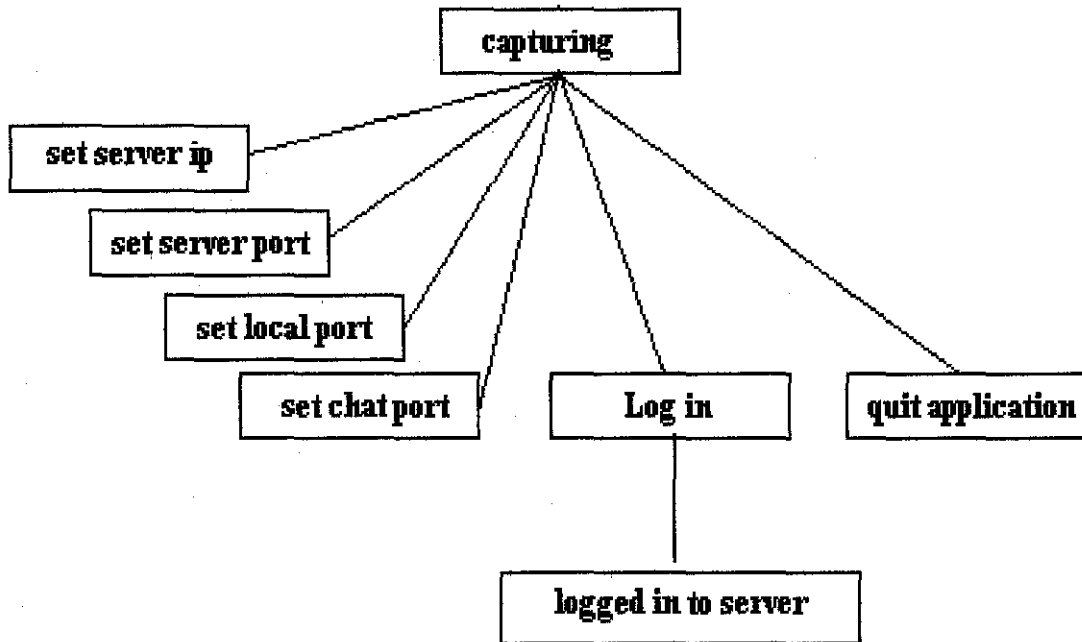


Figure 4.2

-Figure 4.2- shows the options available to the user after capturing media. The next step to conferencing is to login to the server. This function should prompt the user for a name to login under on a pop up menu with ok or cancel options. Cancel should abandon the function. Upon entering a name and clicking ok the system should contact the server, register the new user and return a confirmed login message. Upon arrival of this confirmed logged- in message from the server (almost instantaneous) the user should be presented with a confirmed logged-in pop up message. The server should also send a list of currently active groups. This list should be displayed in the groups list on the button panel. From this stage on the user is logged- in to the system until the application is quitted. -- Figure4.3- shows the available functions after logging in to the server.

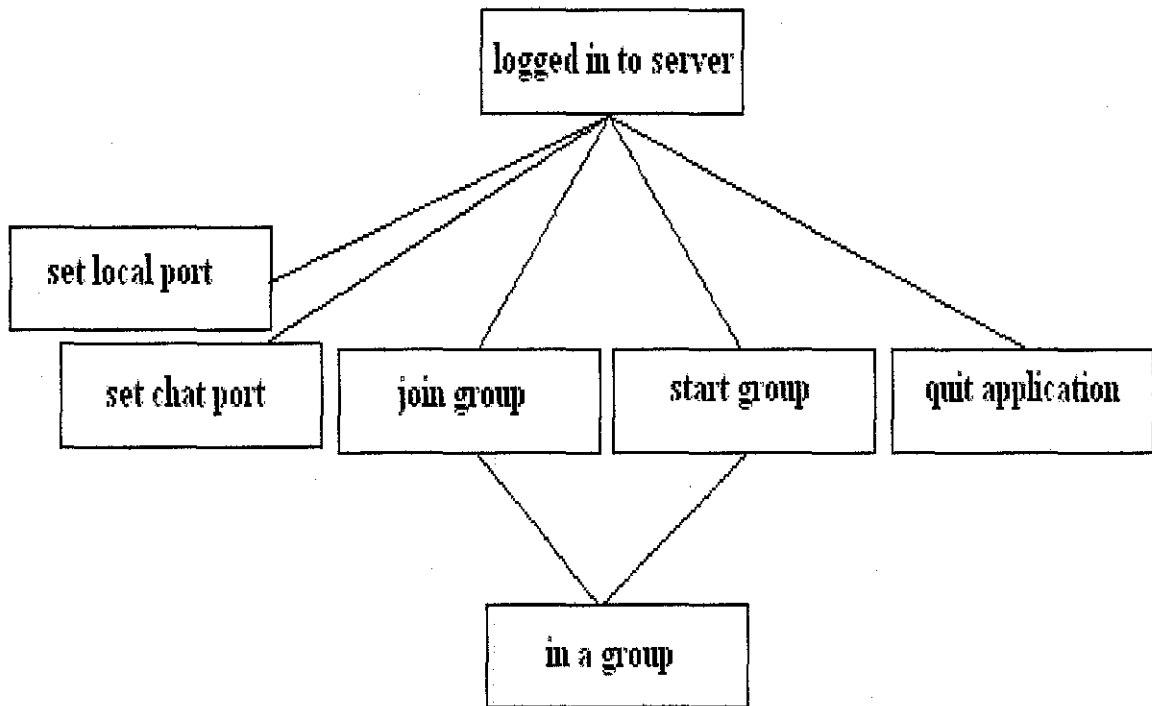


Figure 4.3

C4.3 Start Group [accessible through button panel and menu-bar]

Upon opting to start a group another popup dialog should be presented querying the name of the group to start. Again as with all the popup dialogs the options to confirm action (ok) or abandon (cancel) should be available. Upon entering a group name and clicking ok the new group request should be sent to the server, and if there is not a group currently of that name the group should be created. The server should send you a new members list for the group you're now in (only you in the members list at the moment) and send all logged-in clients the new group name for their groups panel (including yourself).

C4.4 Join Group [accessible through button panel and menu-bar]

This function should require two steps. Upon clicking the join group button the user will be prompted with a message to indicate which group to join by highlighting the group on the groups list. If the user now highlights a group or had already highlighted a group and the join group button is pressed (or activated from the menu-bar) a 'confirm joining group' message is presented. As normal,

if the user cancels here the action is abandoned. If the action is confirmed the message should be sent to the server requesting membership of the selected group. At this stage there are no security or password restrictions on the application so this process should be fairly straightforward. The client should receive the group list including the IP addresses of each member. The client should then send the media to each other member and listen for the other member's incoming media. During this process, because it can take a few seconds to complete, the user is kept informed of the progress of the action with pop up messages. When the other member's media is received it should be presented in internal frames on the desktop. -fig4.4- indicates the functions available to the user while in a group.

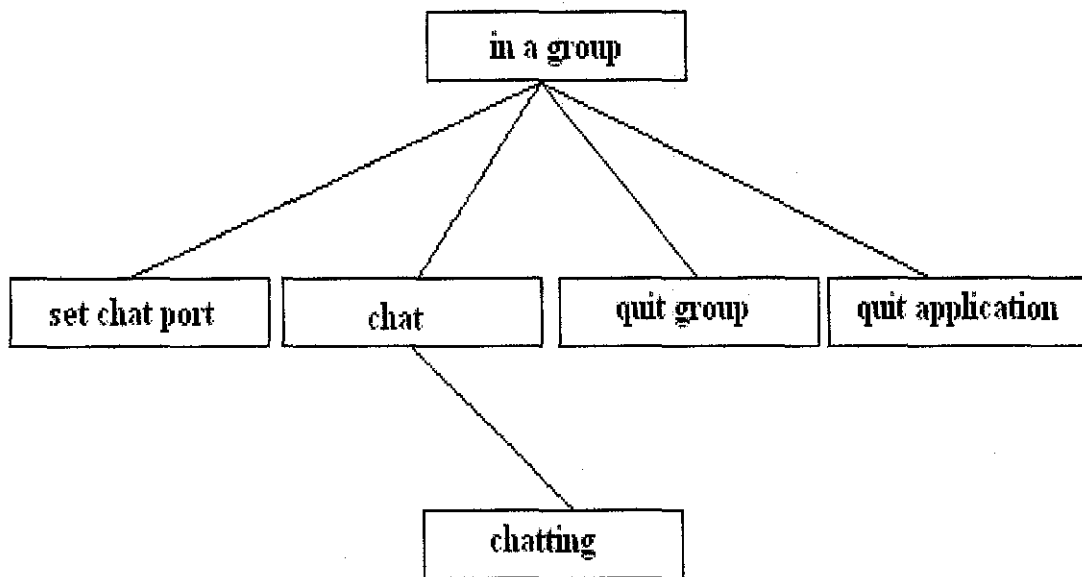


Figure 4.4

Each client in the group should appear on the desktop within a few seconds. Each client should be represented by a video screen and an audio control. All the video screens should cascade across the top of the screen and the audio controls should appear minimized on the bottom left hand side.

The video frames for each client should be similar to the monitor except they should be resizable. They should be movable, and resizable. As with the

monitor they should contain the streaming video itself and a Java Media Framework control panel at the bottom. This panel contains many useful functions like start and stop and lots of info about the data being received and presented. These are standard controls that are included by the JMF and are documented on the JMF sun website. The audio received from each client should be obviously heard through the speakers and is represented on screen by JMF audio control in a frame. Each member's media object should contain the member's name in the title bar together with a colorful icon.

C4.5 Chat [accessible through button panel and menu-bar]

Upon activating this function, again if the user has not selected a member from the member list, he/she will be prompted to do so. If a member from the member-list is selected the user will be asked to confirm the request to chat. If not confirmed (cancel) abandon, if ok, the application will send a request to chat to the other member. A dialog keeping the user informed of this process should be presented. The other member should spontaneously get a pop up dialog asking if he wants to chat with the requesting member. If the chat is agreeable both members get chat windows created on their desktops. If the chat is not agreeable the requesting member gets a message to this effect.

The chat window should operate as the common perception of chat functions go. Each user can type into their text area and the text is not committed to the common text-area until the 'say' button is clicked. When one member decides to quit he/she should simply click the quit button on the frame. The frame disappears and the other user gets a message saying the other user has quit. Upon clicking ok the chat frame disappears.

The chat frame on each client's desktop should appear on the bottom left corner and should be resizable, moveable and iconifiable. Also the client should be able to chat with any number of clients at once. –Figure 4.5- shows the functions available to the user while chatting.

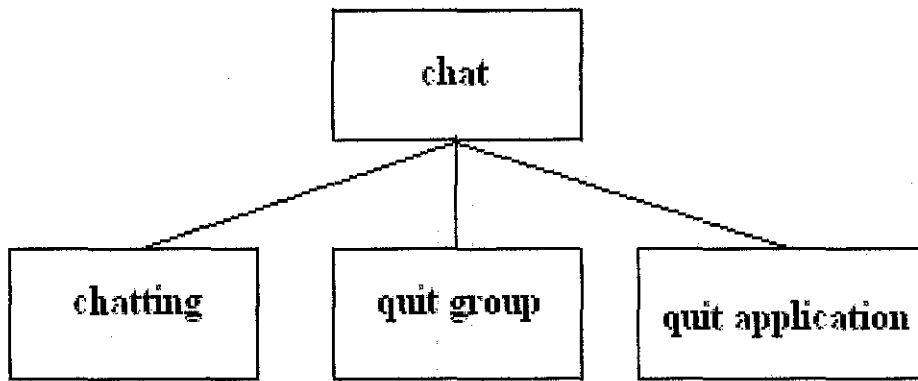


Figure 4.5

C4.6 Quit Group [accessible through button panel and menu-bar]

This function upon activated should send a message to the server telling of the quit desire (the server will tell all the other clients). The client should remove and stop receiving all the members' media and stop sending media to the other members. The client will then effectively go back to the capturing state where he can start or join a group again.

C4.7 Quit application [accessible through button panel and main frame]

This quit function should perform the quit group function (if the user is in a group), tell the server you're logging off and then exit the application.

C4.8 Server-IP setting [accessible through menu-bar]

This function should simply allow editing of the server-IP setting. When we begin the application we should be aware of where the server is situated and know its IP address and listening port. This function should present the user with an editable textfield containing the current default server IP address (set in code). The dialog should contain buttons to cancel or ok the change. If a change is made the new default server-IP will be set for the login process, if the change is made and the user cancels/abandons the operation no change is made to the default server-IP.

C4.9 Server-port setting [accessible through menu-bar]

The server port is the port that the server is permanently listening for new clients on. This function should operate exactly as the above set server-IP function operates.

C4.10 Local-port setting [accessible through menu-bar]

The local-port is the first port that the client sends its media out of. This function should operate exactly as the above set server-IP function operates.

C4.11 Chat-port setting [accessible through menu-bar]

The chat port is the port that the client listens for requests to chat on. The client is, like the server, listening permanently for approaches from other clients on this port. This function should operate exactly as the above set server-IP function operates.

C4.12 New member Joins group [not client activated]

When the user is a member of a group and another member joins, the server sends the new members name and IP address each client. The user will see a new member appearing in the members list. Each client then listens for and sends media to the new client. After a moment the new member's media should be rendered to the screen and the new member should be receiving the sent media. During this process the user should be kept informed of the goings on by message dialog.

C4.13 Member quits group

When a member quits, the server sends a message to each client informing them of this event. Each client then removes the member from the send and receives lists. Automatically this happens and the members' media should be removed from the screen and the name from the members list.

C5 Permissions

The current status of the client at any point in time should be noted on the title-bar. There are 4 statuses (5, when including the start when nothing is happening), Capturing, logged-In, in-a-group and chatting. When the client is in any of these states it should be indicated on the title-bar containing information about the particular state. The following are a description of the functions available to the user while in each of these particular states. These were all illustrated in the preceding figures included with the function descriptions

Begin: Upon starting the application the user should only be able to capture media, set the settings or quit the application. All the other functions should be greyed out and unavailable.

Capturing: When the application is capturing and the monitor is on the screen the only options available should be to login, change the settings or quit the application.

Logged-In: When the user is logged- in to the server the options available should be to start or join a group, to change the chat and initial transmission port settings, and to quit the application.

In-A-Group: When the user is in a group (started a group or joined a group) the only functions available should be to quit group or application, chat and the chat port setting. All other functions should be greyed out and unavailable.

Chatting: When the user is chatting to another client the only options available should be to quit group or application or to chat with another client.

4.1.2 Functional Specification for Server

S1 Main frame

The main frame of the server should be presented upon running the application. The frame should be resizable, moveable and iconifiable. There will not be much user interaction with the server GUI therefore the layout can be as simple as possible. The GUI should have a menu-bar, a main list for logging server events, a logged- in list, a groups list and a members list.

S2 Monitor list

The monitor list should be a large list into which constant logging of server events will go. Every time a client interacts with the server it should be logged here. If that user interaction results in the server taking action (e.g. last member of a group quitting ends the group, all other logged- in clients need to be told) the action should be logged here. This list should be constantly updating with client server action. The ability to clear this list should also be provided in the form of a button below the list.

S3 Lists panel

The server will contain 3 lists of information about the state of the system. A logged in list, a groups list and a members list, similar to the clients' one, will be provided. The logged- in list will contain a real time list of all logged-in members. The groups list will list all current groups. Upon clicking on a group in this list the members of the group will appear in the members list. Otherwise the members list will contain a list of the members of the most recently created group. These lists are to be constantly updating in response to client actions and with each update the corresponding action should be logged- in the monitor list.

S4 Menu-bar Functions

The functions of the server are to be mainly automatic. The menu-bar should contain the following headers. Server, settings and quit. The settings option should contain a server-IP and server-port setting function. The server option should only contain one function

'begin listening' and the quit option should only contain one function which is quit application.

S4.1 Server - Begin Listening

Upon starting the application the server will not be 'active' until this 'begin listening' function is activated. This will open up the channel for contact from clients. The server will constantly be listening for contact until the application is quit.

S4.2 Settings- set server-IP and set server-port:

The server-IP is the IP address of the machine the server is currently running on. The server Port is the port to be listening for client communications on. These settings functions should perform exactly the same as the above settings functions in the client.

S4.3 Quit

The user should be able to quit the application from the menu-bar or from the top right hand corner frame button. Either way, the server will stop listening to the server port and the application will be stopped.

S5 Permissions

Once the server has 'began listening' the server-IP and Port cannot be changed and should be greyed out. –Figure 4.6-

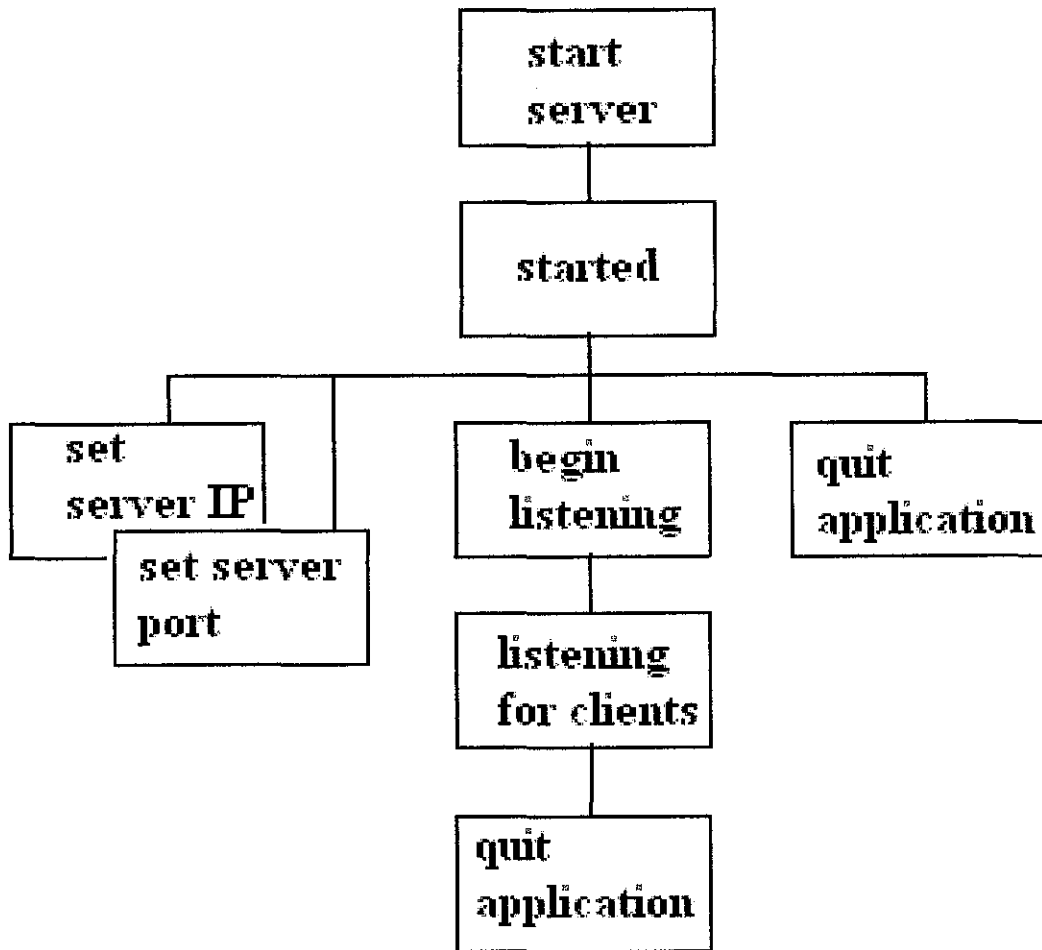
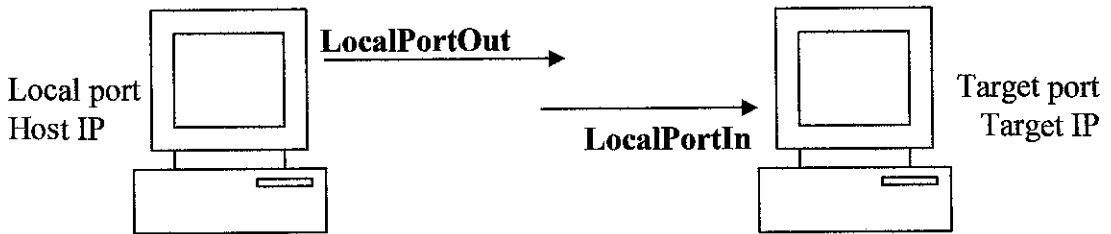


Figure 4.6

4.2 Discussion on Port Control Issues

The administration of ports proved to be one of the tricky areas of the design. As illustrated by the model below –Figure 4.7-, a simple transmission from one machine to another, for each media stream, requires the use of a sending (or local port) and a receiving (or target port) at the other end. On the sending side the JMF uses a session-address to create the Real-Time Transport Protocol (RTP) session. The session-address consists of the host IP address and the port out of which the media will stream. Each target is added to the session using a target-address. The target address is the IP address of the client to receive the media and the port the client application is going to take the media in on. On the receiving end the RTP session listens to targets described by the IP

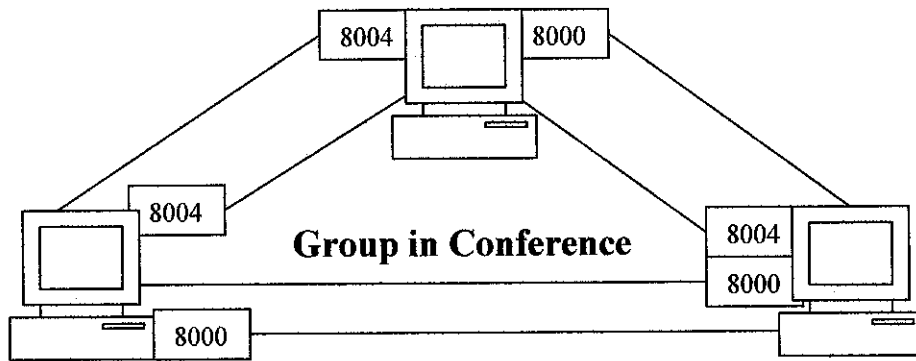
and the local port to use in accepting the media stream. This is the same port that the sender is sending to. It is essential that the port that the receiver receives on is known by both clients so the sender knows what port the receiver is listening for the stream on. If a stream from a different IP address turns up at the port designated for another IP the stream will not be received.



- **Transmit** requires home and away address
- Session address: LocalPortOut, Host IP
- Target address: Target port, Target IP
- **Receive** requires away IP and home port
- Host IP, LocalPortIn

Figure 4.7

Consider a scenario where we have group in conference with, for example 3 members -Figure 4.8-, where each member is streaming its video and audio to both the other clients. This client is also receiving video and audio from both the other members. All the clients are sending their media out a default local-port (9000 for video and 9002 for audio) to each other client. Each client is receiving 4 streams from the other group members. As each member joined the server sent the new member and the existing members IP addresses and ports to use in the streaming. This process is described next as a new member wants to join the group.



- Use 4 ports for each member
- What if new member joins group, what ports should new member listen on and send to?
- What ports should the existing member send to and listen?

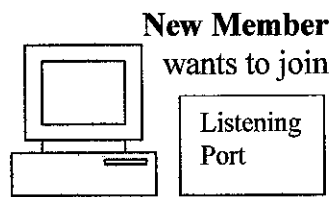


Figure 4.8

New group member

The new group member needs to know the targets to send the media to and the targets to receive from.

New group member – Transmit

The new member needs to know the addresses of the other members of the group to send his media to. Each client maintains a default starting port which is used to receive the first stream through. It is set as 8000 by default but can be changed in the settings menu if desired.

For each group the server maintains a variable describing the number of members in the group (NIG). When a new member joins this number is incremented. But when a member leaves it is not decremented (the reason for this will follow).

When the new member joins the group the server sends a list of the existing members, their IP addresses and this NIG variable. For each member the client simply sends the video to the members IP address and starting Port+

($NIG-2*4$) in this instance. $8000 + (4-2*4) = 8008$ and the audio to this port $+2=8010$. Each client gets the new members media on port 8008 and 8010. The reason for the gap of 2 between ports is that at the JMF uses the next port for each stream to send RTCP (real time control protocol) streams on. So in this instance the sending of media involves 4 ports, 2 media and 2 controls.

New group member - Receive

The new member needs to receive the 3 other members media. He already knows their IP addresses but he needs to know what local ports to listen for their media on. The new member simply starts at the default local port and goes through the list sent from the server sequentially and listens for each member on local port adding 4 each time. So he listens on;

- 8000/8002 from the first member in the list
- 8004/8006 from the second member in the list
- 8008/8010 from the third member in the list

Each existing member

Each existing member needs to know the IP address of the new member and the ports to send and receive on.

Each existing member - Transmit

When the server sends a list of existing members to the new member it compiles the list from its own members list. The server must also send the new members IP address to each existing member and uses this members list to do so. It goes through this list one by one and sends the new IP address to each member, but it also adds a sequence number SEQUENCE, to the message sent to each existing member. So the first member will get the new members IP and sequence number 1, the second gets IP and 2, and so on. Each member sends its video to the new members IP address into port local port $8000 + SEQUENCE * 4$ and the audio to this $+2$.

- The first member transmits to port $8000+0*4=8000/8002$
- The second member transmits to port $8000+1*4=8004/8006$
- The third member transmits to port $8000+2*4=8008/8010$

This procedure ensures that the new member listens to the same port as the existing member is sending to.

Existing member - Receive

In addition to the new members IP address and the SEQUENCE number, the existing members also receive the NIG (number in group) variable from the server. This lets them know, in the same way the new member knows, which port to listen on for the new members media $8000+(NIG-2*4)=8008$.

The result of all this is we have a 4 man group each sending and receiving each others video, audio and control streams. – Figure 4.9-

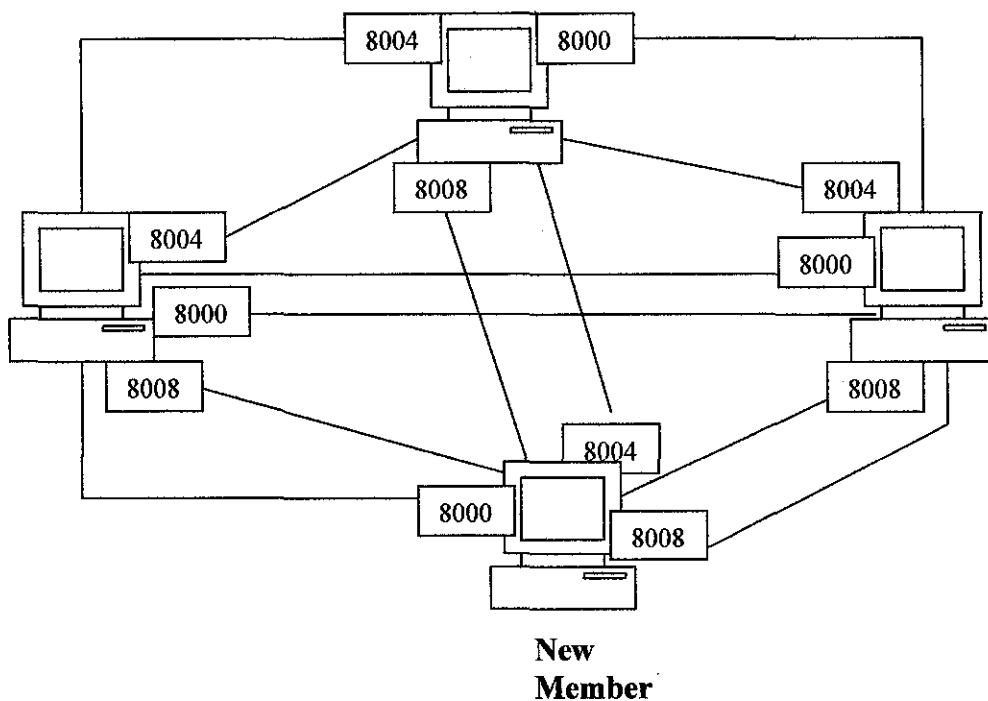


Figure 4.9

Member Leaving

When a member leaves the group the ports used by each remaining group member and the leaving member are freed up. If the leaving member (or any new member) joins the same group again, the whole process above, in deciding ports, is carried out again. The new member will end up sending and receiving again using the same ports as previous on his own machine. But on the existing members machines the ports used previously (by the old member) are not used again, instead the next available ports are used. This is indicated by using the NIG variable held by the server in the formula described earlier. To ensure that the next available ports on the existing members' machines are used, the NIG variable for a group (maintained in the server for each group) is not decremented when a member leaves. So for the scenario above if the new member left and joined the group again the NIG variable for the group would be 5 even though there would be only 4 in the group. The new member would then send to port $8000 + (NIG - 2 * 4) = 8012/8014$ on the existing members machines, and the existing members would listen for the new member on $8000 + (NIG - 2 * 4) = 8012/8014$ thereby ensuring a fully functioning conference each time.

CHAPTER 5

CONCLUSION AND RECOMMENDATION

5. CONCLUSION AND RECOMMENDATION

This chapter highlights the most significant things in relation the objectives of this project. Later in this section, also included the recommendations for future project works.

5.1 Conclusion

As a conclusion, I can say that this project will be challenging for me. Major considerations have to be given in the development and evaluation phases. These two faces give a lot of challenges in my previous projects. But the most important thing is that many new experiences were gained during the whole process of this project.

This project goal was to implement a conference system that not only met the current needs and requirements, but would also provide response to the evolving technologies. Education and learning may benefits a lot from this system so that it will help lecturers and students improve their communications ability. Many costs may incur but the advantages gained by implementing this project may worth it.

The research on this project has been done but sincerely, I still need some guidelines from the experts. Many areas can be improved in order to complete the project goals and objectives. By the end of this project, the application should be able to:

- Initiate and terminate the sending and receiving of live streaming video and audio data. This data to be displayed in a GUI application for manipulation by the user and interaction with the video conference server.
- Manage the initiation, monitoring and termination of the sending and receiving of live streaming video and audio data to and from several clients. With a GUI to help manage these tasks and to be the users interface to the server

5.2 Recommendation

Detailed research can be done on topics of this project to enhance the system. Telecommunications technologies evolve rapidly and hope that we are not left behind. As the application can be enhanced, the same goes to the user interface design and the functionality of the application. Here are some potential further works for future project work.

1) **Performance testing over ISDN and ADSL on hi spec machines using many different makes of web-cam**

The performance testing done so far has only really scratched the surface.

With the available hardware a lot more testing could be done on different spec machines. It would prove very interesting to observe the performance of the application on very hi spec desktop machines over hi bandwidth networks.

Also some testing on a variety of web-cams and microphones could yield some interesting results.

2) **Performance testing running on many different operating systems**

The fact that the application was written in pure java leads me to believe that is compatible with non windows operating systems. Some further testing on different operating systems (including non-windows) could be interesting.

3) **Audio capture and encoding options for the user**

I have really concentrated on the video media in this project (due to the critical effect it has on bandwidth usage), I would like to consider some further enhancements in the audio area. i.e. to allow the user choose audio capture and encoding formats.

4) **Auto arrange button on the desktop**

A button to arrange everything on the desktop (like client media frames, chat frames and the monitor) would be useful.

5) **Document sharing, Shared whiteboard**

Commonly among video conferencing applications are document sharing and shared whiteboard functions. In such applications these functions can prove to be the primary functions and the video and audio only really a support for the collaboration. This would be a major enhancement and I am not sure it would be feasible using java; nevertheless it would provide some interesting research.

6) **Investigate the potential of using custom or hardware codec's with the JMF**

The JMF provides a plug-in architecture that could be used to implement custom codec's. Because of the high CPU usage the compression/decompression process requires, some research into whether this could be done more efficiently, or even in hardware might be useful.

7) **More research into how professional video conferencing applications are put together**

Although I have researched the structure of the main video professional conferencing applications, I would like to learn more about the software developed to run these systems.

8) **Add a client timeout check to the server**

The server periodically checks to see if all logged-in clients are awake. If the client does not respond to the call, it is presumed terminated and will be removed from the list. This allows for the scenario where a client machine crashes or the application is terminated in some other way other than the quit button. The server will always be aware who is actively logged- in.

9) **Add a security element to the application**

Each group could have a password to join. Anyone not knowing the password cannot join. Maybe maintain a list of registered users that the server checks when someone log's in. At the moment the user is free to do as he/she pleases.

REFERENCES

1. <http://www.vide.gatech.edu/cookbook2.0> section on 'Uses of Video Conferencing'
2. <http://www.kodak.com/US/en/digital/dlc/book1/chapter6/11.shtml> Chapter VI, Desktop Video Conferencing: Lesson I, Introduction To Desktop Video Conferencing
3. <http://www.vide.gatech.edu/cookbook2.0/advanced.html> "Advanced Video Conferencing Functionality and Management"
4. Theo Mandel, 1997, Element of User Interface Design, United States of America, John Wiley & Sons Inc.
5. Gareth O.Baker, James R. Albaugh, Definitions of interactive learning: "What we see is not what we get", Journal of Instructional Delivery Systems, 1993.
6. Alan Dix, Janet Finlay, Gregory Abowd, Russell Beale, Human Computer Interaction.
7. Japan's DoCoMo develops world's first 3G mobile phone video conference, Agence France-Presse - March 13, 2002.
8. Principles of Digital Audio', K. Pohlmann, Howard W. Sams & Co., 1985.
9. 'Digital Audio Compression', D. Pan, Digital Technical Journal, Vol. 5 No. 2.
10. http://www.inforamp.net/~poynton/notes/colour_and_gamma/ColorFQ.html.
11. <http://www.siggraph.org/education/materials/HyperGraph/video/codecs>
12. 'Video Compression', Peter Symes, McGraw-Hill, April 1998

APPENDICES

6.1 TECHNICAL SPECIFICATION

The technical specification below describes the architecture for the application and class descriptions for each class. There are 23 classes to the client and 8 for the server altogether containing 5868 lines of code.

6.1.1 System Architecture

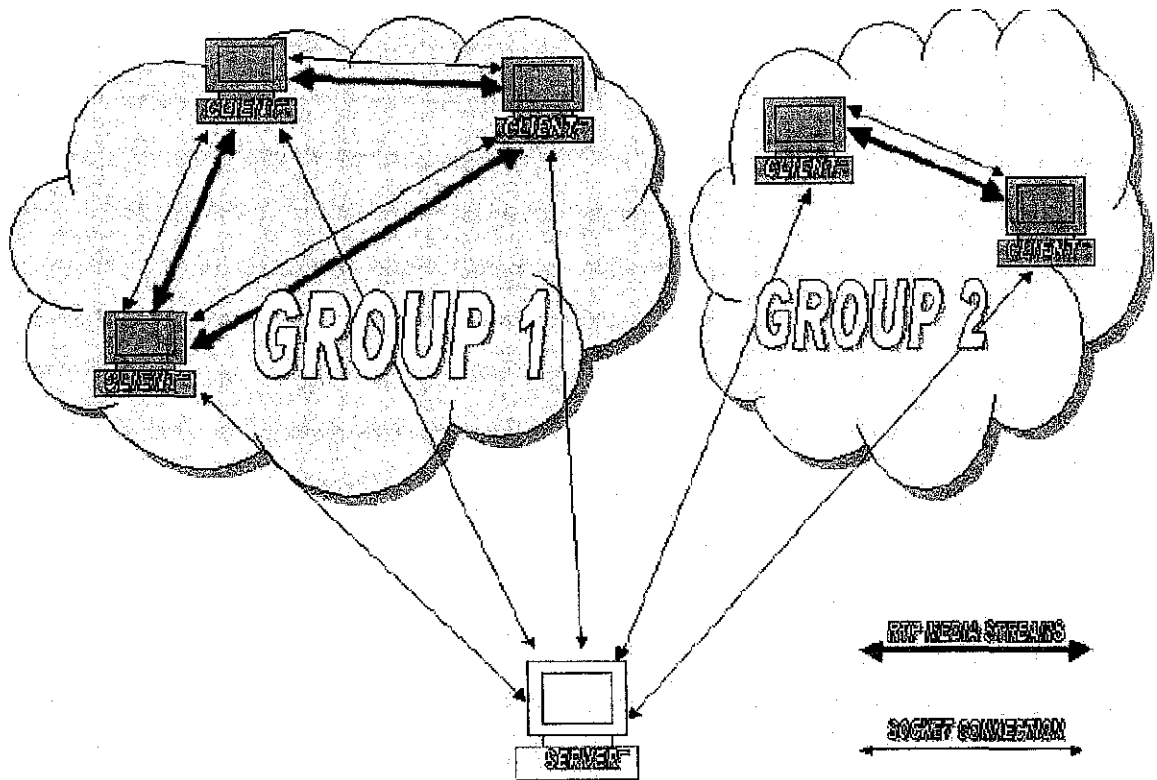
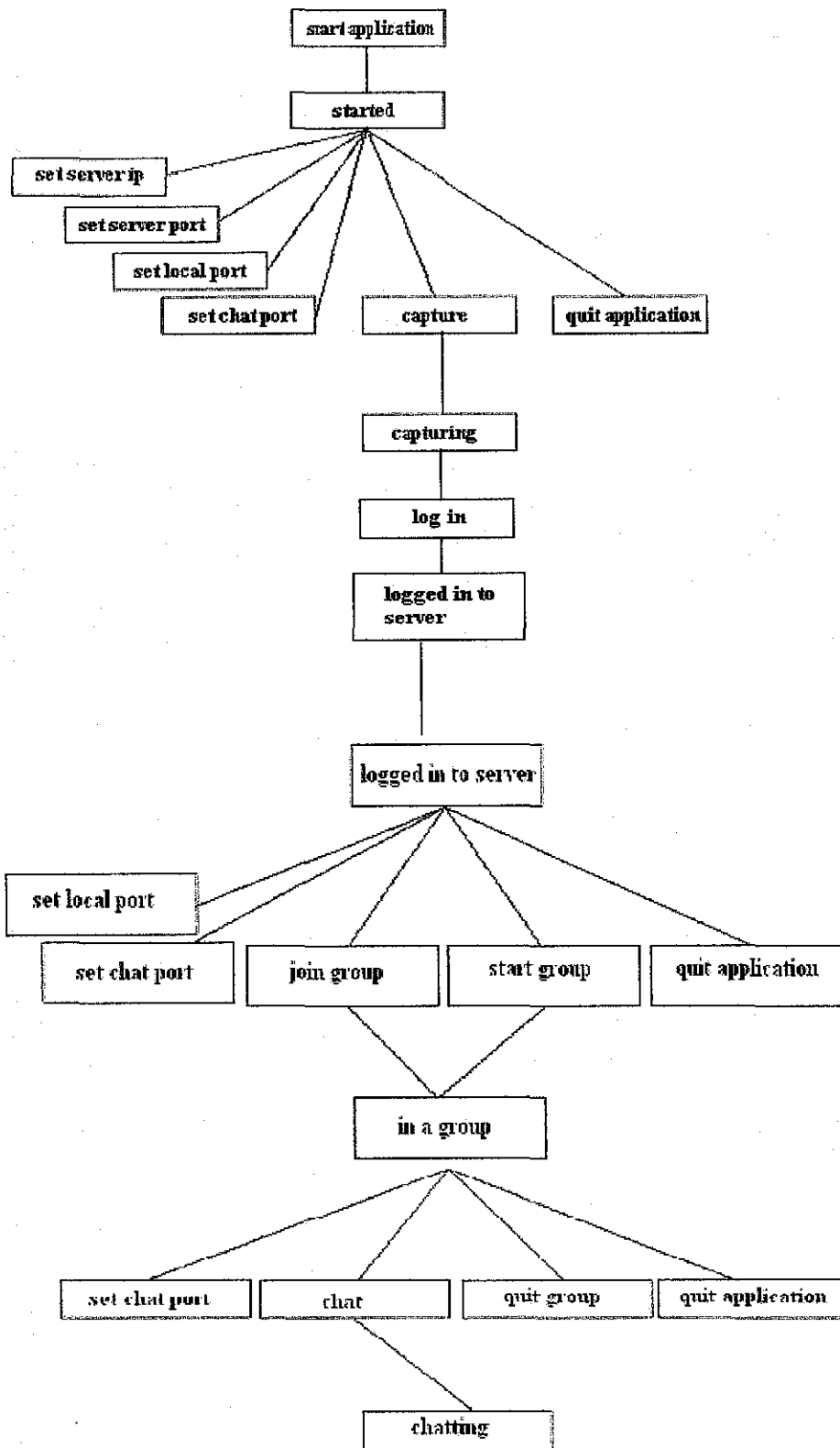


Figure 6.1

This application uses a peer to peer system, where each client streams the media to the other clients, and the server monitors and controls the client and group administration (acting as a decentralized MCU in control mode only, see section 2.1.2 for more on MCU's) . The server would not have the power or the bandwidth available to accept, combine and serve multiple streams. The server here –Figure 6.1- basically holds the entire client and group states, and communicates the changes in these states to the clients. (E.g. Group member leaving, joining).

6.1.2 Application Functional Architecture



6.1.3 Class Description (Server)

1. ConfServer2 [JFrame] main routine 334 lines

- creates GUI, logged-in panel, members panel and groups panel
- creates the commsViewer panel
- creates the listener socketConn
- provides a menu-bar for user interaction
- ends with application

2. LogPanServ [JPanel] manages logged-in panel object 131 lines

- creates the list and list-model
- provides functions to add delete and query the list-model
- ends with application

3. MemPanServ [JPanel] manages member's panel object 146 lines

- creates the list and list-model
- provides functions to add delete and query the list-model
- ends with application

4. GroPanServ [JPanel] manages groups' panel object 135 lines

- creates the list and list-model
- provides functions to add delete and query the list-model
- ends with application

5. socketConn [thread] accepts connections from clients 206 lines

- created by main when user wants to begin listening
- creates and listens to a serverSocket
- when a client log's- in it creates a clientConn to manage communications
- it also provides lots of functions allowing sending of messages to all clients
- ends with application

6. clientConn [thread] handles individual client communications 237 lines

- accepts all messages from the client
- processes the messages by code
- responds by updating the lists, communications viewer and the other clients
- ends when client quits

7. commsViewer [JPanel] list of all communications with clients 261 lines.

- creates the list and GUI
- provides functions to clear list
- provides function to write to list
- ends with application

8. group [utility] group object 11 lines

- created in socketConn upon new group creation
- group object holds info about the group
- ends when group ends

6.1.4 Class Description (Client)

In addition to the main VCClient class, the client's classes can be separated into 6 groups each performing a chunk of the clients work at runtime. The areas of the client are as follows:

Main

1. VCClient [frame] main routine 876 lines

- main routine started from command line
- sets up and manages the GUI
- provides a central manager for all the application functions
- the menu-bar and button-panel classes respond by activating routines here
- these routines perform the core functions of the application login, start, join, chat and quit groups
- ends when the user quits

A-GUI classes

2. Menubar [JPanel] processes menu functions 272 lines

- Panel created as part of GUI by VCClient
- processes all menu user interaction by activating routines in VCClient
- ends with main frame

3. buttonPanel [JPanel] processes button functions 241 lines

- Panel created as part of GUI by VCClient
- processes all menu user interaction by activating routines in VCClient
- ends with main frame

4. groupsPanel [JPanel] manages the groups list 169 lines

- panel created by VCClient to hold contents of groups list
- processes user interaction with the list
- ends with the main frame

5. membersPanel [JPanel] manages the groups list 198 lines

- panel created by VCClient to hold contents of groups list
- processes user interaction with the list
- ends with the main frame

6. waiting [JDialog] waiting dialog 30 lines

- created during main routine when waiting
- provides a dialog to waiting client]
- ends when finished

B- Server connection classes

7. connection [utility] creates connection to server 57 lines

- created at login
- creates socket connection with the server allowing communications to the server
- creates listener thread to process messages from the server
- ends with user quitting application

8. fromServer [thread] listens to the server 95 lines

- created with connection to server
- processes messages from the server by calling methods in VCCClient
- ends with server or application ending

C - Media capture, monitor and transmission classes

9. monitorThread [thread] creates the monitor 55 lines

- created by user/VCCClient
- this is a thread to stop system losing focus during capture
- creates the monitor
- handles hide/unhide functions
- ends when monitor finishes

10. Monitor [JInternalFrame] creates processor and monitorStream 339 lines

- created by monitor thread when user requests
- uses captureUtil to create custom datasource monitorCDS
- creates a processor from the datasource
- configures the processor for output, encoding of each track
- realizes the processor and makes it available for using by other objects
- it extracts a monitor from the monitorCDS, custom datasource object
- gets the visual and control components and adds them to the frame
- ends when user wants to quit

11. captureUtil [utility] creates the datasource 164 lines

- class contains function getCaptureDS
- it queries the captureDeviceManager and picks the first for audio and video
- it creates a datasource with the mediaLocator of the capture device
- it queries the capture device for supported capture formats
- gets the user choice for capture format
- gets the format control from the capture device and sets the user format choice
- it then creates a custom monitor datasource using the video datasource object
- we then create a merged datasource and return it to the monitor object
- this is a utility class used only to create the datasource

12. monitorCDS [pushBufferDatasource] custom datasource 99 lines

- created by captureUtil
- custom datasource allowing monitoring of the video stream
- ends with processor

13. monitorStream [pushBufferStream] allows monitoring of the stream 178 lines

- created by monitorCDS
- this object is a stream with a monitor built in
- takes the media from the stream and renders it presentable in the monitor
- ends with the processor

14. Transmitter [utility] creates RTP session 220 lines

- created at login from VCCClient
- creates an object that can RTP stream to the clients
- gets the datasource from the monitor object
- creates RTP session
- creates a ssrcConn object each time a stream is to be sent, sends the SSRC info
- adds/removes targets to the RTP session
- ends with the processor

D - Media Reception classes

15. AVReceiver [utility] receive stream listener 650 lines

- created when group joined or member joins our group
- allows adding or removing of targets
- upon receiving of stream it creates a new player with the stream
- A playerWindow [JInternalFrame] is then created with the player
- 18.1. playerWindow [JInternalFrame] plays the stream
- gets the SSRC from the stream
- searches the SSRC vector for the corresponding stream
- adds the name and type to the object
- gets the video and control components from the realized player
- adds the correctly sized video and control components to the frame
- it then adds itself to the desktop
- this object also contains a function to remove invalid playerWindows when target removed
- ends with user ending application

16. Target [utility] target object for receiver 15lines

- created when initializing the receiver with the first target

E - Chat function classes

17. socketConn [thread] listen for chat, creates clientConn 74l ines

- Thread started from VCClient.
- Server socket continually listening on 5555
- upon receiving approach to chat from other client
- it binds the connection to a new socket on a new port
- and creates a clientConn object with the new socket for the two clients to communicate
- this stays listening while the application is running

18. clientConn [thread] process socket communications with client 181 lines

- started from socketConn
- listens to socket, processes chat messages and provides function to write to socket
- finishes when chat finished

19. Chat [JInternalFrame] this is the chat window 267 lines

- created by the clientConn
- created if we get a message back that the other client agrees to a chat
- or if we agree to a chat from another user
- this object process user input to the frame by sending message to the clientConn
- this also allows incoming other user chat messages to be displayed in the frame
- object dies when chat finished

20. chatConnTo [utility] creates connection to other client 62 lines

- created by VCClient by user
- creates clientConn to handle any further communications
- provides facility to ask client for a chat
- ends with chat

F- SSRC handling classes

21. ssrcConnListen [thread] listen for SSRC messages 73 lines

- thread started from VCClient
- listens for incoming SSRC messages
- sends any new messages to VCClient
- stays listening while the application is running

22. SSRC [utility] holds info about stream 37 lines

- created by VCClient anytime a new SSRC arrives
- holds name SSRC and type of stream to expect
- object held in vector and goes away when removed from vector

23. ssrcConn [utility] sends SSRC before stream 55 lines

- created by transmitter to send SSRC info to client before stream
- ends after message is sent