Secure Video Conferencing for Web Based Security Surveillance System

A Thesis Submitted
in Partial Fulfillment of the Requirements
for the Degree of
Master of Technology

by

Gurmeet Singh

to the

Department of Computer Science and Engineering
Indian Institute of Technology, Kanpur

July, 2006

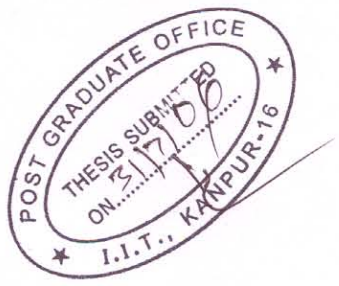
Certificate

It is certified that the work contained in the thesis entitled "*Secure Video Conferencing for Web Based Security Surveillance System* " by *Gurmeet Singh* has been carried out under our supervision and that this work has not been submitted elsewhere for a degree.

July, 2006

(Dr. T. V. Prabhakar)
Department of Computer Science and Engineering,
Indian Institute of Technology, Kanpur

(Mr. A. G. Apte)
Head, Computer Division, BARC, Mumbai



Abstract

We have developed a secure video conferencing system that allows the multiple users to exchange their real time audio and video streams in a secure manner. It is a client server based application implemented using Java Media Framework (JMF) API. The server manages the connection between the clients and provides session key for encryption/decryption of the audio/video streams. At the time of joining the conference by a user who is already registered at the server, the authentication is done using RSA public-private key pairs. For each new conferencing session server generates a fresh secrete key (Triple-DES) which is transmitted to the client after authentication in a secure manner.

The video streams are compressed in H.263 format using the software codec. The audio data is compressed using G.723 format. After compression, both audio and video streams are encrypted using Triple-DES algorithm with 168 bit key and then transmitted over the network using RTP over UDP.

The system also provides QoS features. The server gets the feedback reports from the receivers about the quality of the audio/video streams. The feedback reports include the information like frame loss and delay jitter. From the feed backs server makes the QoS decisions and sends the control signals to the transmitters to change the quality of their audio/video streams.

We are doing all the processing like compression, decompression, encryption, decryption in software so no extra hardware is required. The system allows up to 12 active participants to take part in the conference using P-IV, 2.4GHz processor with 512 MB of RAM.

Acknowledgements

I am extremely grateful to Prof. T. V. Prabhakar, my project guide from IIT Kanpur for his guidance and support during my thesis. I feel fortunate for doing my thesis work under him. I am most thankful to him for reviewing my dissertation closely, pointing out the mistakes, and for the suggestions that he gave.

I am also grateful to Mr. A. G. Apte, my project guide from BARC Mumbai and Mr. S. K. Parulkar, from BARC for their invaluable guidance and endless encouragements throughout the period of my thesis work. I feel highly privileged to have been given a chance to work under them.

I would also like to thank Mr. Bhagwan N Bathe and Miss. Renuka Ghate, from BARC for their help by providing useful information and advice right from the beginning, and for reviewing and testing my code.

Finally, I would like to thank my parents for taking me to this stage in life, it was their blessings which always gave me courage to face challenges and made my path easier.

Contents

List of Tables

List of Figures

Chapter – 1

Introduction

1.1 Motivation

Video conferencing is now certainly growing very rapidly. It saves significant amount of money in terms of both travel and time. During the last decade availability of the equipments and internet connections at the reasonable cost make it affordable to a much wider group of users. It is now widely used in the fields of education, collaborative work between researchers and business communities, telemedicine and in many other fields.

On the other hand recent terrorist activities have forced many organizations and government agencies to re-examine their existing security policies and mechanisms. The government agencies must now carefully safeguard their sensitive data transmission, for example transmission of voice, video and other data during videoconference meetings.

Bhabha Atomic Research Centre (BARC) is developing a security surveillance system for monitoring various security activities inside BARC. We are developing a secure video conferencing system to be used along with the security surveillance system so that various security posts inside BARC can use existing intranet infrastructure for audio and video communication. The main objective is to provide centralized control over the users and to provide smooth quality of service.

The secure video conferencing tool is proposed to be used for BARC only but depends on the success of the tool it can be used up to DAE level or for the commercial purpose also.

1.2 Project Description

The objective is to develop the video conferencing system with security features incorporated in it as part of the web based security surveillance system being developed at BARC. The system include a centralized server and distributed clients. The development is comprised of two parts, namely, server side application and client side application.

Server side application

Following features are proposed to be incorporated in the server side application.

- Initialization and authentication of security Nodes: Server should take care of authentication of any user before he joins the conference and provide user required information for initialization.
- Key management required for encryption: Server has to create the session key when user joins the conference and it has also to manage the session keys for subsequent communication. The server is also responsible for the distribution of the public and private keys to all users.
- User management: Server is responsible for forming various user groups, deciding the policy for joining the group, assigning the various levels of permission to the user.
- Selection and configuration of communication protocol between nodes: Server is responsible for smooth communication between all users. Server will decide what is the communication protocol to be used, how to manage the various connection, how to forward the data from one user to another user.

Client side application

Following features are proposed to be incorporated in the client side application.

- Capturing and formatting of the audio video data: Client application is responsible for capturing of audio and video data, compressing data, encrypting /decrypting then encapsulating the data in a format suitable for transmission over the network.
- Creation of RTP sessions to transmit the audio/video data over the network.
- Reception of audio/video streams of the remote participants.

1.3 Outline of Thesis

Thesis is organized in the following way: -

Chapter-2 presents an overview of the video conferencing system and its applications. Related technologies and popular standards are discussed.

Chapter-3 presents the overview of the internet security. The encryption algorithms that we use in our application are discussed. Vulnerabilities in IP are also discussed.

In Chapter-4 the Java Media Framework (JMF) API is introduced using which we have developed over application.

Chapter-5 describes the design and implementation of the system.

Chapter-6 presents the testing results. The scope of future improvements is also discussed.

Chapter – 2

Introduction to Videoconferencing

2.1 Introduction

Video conferencing can be described as a method of conferencing between people at two or more physically separated locations where the transmission of synchronized audio and video is done through the telecommunication networks.

Video conferencing has been with us for decades. For example, multinational corporations and television broadcasters have been using it for decades but they used dedicated connections between the sites which are very expensive. During the last decade availability of the internet connections at the reasonable cost make it affordable to a much wider group of users. Video conferencing is now certainly growing very rapidly. It saves significant amount of money in terms of both travel and time.

2.2 Applications of video conferencing

Video conferencing is implemented in a wide range of applications. Its implementation is most common in the field of education. Video conferencing provides students with the opportunity to learn by participating in a two-way communication process. Another common application of video conferencing is in telemedicine and telenursing applications such as diagnosis, consulting, transmission of medical images etc. It is also widely used by the business community as it substitutes the actual physical presence of the remote participants thereby reducing the travel costs and time.

The judicial system has also found it very useful. A number of countries have begun to install videoconference systems in jails and courthouses. It reduces the security risks associated with transporting and handling defendants. [2]

Video conferencing can also be used one-way monitoring technology to continuously monitoring the remote sites. Cameras are available today with the features like auto-tracking, auto-signaling. With the use of a cheap wireless video transmission system, the cameras can be placed where network cables don't reach. [2]

2.3 Video Conferencing Equipments

The basic conference requires a camera to capture images of the participants, microphone to pick up their speech, codecs to compress the audio/video streams and off course speakers and display device to play the audio/video streams of the remote participants.

Cameras

The camera converts light images into an electrical signal so that it may be displayed, recorded or transmitted. The camera lens focuses the light from the image onto a charge coupled device or CCD. Webcam is a very low cost single chip with a fixed focus lens camera. It can be used for a desktop conference.

Codecs

The Codecs ("Coder/Decoder") are hardware devices specially designed to offload the compression and decompression task from the PC allowing the endpoint overall to achieve good performance. However recent developments in processor technology make it possible to compress and decompress multimedia streams on general purpose processor such as Pentium family in real time.

2.4 Audio Compression

"When there is a trade-off between audio and video quality, good quality audio is usually preferred. Since bad audio puts a much bigger strain on the listener than bad video."[3].

To choose the appropriate audio codec under given circumstances, it is useful to consider some of the important characteristics of audio codecs.

Bandwidth

Audio bandwidth is expressed in hertz (Hz) and is a measure of the frequency that can be represented. To obtain a satisfactory reconstruction of original signal, the sampling rate must be at least twice the audio bandwidth. Speech bandwidth (200 – 3400 Hz) is usually sampled at 8000Hz.

Bit rate

The bit rate is simply the number of bit per second required to transmit the coded signal. For example an uncompressed audio signal with sampling rate 8000 Hz and sample size 16 bit, would require bit rate of 128 kbps.

Delay

The delay of the audio is the some of time require to buffer the frame and look ahead data (algorithmic delay), time requires to compress the frame, time requires to transmit the frame and time requires to decompress the frame. For ease of communication the total delay should be as small as possible. If there are echoes, then even small delay (100ms) can't be tolerated.

Complexity

Complexity of the coders is described as the number of instructions required to encode the signal and is usually measured in MIPS (millions of instruction per second). If we wants to use a software coder then this attribute should also be kept it minds.

Some of the most commonly used audio CODECs for videoconferencing are listed in Table-2.1[3]. All the CODECs listed in Table-2.1 provide an audio bandwidth 3 KHz, which is equivalent to phone quality speech except G.722. All the CODECs are supported by the H.323 standard.

CODEC	Bandwidth	Bit Rate	Typical Delay
G.711	3 KHz	48, 56, 64 kbps	< 1 ms
G.722	7 KHz	48, 56, 64 kbps	< 2 ms
G.723.1	3 KHz	5.3, 6.4 kbps	< 100 ms
G.728	3 KHz	16 kbps	< 2 ms
G.729	3 KHz	8 kbps	< 50 ms

Table – 2.1 audio CODECs for videoconferencing

G.723.1 audio coding [11]

ITU-T G.723.1 coder is based on the principles of linear prediction analysis-by-synthesis coding. The encoder operates on frames of 240 samples each. That is equal to 30 ms(milliseconds) at an 8 kHz sampling rate. There is an additional look ahead of 7.5 ms, resulting in a total algorithmic delay of 37.5 ms. Additional delays are due to processing delays of the implementation, transmission delays in the communication link etc.

This coder has two-bit rates associated with it, 5.3 and 6.3 kbps. Both rates are a mandatory part of the encoder and decoder. It is possible to switch between the two rates at any frame boundary. This coder was optimized to represent speech with a high quality at the above rates using a limited amount of complexity. So G.723.1 is a suitable coding for video conferencing application.

2.5 Video Compression [13] [14]

There are two kinds of redundancies present in video data. The first kind of redundancies is spatial, while the second kind is temporal. Spatial redundancy refers to the correlation present between different parts of a frame. Spatial redundancy is removed through the use of transform coding techniques. Temporal redundancies, on the other hand are the redundancies present between frames. Temporal redundancy is removed through the use of motion estimation and compensation techniques.

For videoconferencing application there are three commonly used video coding standards H.261, H.262 and H.263. H.262 is only used in the H.310 standard for broadband audiovisual communication. H.261 is the oldest standard and it is limited to using CIF (352X288) and QCIF (176X144) resolution video. H.263 was developed as an extension to H.261. H.263 is specifically designed for low bit rate

H.263 video coding standard

The H.263 standard specifies the requirements for a video encoder and decoder. It does not describe the encoder and decoder themselves, instead, it specify the format and content of the encoded stream. H.263 uses block based coding scheme in which the picture is sub-divided into smaller units called blocks that are processed one by one. A block is defined as a set of 8x8 pixels. As the chrominance components are down

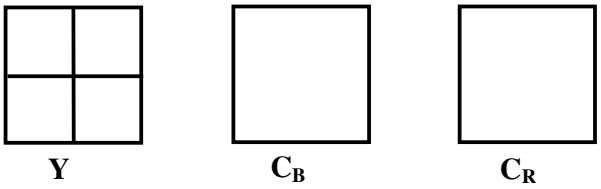


Figure - 2.1 Blocks in a Macroblock

sampled, each chrominance block corresponds to four Y blocks. The collection of these 6 blocks is called a macroblock (MB). A MB is treated as a unit during the coding process.

H.263 Encoder

Figure 2.2 describes the typical H.263 encoder. The motion estimation module compares each macroblock in the current frame with its surrounding area in the previous frame and attempts to find a match. The motion compensation module moves the matching area in previous frame into macroblock position.

Motion compensated macroblock of the previous frame is then subtracted from the macroblock of current frame. The remaining macroblock is called “residual” macroblock. The DCT (Discrete cosine transform) transforms the residual macroblock into frequency domain. It compact the energy in block of values into a small number of coefficients.

The Quantizer module reduces the precision of coefficients by dividing each coefficient by an integer scale factor and truncating the result. An entropy encoder (such as Huffman encoder [15]) replaces frequently occurring values with short binary codes and infrequently occurring values with longer binary codes. It is used to compress the quantized DCT coefficients.

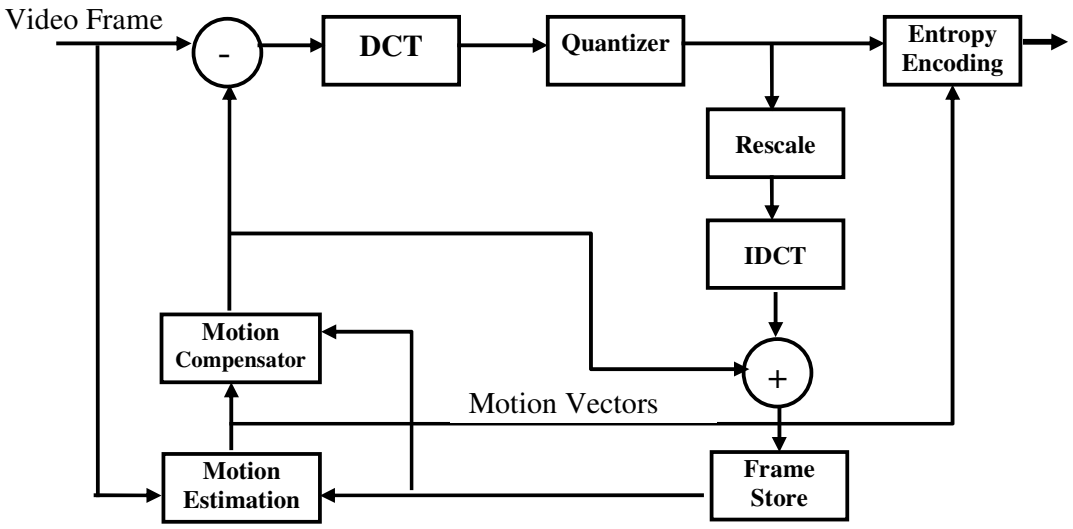


Figure-2.2 H.263 Encoder

The frame store module stores the current frame to be used as a reference for next frame. Since DCT and Quantizer are lossy transforms the reference frame is reconstructed from the quantized coefficients. This ensures that the contents of the frame store in the encoder are identical to the contents of the frame store in the decoder.

H.263 Decoder

Figure 2.3 describe the decoder. It does the reverse of the encoder. The entropy decode module extract the motion vectors and quantized coefficients from the variable length codes. Quantized coefficients then rescaled by the same integer factor as that was used at the Quantizer and then IDCT (inverse of DCT) is performed on each macroblock to reverse the DCT operation.

The output block of IDCT typically corresponds to the difference values that were produce by subtracting the motion compensated previous macroblock. The difference

values are added to a reconstructed area from the previous frame. The motion vector information is used to pick same reference area that was used in the encoder. The result is a reconstruction of the original frame. The reconstructed frame is placed in a frame store and it is used to motion-compensate the next received frame.

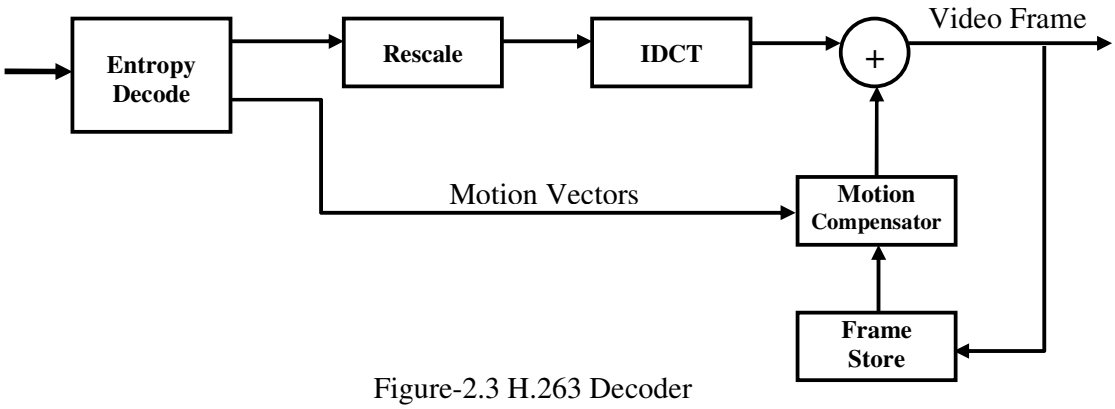


Figure-2.3 H.263 Decoder

2.6 Networks

Integrated Services Digital Network (ISDN)

The quality of ISDN conferencing is limited by the bandwidth of the connection, varying from 64kbit/s (ISDN-1) to 2 Mbps (ISDN-30). ISDN provides Guaranteed Quality of Service (GQoS), which means that if for example a 384kbit/s link is dialed then unless there is a fault condition the full 384kbit/s will be provided for the duration of the conference.

IP based network

Packet-based network such as Local Area Networks (LANs), Wide Area Networks (WANs), Metropolitan Area Networks (MANs) and the Internet all uses the Internet Protocol (IP) to pass information between networks. Disadvantage of IP transmission is that all data traffic has to compete for the available bandwidth. These networks are currently unable to offer Guaranteed Quality of Service (GQoS). Also the bandwidth requirement is higher (about 20%) on IP networks than on ISDN networks because of the packetizing overhead of IP. The advantage of making an IP based videoconferencing is

that many people already have a connection to an existing IP infrastructure. So making an IP based videoconferencing system is the cheapest solution.

2.7 RTP [4]

The real-time transport protocol (RTP) provides end-to-end delivery services for real time data, such as interactive audio and video over unicast or multicast networks. Those services include payload type identification, sequence numbering, time stamping and delivery monitoring. The data transfer is augmented by a control protocol (RTCP) to allow monitoring of the data delivery.

RTP itself does not provide any mechanism to ensure guaranteed quality-of-service (GQOS), but relies on lower-layer services to do so. For example it does not prevent out-of-order delivery but the sequence numbers included in the RTP packets allow the receiver to reconstruct the sender's packet sequence.

Figure 2.4 shows the RTP header format. First 12 bytes are present in every RTP packet, while list CSRC is present only when inserted by a mixer.

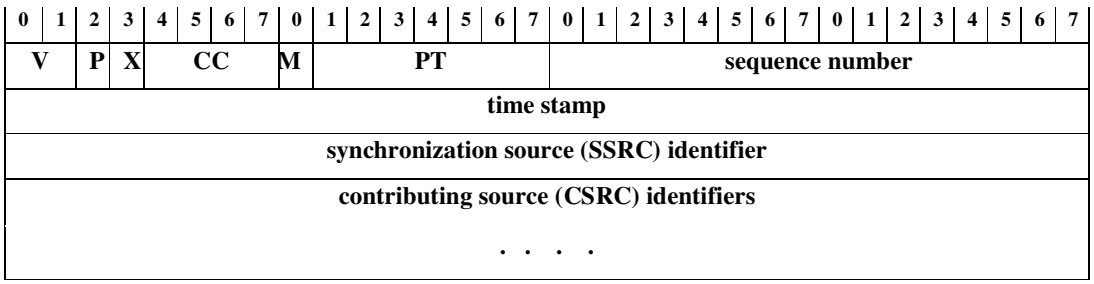


Figure – 2.4 RTP Header Format

The fields have following meanings: -

Version (V) identifies the version of RTP. **Padding (P)** set to 1 means the packet contains additional padding bytes. Last byte of padding contains a count of padding bytes including itself. **Extension (X)** set to 1 means header extension is used. **CSRC count (CC)** contains the number of CSRC identifiers. **Marker (M)** bit may be defined by a profile. **Payload type (PT)**identifies the format of the RTP payload.

Sequence number - 16 bit sequence number initialized with a random number and is incremented by one for each RTP packet sent. It can be used to detect the packet loss and to regenerate the senders order of the packets at the receiver.

Time stamp - 32 bit field reflects the sampling instant of first byte in the RTP data packet. Several consecutive RTP packets may have same time stamp if they are logically generated at once. Such as same video frame may be divided into multiple RTP packets.

SSRC - 32 bit field identifies the synchronization source. Each source should have a separate SSRC value within a RTP session. Multiple data streams from a single participant contain independent SSRC identifier, for example audio and video streams in a video conferencing application.

CSRS list - Contains up to 15 **SSRC** identifiers contributing source for the payload contained in the RTP Packet. For example an audio mixer add SSRC identifiers of all the sources that were mixed together.

RTP Translator and Mixer

RTP support the notion of two intermediate systems at RTP level "translator" and "mixer". A translator may be used to bridge two different type of network. A mixer may be used to mix multiple audio/video input streams into a single out stream after performing synchronization functions on input streams.

RTP Control Protocol (RTCP)

RTCP periodically transmits control packets to all participants in the session. It performs following four functions: -

1. Provide feed back on the quality of data distribution. It may be directly useful for control of adaptive encoding of audio/video streams in a videoconferencing application.
2. RTCP carries a persistent canonical-name (CNAME) for an RTP source to keep track of each participant. Since SSRC may change during a RTP session due to a collision or program restart. CNAME is also required to associate multiple streams from a single participant, for example to synchronize audio and video

streams coming from same participant. Since each stream contains different SSRC identifier.

3. Controlling the rate of the RTCP packets according to the number of participants. Since each participant sends its control packets to all other participants, each can independently observe the number of participants.
4. The fourth optional function is to convey other session control information, for example participant identifications to be displayed in the user interface.

RTCP specifies following types of control packets: -

- **SR** - Sender report for transmission and reception statistics from participants that are active sender.
- **RR** - Receiver report for reception statistics from the participant that are not active sender.
- **SDES** - Source description items including CNAME.
- **BYE** - indicating end of participation.
- **APP** - For application specific function.

2.8 Popular Standards

The International Telecommunication Union – Telecommunication Standardization Sector (ITU-T) has developed a family of recommendations for conferencing services over different network media. They are:

- H.310 – Broadband over Asynchronous Transfer Mode (ATM) networks.
- H.320 – Narrowband over Integrated Services Digital Network (N-ISDN).
- H.322 – Narrowband over LANs (Local Area Networks) that provide Guaranteed Quality of Service (QoS), an improved method of Internet Protocol (IP) transmission.
- H.323 – Narrowband over Packet Based Networks (IP), that do not provide guaranteed QoS.
- H.324 – Very narrow bandwidth over the existing General Switched Telephone Network (GSTN).

Each of the above recommendations defines a set of protocols, some mandatory others optional, that must be implemented by manufacturers of H.3xx equipment. It ensures interoperability of terminals and software applications.

2.8.1 H.323 standard [5]

H.323 specifies the components, protocols, and procedures for providing multimedia communication over packet-based networks. it specifies four kinds of components terminals, gateways, gatekeepers and multipoint control units (MCU).

Terminal can either be a personal computer (PC) or a stand-alone device, running an H.323 and the multimedia applications. It supports audio communications and can optionally support video or data communications.

Gateway bridges the H.323 video network to a non H.323 network. It translates the protocols for call setup and release, converts media formats between different networks, and transfers information between the networks.

Gatekeeper provides important services such as addressing, authorization and authentication of terminals and gateways, bandwidth management, accounting, billing and charging. Gatekeepers may also provide call-routing services.

MCU provides support for conferences of three or more H.323 terminals. All terminals participating in the conference establish a connection with the MCU. The MCU manages conference resources, negotiates between terminals.

Example of H.323

Microsoft's NetMeeting is a H.323 based application.

2.8.2 SIP [3]

Session Initiation Protocol (SIP) is an Internet Engineering Task Force (IETF) standard. SIP aimed at providing user location, user availability, user capabilities, and call setup. SIP provides many of the same features that H.323 does, but the SIP architecture is different. SIP users are located by a SIP URL which looks just like an e-mail address.

SIP is a text based protocol similar to HTTP, and it uses the Session Description Protocol (SDP) in the payload to describe user capabilities, i.e. what CODECs are supported by the user agent.

Example of SIP

Microsoft's Windows Messenger is a SIP user agent with support for both audio/video chat and application sharing.

2.9 Quality of Service

Quality of Service or QoS is the ability of a network element (e.g. an application, host, router) to have some level of assurance that its traffic and service requirements can be satisfied. The current Internet Protocol (IP) doesn't provide guaranteed QoS. It is based on the available network bandwidth and FIFO forwarding schemes. But the audio/video streams of a videoconferencing application are delay-sensitive and error-sensitive.

The Internet Engineering Task Force (IETF) and other research institutes have proposed many QoS protocols and schemes. Most common are ReSerVation Protocol (RSVP), Differentiated Services, Constraints based Routing, MultiProtocol Label Switching (MPLS), Internet2, etc. All of these are trying to present approaches to deliver network-level QoS controls over the Internet.

Besides providing the network-level QoS guarantee, the application-level QoS should also be provided. The sending end application may receive the feed backs from the receiving end applications to analyze the actual network load and may change its QoS parameters accordingly, for example we can vary the bit-rate and key frame rate of H.263 encoder.

Chapter - 3

Network Security: An Overview

As internet is becoming more and more commercial and larger, the security of the computer system is becoming much important. So the growth of the internet has spawned additional attacks to the computer system. The objective of security is to protect the computer systems from unauthorized persons and at the same time allow access to the authorized users. In this chapter we will briefly look at the basic principles of computer system security. We will focus mainly on the internet security and will discus the encryption algorithms that we are using in our secure video conferencing system.

3.1 Some Definitions

- **Security Policy**: A security policy is a formal statement of the rules which must be followed by people who are given access to an organization's information. The policy should specify the mechanisms through which these requirements can be met.
- **Security Attacks**: Security Attack is an action that compromises the security of the information owned by an organization for example, an unauthorized access to the information.
- **Security Mechanisms**: Mechanisms are designed to detect, prevent, or recover from the security attack
- **Security Services**: A service is that enhances the security of the information transfers of an organization. The services make use of security mechanisms.

3.2 The OSI security Architecture [6]

ITU-T Recommendation X.800, Security Architecture for Open System Interconnection (OSI), defines a systematic approach to organizing the task of providing security. It is focus on security services, mechanisms and attacks.

Security services

- **Authentication**: The authentication service is concern with assuring that the communicating entity is the one that it claims to be.
- **Access Control**: This service controls the access to the resources.
- **Data Confidentiality**: protects from unauthorized disclosure of transmitted data.
- **Data Integrity:** Data integrity assures that data received are exactly as sent by an authorized entity. That means the data received contains no modifications, insertion, deletion, or replay.
- **Nonrepudiation:** prevents either sender or receiver from denying a transmitted message.
- **Availability Services:** Protects a system from denial-of-service attacks.

Security Mechanisms

Security mechanisms include encryption, digital signatures, use of trusted third party (Notarization) etc. Basic building blocks are as follow:

- Encryption is used to provide confidentiality, can provide authentication and integrity protection.
- Digital signatures are used to provide authentication, integrity protection, and non-repudiation.
- Checksums/hash algorithms are used to provide integrity protection, can provide authentication.

Security Attacks

Security attacks can be divided into two parts, passive attacks and active attacks. In passive attacks the goal of the opponent is to obtain information being transmitted.

Passive attacks do not alter the information so they are very difficult to detect. Active attacks involve some modification of the information or creation of false information. There may be following kind of active attacks:

- **Masquerade Attack**: In this attack one entity pretends to be a different entity to gain unauthorized privilege. This attack usually includes one of the other active attacks.
- **Replay Attack**: Involves the passive capture of the messages being transmitted and its subsequent retransmission to produce an unauthorized effect.
- **Modification of Messages**: Some portion of a legitimate message is altered or messages are delayed or reordered to produce an unauthorized effect.
- **Denial of Service (DoS) Attack:** DoS Attacks are concerned only with consuming bandwidth and resources. DoS attacks are very easy to launch and difficult to track. It is not easy to refuse the requests of the attacker, without also refusing legitimate requests for service.

3.3 Symmetric key Encryption

Figure 3.1 shows a model of symmetric key encryption scheme. A source produce a message X called *plaintext*. A key K is generated. If key is generated at source end it must be provided to the destination by means of some secure channel. The encryption algorithm encrypts the plaintext X using key K and produce *ciphertext* Y. We can write this as

$$Y = E_K(X)$$

This ciphertext Y is transmitted over the network to the destination. At the destination Y is decrypted back to X by the decryption algorithm using same key K.

$$X = E_K(Y)$$

An opponent called *cryptanalyst*, observing Y but not having access to K, may attempt to recover X or K or both using some statistical information of plaintext. Cryptanalyst may know the encryption algorithm and also may have some sample plaintext-ciphertext pair for the key K.

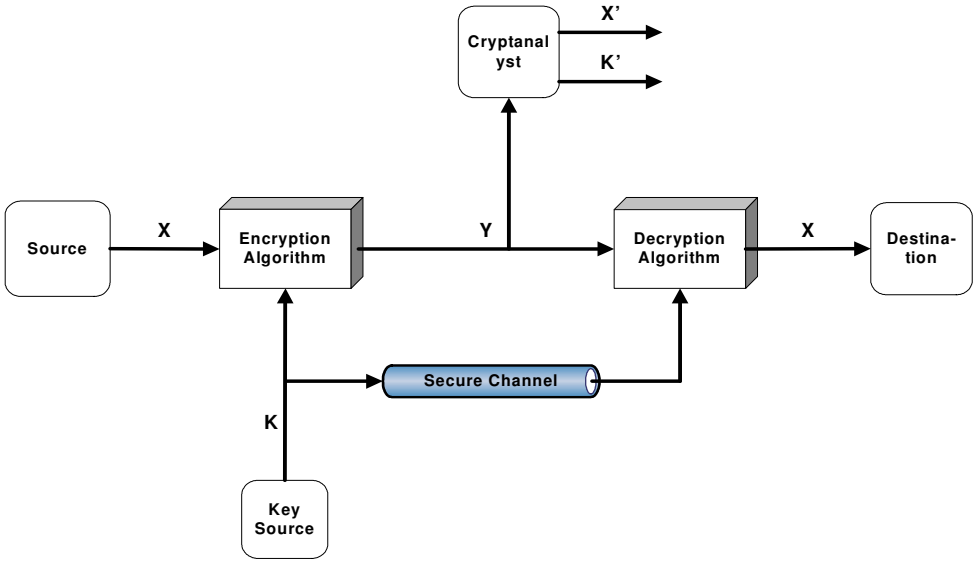


Figure 3.1 Model of Symmetric cipher

3.4 Block Cipher

In a block cipher (cryptographic system) a block of plaintext is treated as a whole and used to produce a ciphertext block of equivalent length. Almost all block ciphers are based on a structure referred as Feistel Cipher described next. Table 3.1 shows some popular symmetric block ciphers.

Cipher	Key Size	Block Size	Security
DES	56 bit	64 bit	Weak
AES	128,192,256 bit	128 bit	Strong
Blowfish	Variable up to 448 bit	64 bit	Variable
3DES	168 bit	64 bit	Strong

Table 3.1 - Comparison of common Block Ciphers

Confusion and Diffusion

In cryptography, **confusion** and **diffusion** are two properties of a secure cipher which were identified by Claude E. Shannon in his paper, "Communication Theory of Secrecy Systems" [7]. Confusion means that ciphertext statistics should depend on plaintext

statistics in a manner too complicated to be exploited by the cryptanalyst. Confusion can be achieved by the use of complex substitution algorithm.

Diffusion means that each digit of plaintext and each digit of the key should affect many digits of the ciphertext. Diffusion can be achieved by repeatedly performing some permutation on the data followed by applying a function to that permutation so that many bits from different positions in plaintext contribute to a single bit of ciphertext.

The Feistel Cipher

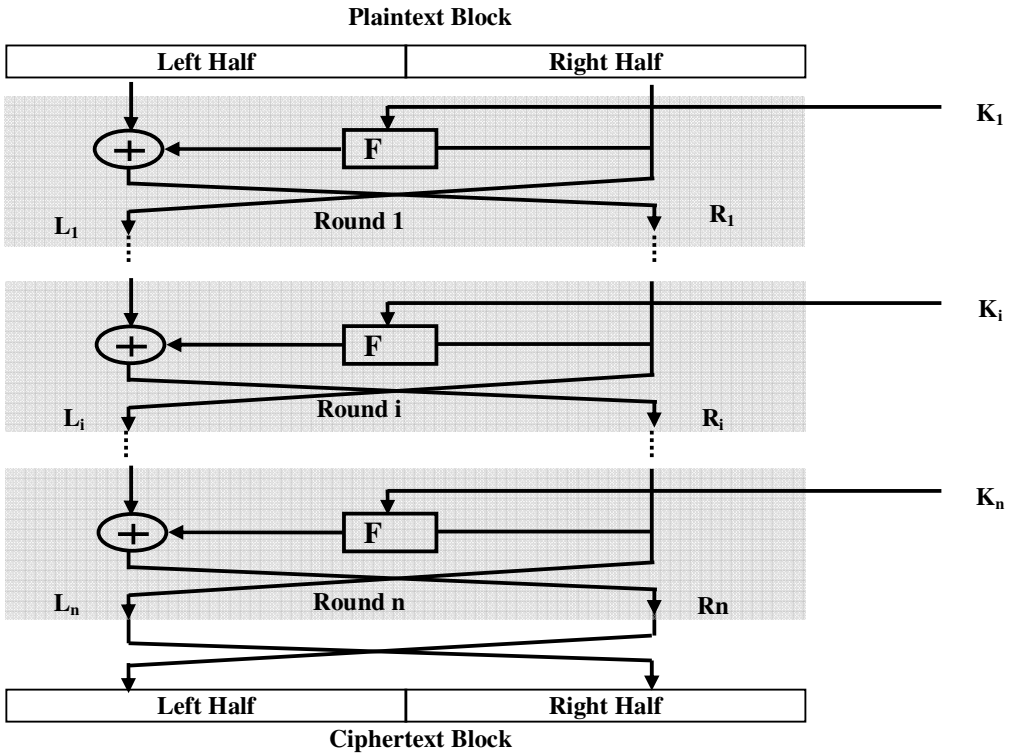


Figure 3.2 - Feistel Cipher structure

Feistel proposed the use of a block cipher that alternates substitutions and permutations. In fact, this is a practical application of Shannon's proposal. Figure 3.2 depicts the Feistel cipher structure. In Feistel structure the plaintext block is divided into two halves L and R. The two halves of data are passed through n rounds of processing and then combined to produce the ciphertext block. Each round i has as inputs L_{i-1} and R_{i-1}, derived from the previous round, as well as a sub key K_i, derived from the overall key K.

3.4 Data Encryption Standard (DES) [6]

The Data Encryption Standard (DES) is based on Feistel structure and is most widely used encryption scheme today. In DES, data is encrypted in 64 bit blocks using a 56 bit key. First the 64 bit plaintext passes through an initial permutation. This is followed by a phase consisting of 16 rounds of same function, which involves both permutation and substitution functions. Figure 3.3 depicts a single round of the 16 rounds.

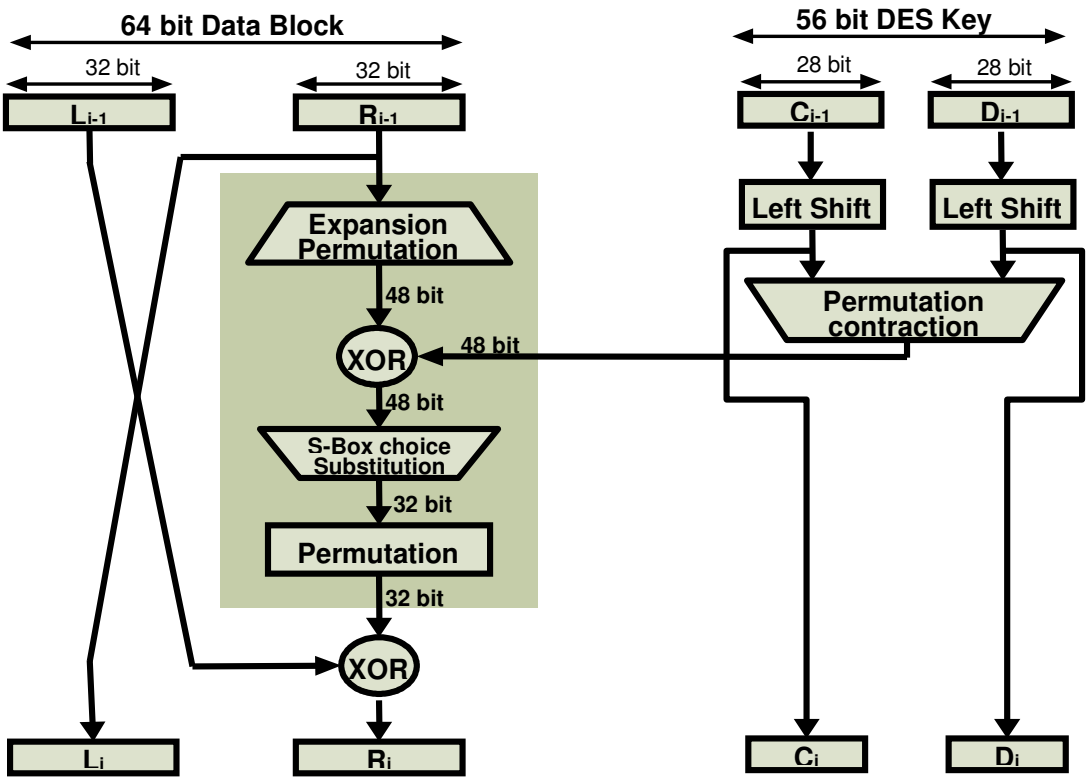


Figure 3.3 -One Round of DES Encryption Algorithm

Decryption uses the same algorithm as encryption by applying sub keys in reverse order. DES exhibits a strong *avalanche effect.* A change in one bit of plaintext or one bit of key results in differing about half of the bits in ciphertext.

Triple-DES

In DES only 56 bit key is used therefore it is vulnerable to a brute-force attack due to advances in computers. To increase the key size we can use multiple encryptions with

DES using multiple keys. Double-DES (with two encryption stage having two different stages) has a problem of *meet-in-the-middle* attack. Meet-in-the-middle attack requires $O(2^{56})$ time to find the 112 bit key using a single plaintext-ciphertext pair. Triple-DES uses three DES keys to encrypt plaintext P to ciphertext C. It has an effective key length of 168 (56X3) bits.

$$C = E_{K3}[D_{K2}[E_{k1}[P]]]$$

A number of internet applications have adopted three-key Triple-DES, including PGM and S/MIME. Currently, there is no cryptanalytic attack on Triple-DES.

3.5 Public key Cryptography and RSA

Because all keys in a secret-key cryptosystem must remain secret, secret-key cryptography often has difficulty providing secure key management, especially in open systems with a large number of users.

In public-key cryptography each person gets a pair of keys, one called the public key and the other called the private key. The public key is published, while the private key is kept secret. The need for the sender and receiver to share secret information is eliminated. All communications involve only public keys, and no private key is ever transmitted or shared. The only requirement is that public keys be associated with their users in a trusted (authenticated) manner. Anyone can send a confidential message by just using public information, but the message can only be decrypted with a private key by the intended recipient.

The RSA algorithm is named after Ron Rivest, Adi Shamir and Len Adleman, who invented it in 1977. The RSA algorithm can be used for both public key encryption and digital signatures. Its security is based on the difficulty of factoring large integers.

RSA Key Generation Algorithm

1. Generate two large random primes, p and q, of approximately equal size such that their product n = p*q is of the required bit length, e.g. 1024 bits.
2. Compute n = p*q and f = (p-1)*(q-1).

3. Choose an integer e, $1 < e < f$, such that GCD(e, f) = 1.
4. Compute the secret exponent d, $1 < d < f$, such that $e*d = 1 \bmod f$.
5. The public key is (n, e) and the private key is (n, d). The values of p, q, and f should also be kept secret.

Encryption

1. Obtains the recipient's public key (n, e).
2. Represents the plaintext message as a positive integer m.
3. Computes the ciphertext $c = m^e \bmod n$.
4. Sends the ciphertext c to recipient.

Decryption

1. Use private key (n, d) to compute $m = c^d \bmod n$.
2. Extracts the plaintext from the integer representative m.

Why RSA works

$$c^d = (m^e)^d = m^{e*d} = m^{kf+1} = m^{kf} * m \pmod n$$

but $m^f \bmod n = 1$ (Lagrange theorem)

Security of RSA

Finding d given (n, e) requires to factorize n into its two prime factor p and q. This enables calculation of $f = (p-1)*(q-1)$, which in turn, enables determination of $d = e^{-1} \bmod n$. So determining f given n is equivalent to factoring n. Hence, we can use factoring performance as a benchmark against which to evaluate the security of RSA.

Chapter – 4

Java Media Framework (JMF)

The Java Media Framework (JMF) is an application programming interface (API) for incorporating time-based media into java programs. It provides support for media playback, capturing and storing media data and performing custom processing on media data streams. It supports media data reception and transmission using RTP and RTCP. It also defines a plug-in API that enables advanced developers to customize and extend JMF functionality. JMF plug-in architecture supports custom codecs, multiplexer, demultiplexers, effects, and renderer plug-ins.

4.1 JMF Architecture

Figure 4.1 shows the high lever JMF architecture. A higher level Presentation and Processing API, manages the capturing, processing and presentation of media streams. A low level Plug-In API, allows the developers to customize and extend the JMF functionality.

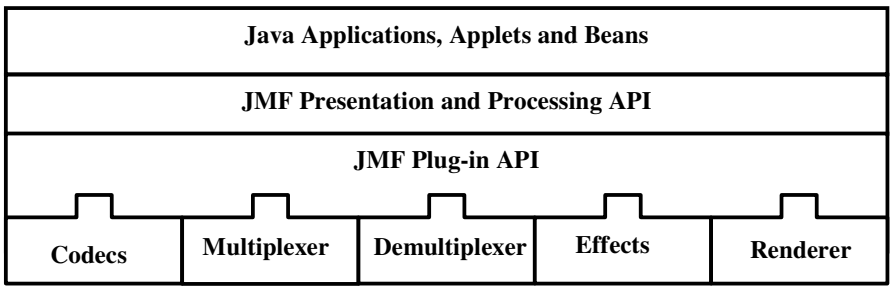


Figure - 4.1 High level JMF Architecture

4.2 JMF basic classes and interfaces

Following are the basic classes/interfaces of JMF API: -

Managers

JMF provides four manager classes:

- Manager - Handles the construction of Players, Processors, DataSources, and DataSinks. Manager's create methods is used to construct the Players, Processors, DataSources, and DataSinks.
- PackageManager - Maintains a registry of packages that contain JMF classes, such as custom Players, Processors, DataSources, and DataSinks.
- CaptureDeviceManager - Maintains a registry of available capture devices.
- PlugInManager - Maintains a registry of available JMF plug-ins. If we implement a new plug-in, we can register it with the PlugInManager to make it available to Processors that support the plug-in API.

MediaLocator

A MediaLocator describes the location of media stream. It is similar to a URL and can be constructed from a URL, but can be constructed even if the corresponding protocol handler is not installed on the system.

DataSource

DataSource manage the transfer of media-content. It encapsulates both the location of media and the protocol and software used to deliver the media. A DataSource is identified by a JMF MediaLocator. JMF DataSource objects can be categorized as, *Pull Data-Source* (the client initiates and controls the data transfer), *Push Data-Source* (the server initiates and controls the data transfer). A cloneable data source can be used to create clones of either a pull or push DataSource.

Player

A player processes an input stream of media data and renders it at a precise time. A DataSource is used to deliver the input media-stream to the player. The rendering

destination depends on the type of media being presented. Figure 4.2 shows the JMF Player class. A player can be in one of six states, *Unrealized*, *Realizing*, *Realized*, *Prefetching*, *Prefetched and started.* It posts TransitionEvents as it moves from one state to another. The ControllerListener interface provides a way for a program to determine what state a player is in and to respond appropriately. For example, when a program calls an asynchronous method on a Player or Processor, it needs to listen for the appropriate event to determine when the operation is complete.

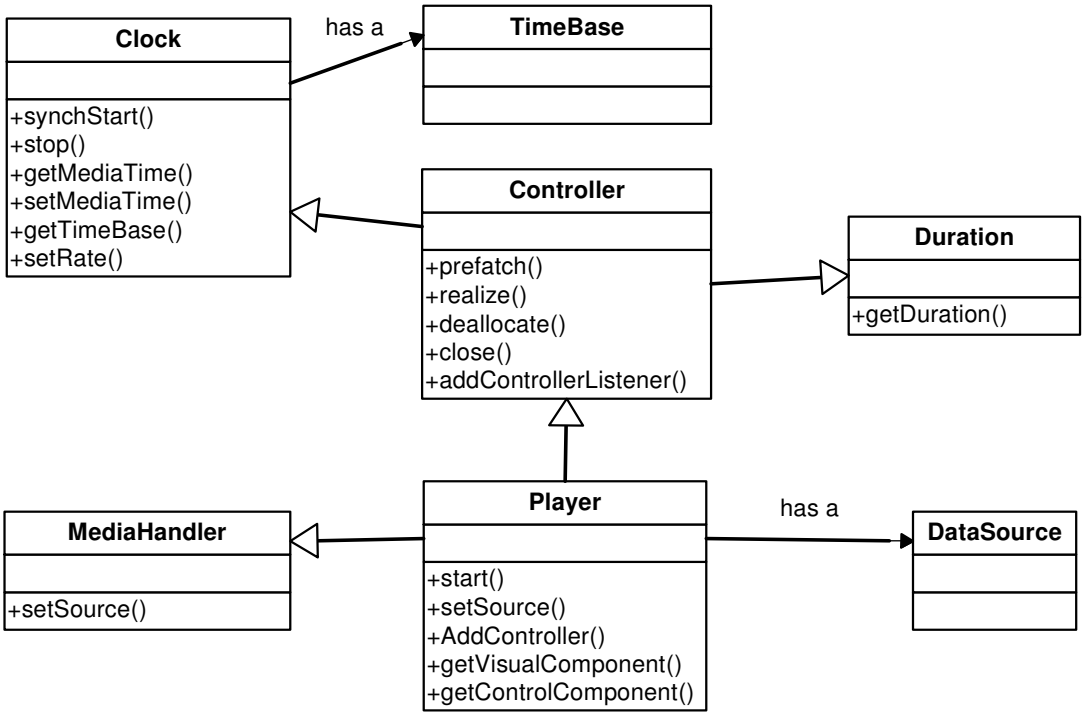


Figure 4.2 JMF Player Class

Processor

A Processor is a Player that takes a DataSource as input, performs some user-defined processing on the media data, and then outputs the processed media data. A Processor can send the output data to a presentation device or to a DataSource. If the data is sent to a DataSource, that DataSource can be used as the input to another Player or Processor, or as the input to a DataSink. A Processor has two additional standby states, *Configuring* and *Configured* in addition with six states of a Player.

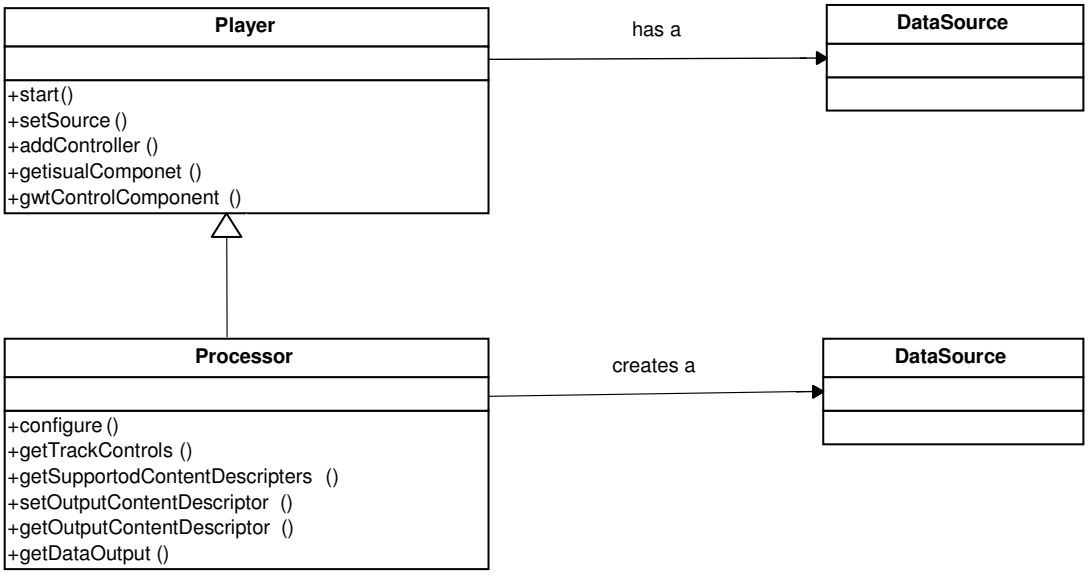


Figure 4.3 JMF Processor Class

While a Processor is in the *Configured* state, getTrackControls() method can be called to get the TrackControl objects for the individual tracks in the media stream. These TrackControl objects enable us to specify which plug-ins we want to use to process a particular track. We can also set our custom Plug-In to perform required operation, for example we implement a codec Plug-Ins to perform Encryption/Decryption on audio/video tracks.

DataSink

A DataSink is used to read media data from a DataSource and render the media to some destination. Destination may be a file or a network. DataSink objects are constructed through the Manager using a DataSource. A DataSink posts a DataSinkEvent to report on its status.

4.3 JMF RTP Architecture

The JMF RTP APIs are designed to work seamlessly with the capture, presentation, and processing capabilities of JMF. Players and processors are used to present and manipulate RTP media streams just like any other media content. Figure 4.5 shows a high level JMF RTP architecture.

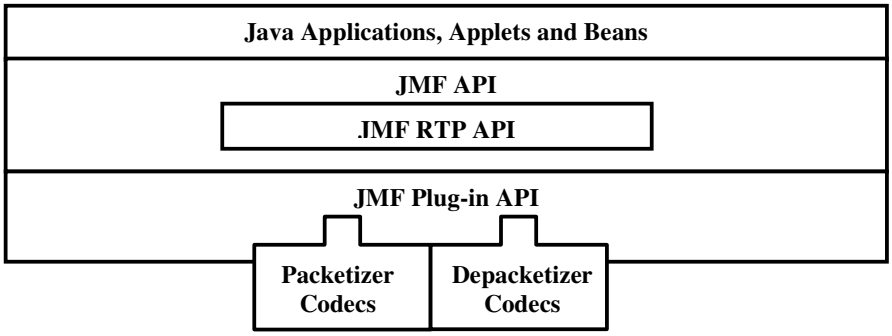


Figure - 4.4 JMF RTP Architecture

RTP Session

An RTP session is identified by a network address and a pair of ports. One port is used for the media data and the other is used for control (RTCP) data. Each media type is transmitted in a different session. For example, if both audio and video are used in a conference, one session is used to transmit the audio data and a separate session is used to transmit the video data.

Session Participant

A participant is a single machine, host, or user participating in the session. Participation in a session can consist of passive reception of data (receiver), active transmission of data (sender), or both.

Session Manager

Session Manager is used to coordinate an RTP session. It keeps track of the session participants and the streams that are being transmitted. The SessionManager interface defines methods that enable an application to initialize and start participating in a session, remove individual streams created by the application, and close the entire session.

Session Statistics

The session manager maintains statistics on all of the RTP and RTCP packets sent and received in the session. Statistics are tracked for the entire session on a per-stream basis. The session manager provides access to global reception and transmission statistics:

- **GlobalReceptionStats**: provides access to overall data and control message reception statistics for this session. We can get GlobalReceptionStats object by calling getGlobleReceptionStats() method of SessionManager.
- **GlobalTransmissionStat**s: provide access to overall data and control message transmission statistics for this session. We can get GlobleTransmissionStats object by calling getGlobleTransmissionStats() method of SessionManager.

These statistics are useful to monitor the quality of the media streams at the receiving end for the purpose of controlling QoS of the streams.

Reception of RTP Media Streams

To play all of the ReceiveStreams in a session, we need to create a separate Player for each stream. When a new stream is created, the session manager posts a NewReceiveStreamEvent. In the event handler of this event we can retrieve the DataSource from the ReceiveStream and pass it to Manager.createPlayer() method to construct a player for this stream. A canonical name (CNAME) is associated with each media stream. The streams with same CANME are synchronized by the JMF internally.

Transmission of RTP Media streams

Following steps are needed to create a send stream to transmit data from a live capture source:

1. Create, initialize, and start a SessionManager for the session.
2. Construct a Processor using the appropriate capture DataSource.
3. Set the output format of the Processor to an RTP-specific format. An appropriate RTP packetizer codec must be available for the data format we want to transmit.
4. Retrieve the output DataSource from the Processor.
5. Call createSendStream on the session manager and pass in the DataSource.

We can control the transmission through the SendStream start and stop methods.

Chapter – 5

Design and Implementation

5.1 Client-Server Architecture

We developed our Secure Video Conferencing System (SVCS) over client-server architecture because of the security needs. The system includes a centralized server and distributed client. Distribution of audio and video data between the client nodes is done using point-to-point connections between clients. The server manages the connection between the clients. It server is also responsible of distribution of session key for encryption of audio video streams and controlling the QoS. Figure 5.1 shows the basic client server architecture of our system.

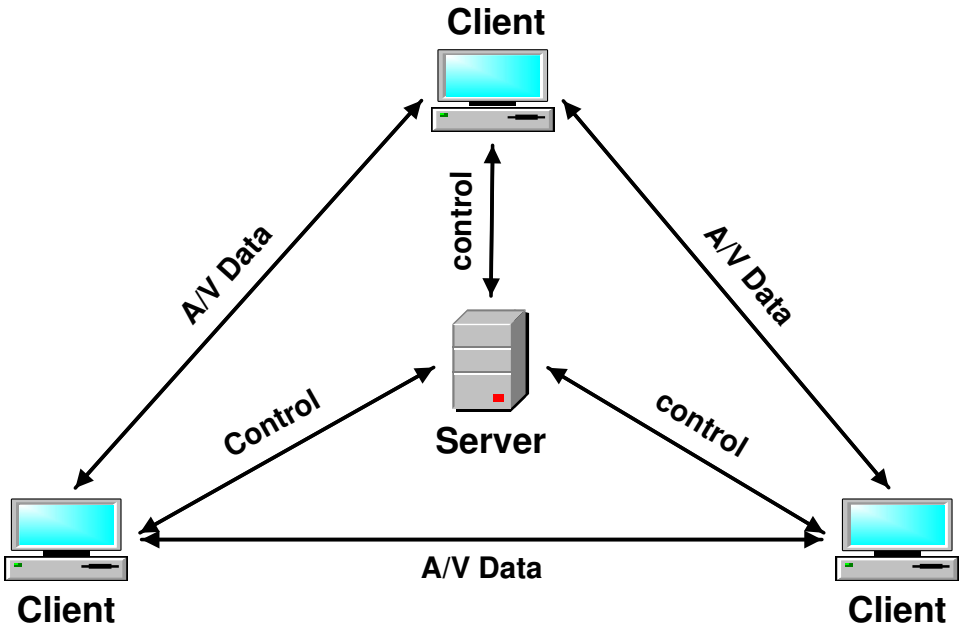


Figure 5.1 – Client-server Architecture

5.2 Functions of the server

Server side application performs following functions: -

- **Registration of the users**: Each user must be registered at the server in order to take part in the conference. At the time of registration user will provide his information to the server along with his/her public key. User will get a unique user-id and server's public key. These public keys will be used at the time of authentication.
- **Authentication of users**: At the time of joining a conference by a user the server and the user, both will authenticate each other using their private-public key pairs.
- **Distribution of the session key**: After authenticating a user, the server will send the session key to the client application running at the user side in a secure manner. This session key will be common for all users taking part in a particular conference.
- **Maintaining the state of the conference**: Whenever a user will join or leave a conference the server will inform the other users of that conference and provide them necessary information in order to maintain the state of the conference.
- **Controlling QoS**: Server will also responsible of controlling quality of service (QoS) of the audio/video data exchanged between the clients. In order to do that, the server gets the feedbacks from the clients about every media stream they are receiving. For each client, the server maintains QoS statistics from the feedbacks of all receivers of that client and whenever required it sends the QoS control signals to the client to maintain the quality of media stream being sent by that client.

5.3 Functions of the client

Following are the basic functions performed by the client side application

- **Session setup**: The client application will interact with the server to get the session key and information of other users to setup a conferencing session.
- **Capturing the audio/video stream**: The client application will be responsible of capturing user's real time audio and video streams from the capturing devices.

- **Compression and Encryption of the audio/video streams**: The client will compress the audio and video streams to reduce the bandwidth requirements. it will also encrypt each packet of the audio video stream before transmission using the session key got from the server.
- **Creation of RTP session**: The client application will create RTP sessions for transmitting real time audio/video streams of the users to other users taking part in the conference. Two separate sessions will be created for audio and video.
- **Opening RTP sessions**: The client application will open RTP sessions for each user whose audio/video streams the user wants to receive.
- **Decryption and Decompression**: Decryption and Decompression will be done both audio and video streams of each user to get the streams in their original form.
- **Rendering the audio/video stream**: Finally the incoming audio and video streams will be rendered and will be sent to the speakers and display unit respectively.
- **Maintaining QoS**: The client application will monitor the incoming audio/video streams and will send the feedbacks to the server. It will change the parameters of the media streams being transmitted accordingly whenever it will get the QoS controls from the server for example reducing the bit rate.

5.4 Security protocols

We use RSA public key algorithm (with 2048 bit key) for authentication of the users at the server and Triple-DES symmetric key algorithm for encrypting audio video streams by the clients before transmission. The symmetric key (called session key) is generated by the server at commencement of every new conference and is distributed in the clients. So every client taking part in a particular conference will have the same session key. Both encryption and decryption is done using same key.

Registration of the users

Every user must be registered at server before they can take part in any communication. Since we don't have any secure channel to pass the user's authentic information to the server via network we assume that the user itself will come at the server room and the registration will be done in the presence of the administrator of the server. The user will

required to generate a RSA key pair on his/her own computer and will bring the public part of the RSA key pair at the server. At the time of registration the user will submit his/her public key and will get a unique user_id and the server's public key.

Authentication and key distribution

Figure 5.2 shows the protocol for authentication and key distribution. In the first message time_stamp is used to provide freshness guarantee of the user's request in order to guard against *replay attack*. The challange1 and challenge2 are long random numbers freshly generated by the client and the server respectively.

After getting challenge1 back from the server, the client ensures the server's authenticity because nobody other than server can extract the challenge1 from first message. Similarly the server ensures the authenticity of client after getting challenge2 back. After first three messages both the client and the server have authenticated each other.

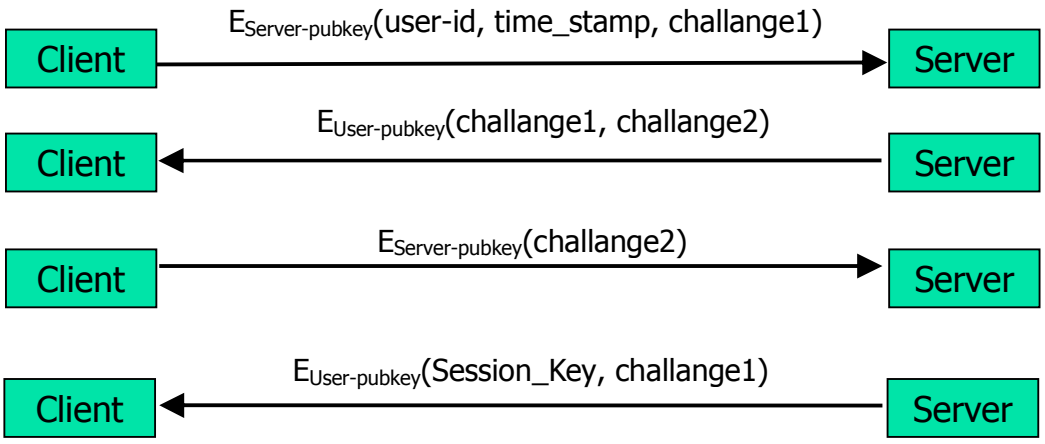


Figure 5.2 Authentication Protocol

In the fourth message server sends the session key to the user. User's challenge (challenge1) is included in the fourth message to prevent replay attack. Every subsequent message from this client to the server includes challange2 and every subsequent message from the server to this client will include challenge1 because these random numbers are freshly generated for current session and ensure the client and the server that message

having these number can't be a copy of message of some previous session and hence prevent the *replay attack*.

5.5 Maintaining the state of the conference

Creating or joining a conference

Every registered user can create a new conference at any time. There can be multiple conferences running at the same time. Following are the steps for creating a new conference: -

1. Client program sends connect request to the server and both the client and server authenticate each other as described in section 5.4.
2. Server sends list of conferences already running. User can select a conference to join or he/she can choose to create a new conference. Client sends the user's request to the server.
3. If the request is for a new conference, the server starts a new conference thread and add that user in the conference. Server then generate a new session key for this conference and send a secure copy of this session key and a unique port base (every client in a particular conference is assigned a unique port base for sending audio and video streams) to the client.
4. If the request is for joining an existing conference the server sends the session key of the conference and port base to the client and then sends following information:
    a. If the user wants to join as active mode
        i. The server sends the information (name, IP, port base) of all other participants in the conference to this client.
        ii. The server sends the information of this participant to all other clients.
    b. If the user wants to join as passive mode(listener only)
        i. The server sends the information of all other active participants in the conference to this client.
        ii. The server sends the information of this participant to all other active clients.

Leaving the conference

Client sends a left request to the server. Server disconnect the connection to this client and does one of the following:

1. If this user was the last user in the conference then server close the conference and remove the conference from the list of conference.
2. If the user was an active participant the server sends the information about leaving the conference by this client to all other participants in the conference.
3. If the user was a passive participant the server sends the information about leaving the conference by this client to all other active participants in the conference.

5.6 Audio/Video Transmitter

The transmitter module is a JMF/RTP based application. JMF manager class is used to create a merged data source object for audio and video. This data source object is passed to the manager class to create a processor object. While the Processor is in the *configured* state, it's track control objects are obtained for the individual audio and video tracks. Codecs for compression and encryption are set to both of the audio and video tracks. After setting the codecs both of the tracks, processor's realize method is called. While the processor is in *realized* state the output data source is created from the processor. The output data source object is passed to the RTP manager object to create RTP sessions for audio and video transmission.

We developed our own RTP connecter class to send RTP and RTCP packets to the remote participants over the network by encapsulating them in UDP packets and to receive RTP and RTCP packets from the remote participants. Since we are not using RTCP for getting feedbacks from the receivers, we do not transmit RTCP data all the time. From our RTP connector class we can control the amount of RTCP data to be transmitted. An object of RTP connector class is passed to RTP manager class at the time of creation RTP manager object.

Video Transmitter

We use IBM's implementation of H.263 encoder for video compression. We developed the CODECs for encryption and decryption the audio/video streams using Triple-DES encrypter of Java's crypto library. Figure 5.3 depicts the process of transmission of video stream. Video frames are generated by data source in YUV format. Each YUV frame is compressed to H.263 frame. The compressed H.263 frames are encrypted using Triple-DES encrypter and are passed to the RTP manager object. RTP manager encapsulates the encrypted H.263 frames in the RTP packets and passes them to the RTP connector object, which in turn, send the RTP packets to the remote participants by encapsulating them in UDP packets.

RTP connector sends a separate UDP packet for each RTP packet to each of the remote participant. We can add or remove a recipient dynamically. H.263 encoder provides some controls like bitRateControl, keyFrameRateControl etc, that we can use to control the QoS dynamically during the transmission.

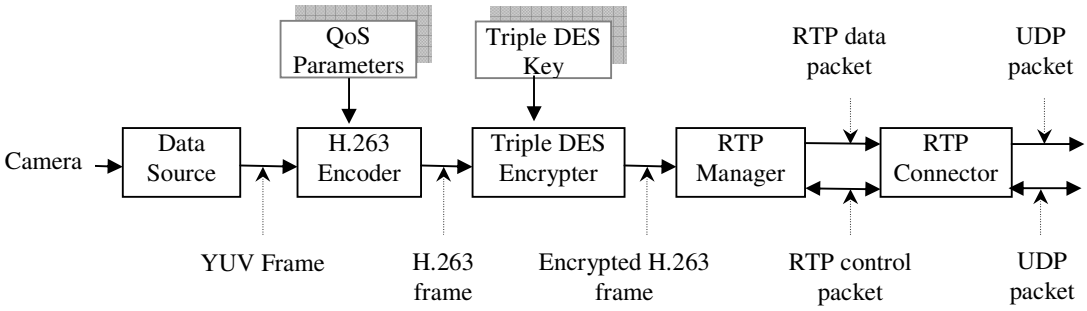


Figure 5.3 - Video Transmitter

Audio Transmitter

Figure 5.4 depicts the process of audio transmission. we use IBM's implementation of G.723 encoder for audio compression. G.723 is very efficient compression technique. Using it we can transmit good quality audio data of human voice at 6 to 7 kbps. Since G.723 encoder generates very small sized packets G.723 packetizer is used to format

packets of size 48 byte. The Encryption is done after the packetizer. Rest of the process is exactly same as for video transmission.

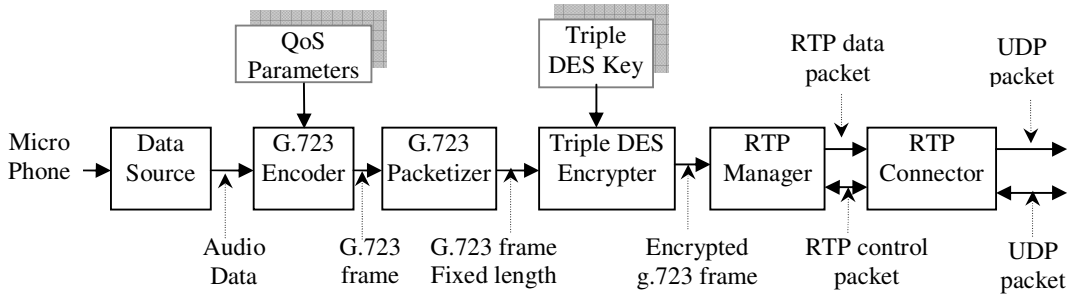


Figure 5.4 - Audio Transmitter

5.7 Audio/Video Receiver

Video Receiver

At the receiver side we get Encrypted H.263 frame from the RTP Manager. It is decrypted using Triple-DES decrypter and then decoded back into YUV format using Sun's implementation of H.263 decoder. YUV frame converted into RGB frame and it is then displayed into the receiver's window using direct video renderer. Figure 5.5 shows the video reception process.

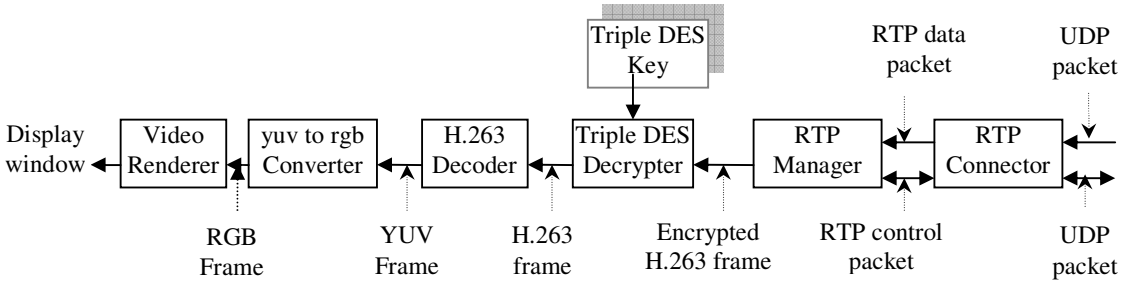


Figure 5.5 - Video Receiver

Audio Receiver

Figure 5.6 shows the video reception process. It is same as video receiver except that it uses G.723 decoder and sound renderer.

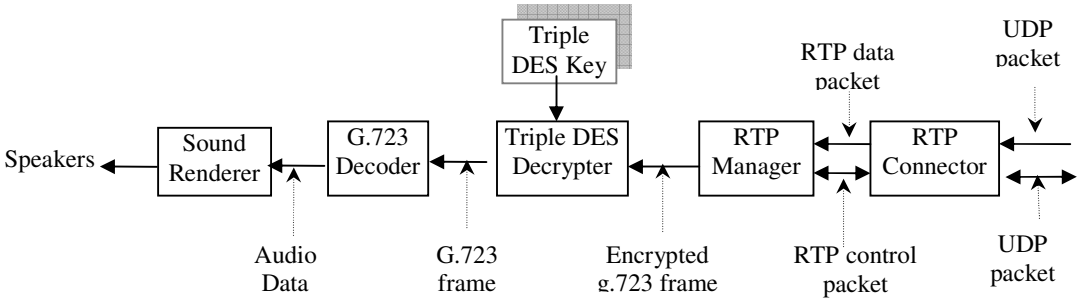


Figure 5.6 - Audio Receiver

5.8 Quality of Service

We provide application level end-to-end dynamic QoS self-adaptation features in our application. The application tries to provide best possible quality of video streams. Audio stream since uses less than five percent of total band width so we always keep the bit rate of audio stream to maximum (6.4 kbps) and we change the bit rate of video stream according to the network load.

In a JMF's RTP session every receiver sends its receiver reports as RTCP packets to the source. We could use these RTCP reports to analyze the network status directly at the source and every source could manage the quality of its output streams directly. But we don't use JMF's RTCP reports because of following two reasons:

- Suppose there are n participant in a video conferencing and each participant is sending audio and video streams to all other participants and each client node issue one RTCP packet per second to all of its sources (n-1) then every client node

need to send n-1 RTCP packets and receive n-1 RTCP packets per second. It may cause source overload.

- Secondly JMF's RTCP reports are not reliable and time consuming also. We tested it for only two nodes sending audio/video to each other over LAN. If we disable the RTCP we get good quality video with full frame rate (15 fps) but if enable the RTCP the video frame rate decreases to about 12 fps.

So we use our server to manage QoS. Server collects the feedback reports of a source from all of it's receivers to analyze the current network load and sends the QoS controls to adjust the bit rate of video stream. Now each client node construct a single packet of feed backs of all of its sources and send that packet to the server after every time stamp. There are few byes of feedback information for every source so the packet size is not to big. Every client node receives the QoS control packet from the server only when there is a need of change. So there is minimal use of bandwidth fraction for QoS control.

Generating feedbacks

RTP session manager object keeps the statistics of the receiving media stream. Session manager's getGlobleReceptionStats() method returns globalReceptionStats object. globalReceptionStats object includes the packet loss information from which we can calculate the percentage of packet loss since previous feedback. Percentage packet loss of all remote participants (source of audio/video streams) is encapsulated in a single packet and is sent to the server after every fixed time interval. Server receives the feed back reports from all participants and for each source it combines the feedbacks received from all of its targets.

Adjusting bandwidth

Running average of packet loss is calculated using following formula [9]. Where λ is the running average, p is the current packet loss and $0 <= \alpha <= 1$ is represents the influence of the current value to the running average we keep $\alpha = .5$ for our application.

$$\lambda = (1-\alpha)*\lambda + \alpha*p$$

Each source may be in one of the five states (1 2 … 5) in increasing quality. Initially all source are set to state number 3. According to the value of the running average of packet

loss server can send the control signals to the source either for decrease or for increase the quality level. JMF's H.263 encoder provides the methods setBitRate() and setKeyFrameRate() to set the bit rate and key frame rate dynamically. On reception of a decrease signal transmitter decrease the bit rate. It may also change the key frame rate. For high packet loss the key frame rate is decreased for the oblivious reason.

5.9 Data Flow Diagrams

Figure 5.7 shows the 0-level DFD for the client side application. It has one data store which maintains the static data like user's settings for audio/video streams, user information (user_id, private_key etc), server information. Data store is also used to store runtime information like information of remote participants. Secure Video Conferencing Client (SVCClient) provides the interface to the user for configuring the system, joining a conference, creating new conference etc. It interacts to the server and other clients through the external entity Network.

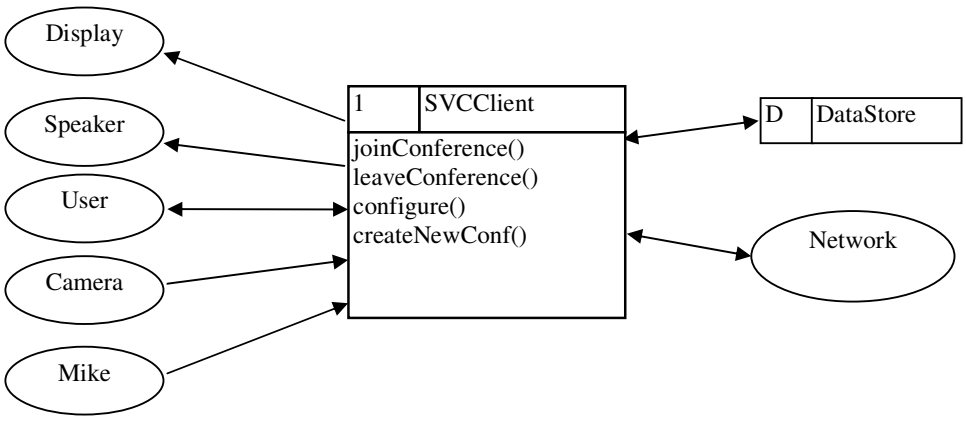


Figure 5.7 : Level-0 DFD of Client

Figure 5.8 shows the Level-1 DFD for the SVCClient. The SVCClient process is divided into five processes. UserInterface provides GUI to the user. DataHandler maintains the information stored in DataStore and provides the stored information to the different processes as needed.

SessionManager process interact with the server to maintain the state of the conference and to send the feedbacks of the client to the server and to get the QoS signals from the server. Whenever it gets the new participant information from the server, it starts a new receiver for that participant and also adds the participant in the target list of transmitter. After every fix interval of time it collects the feedbacks of all receivers and sends to the server.

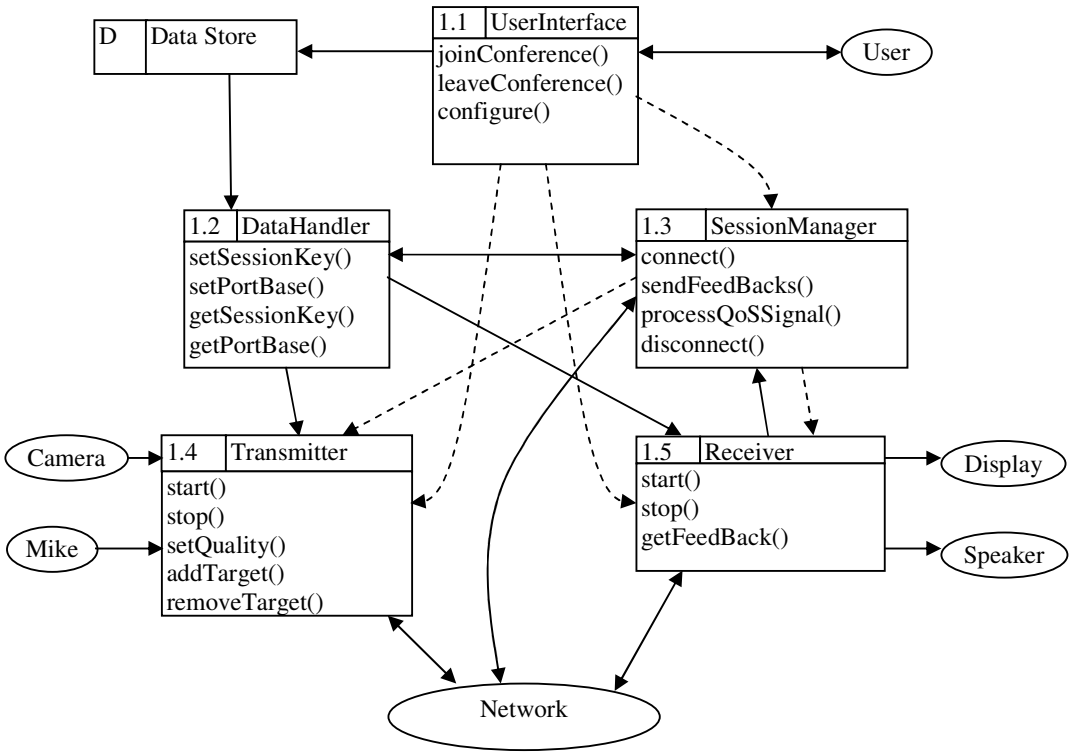


Figure 5.8 : Level-1 DFD of Client

Transmitter process do the transmission of audio and video to the remote participants over the network. It provides the method for adding and removing a target to its target list dynamically. Also we can set the quality of H.263 video stream dynamically.

Receiver process receives the audio/video streams of a single remote participant and sends them to the output devices after rendering. There will be multiple receiver process one for each remote participant. It also maintains the statistics of incoming audio/video streams and provide the feedbacks to the SessionManager process.

5.10 Overall Architecture

Figure 5.9 shows the overall architecture of our secure video conferencing system. Session manager module in the client side and the server thread are responsible of maintaining the state of the conference. For each new participant the server creates a new client listener thread for that participant to receives the feedback of the participant and a new participant object to maintain the participant's statistics.

QoS Feedback thread at the client side sends the feedbacks to the client listener thread which extracts the feedback of individual participant and update the running average maintained in the participant objects as described in the section 5.7. QoS service controller threads at server side checks the running average of all participants after every fix interval of time and sends the QoS controls to the session manager of the client if required.

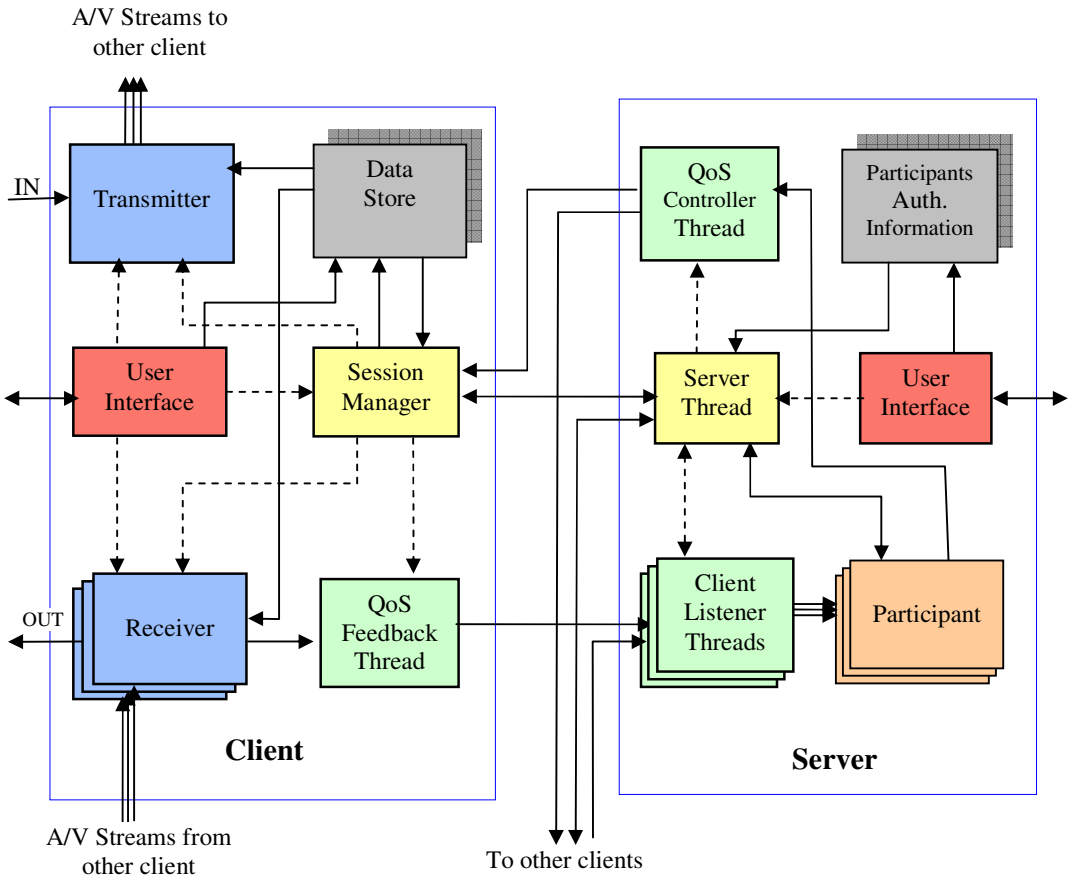


Figure-5.9 Overall Architecture

Chapter – 6

Testing and Future Goals

We tested our application to analyze its performance against network parameters like packet loss and its requirements like CPU, memory, network bandwidth. Figure 6.1 shows a snapshot of the client side application. To simulate the multi-users conference we send duplicate audio/video streams of one client to another client on different ports.

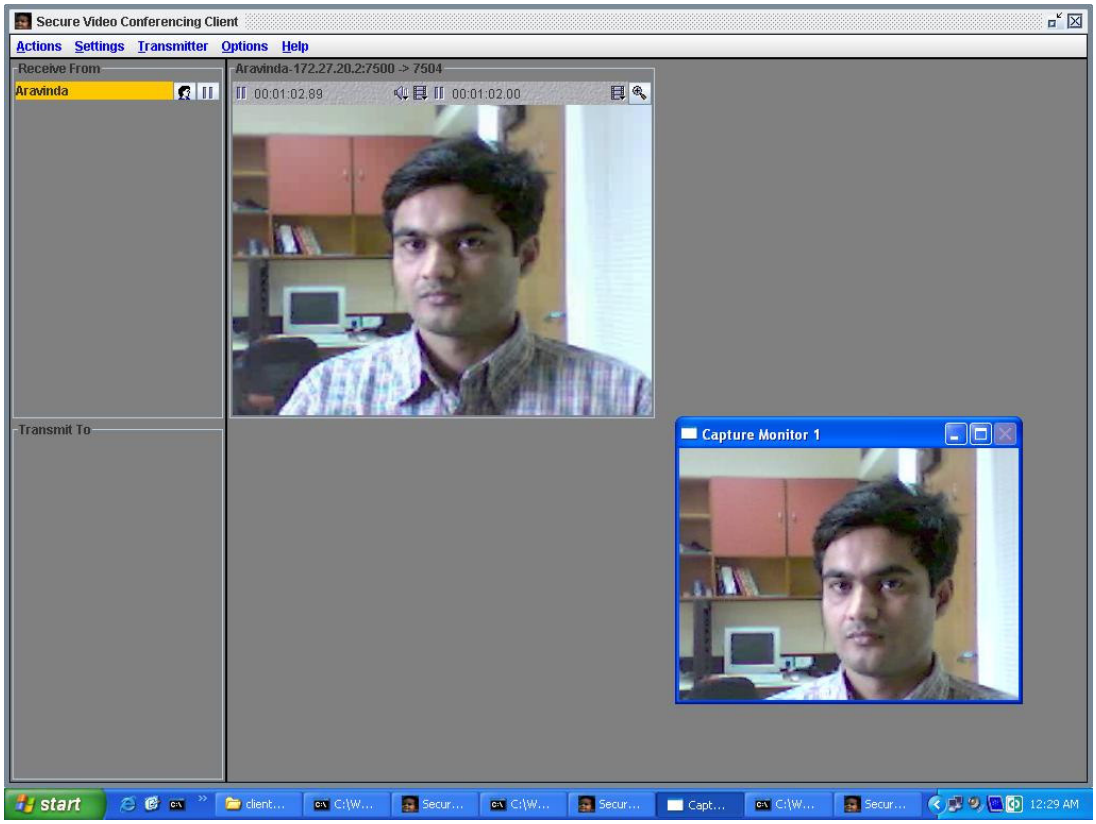


Figure 6.1 A Snapshot of the videoconferencing

6.1 CPU Requirement

We analyze the CPU and memory requirement of the client application by simulating the environment of different number of participant in the conference on a P-IV computer with 2.4 GHz speed and 512 MB RAM. The CPU and memory requirement as shown in Table 6.1 indicates that we can easily run a 10-12 users conference on a P-IV m/c while doing all the processing like compression, decompression, encryption, decryption in software (no extra hardware required).

Number of Targets	Number of Receivers	CPU Usage	Memory Usage
0	0	5 %	30 MB
1	1	35 %	45 MB
5	5	55 %	65 MB
10	10	85 %	85 MB
12	12	94 %	95 MB

Table 6.1 - CPU and memory usage of the application

6.2 Bandwidth Requirement

To analyze the network bandwidth required for transmitting audio and video streams of single user we very the bit rate of H.263 encoder and observe the quality on video stream at the receiver end. We observe that up to 100 kbps we can transmit acceptable quality video of CIF format. Audio stream uses fix bandwidth of about 7 kbps.

Bit Rate in kbps	Frame Rate	Frame Size	Video Quality
300	15	352X288	As captured
200	15	352X288	Good
100	15	352X288	Still acceptable
50	15	352X288	Slightly degraded

Table 6.2 - Bandwidth requirement

6.3 Performance against packet loss

To observe how the quality of video effected with packet losses, we introduce some packet loss by arbitrarily dropping some fraction of packets in the RTP connector module and observe the video quality at the receiver end. For high packet losses we choose the key frame rate small so that receiver can resynchronize more quickly after a loss of packet. Up to five percent packet loss can be tolerated.

Packet Loss (%)	Key Frame Rate	Video Quality
0.5%	15	No much effect
1.0%	12	Slightly disturbed
2.0%	10	More disturbed
5.0%	5	Poor

Table 6.3 - Performance against packet loss

6.4 Future improvements

So far we are using point-to-point connection between the clients to deliver their audio video streams. This may cause wastage of the network bandwidth. IP-Multicast could be a solution but it has some problems. Anyone can launch a *denial of service* attack by sending voluminous media streams to multicast address. Not all routers are multicast enabled. Collisions in multicast-IP/port may also occur. We may implement a distributed brokering system as described in [20] and [21] to distribute the media streams of the users.

Another improvement we can do is detection of QoS degradation as early as possible. As an early detection of packet losses we can use packet delay jitter at the receiver end.

In future we may make our application as a web-service so that registered user may run the client application from any where using a web browser. Digital certificates may be used to register a user online. Smart cart may be used to carry private keys.

Bibliography

[1] http://www.video.ja.net/intro/#top**,** Introduction to Video Conferencing System.

[2] http://www.videnet.gatech.edu/cookbook.en/, Video Conferencing Cookbook.

[3] http://gnrt.terena.nl, TERENA's Guide to Network Resource Tools.

[4] Schulzrinne, Casmer, Fredrick and Jacobson, “RTP: A Transport Protocol for Real-Time Applications”, Nonember 20, 2001.

[5] http://www.iec.org/online/tutorials/h323, H.323 Tutorial from International Engineering Consortium.

[6] William Stallings, “Cryptography and Network Security, Principles and Practice”, Third Edition, Pearson education, 2005.

[7] Shannon, Claude. "Communication Theory of Secrecy Systems", Bell System Technical Journal, 1949.

[8] Wenbiao Zhu and Nicolas D. Georganas, “JQoS: Design and implemention of QoS based internet videoconferencing system using the Java Media Framework(JMF)”, 2001

[9] Ingo Busse, Bernd Deffner, Henning Schulzrinne, “Dynamic QoS Control of Multimedia Applications based on RTP”, March 17, 1995

[10] Richard V. Cox, Peter Kroon, “Low bit-rate speech codec for multimedia communication”, November 1996

[11] http://www.ece.cmu.edu/~ece796/documents/g723-1e.pdf, ITU-T G.723.1 Recommendations

[12] Roalt Aalmoes, “Video compression techniques over low-bandwidth lines”, August 27, 1996

[13] Deepak Turaga and Tsuhan Chen, “ITU-T video coding standards”, 2000 (available from http://amp.ece.cmu.edu/Publication/Deepak/bookchap.pdf)

[14] http://www.4i2i.com/h263_video_codec.htm, An introduction to the ITU-T H.263 video compression standard: concepts, features and implementations

[15] http://www.data-compression.com/lossless.html, Huffman coding

[16] http://java.sun.com/products/java-media/jmf/2.1.1/guide/index.html, Java Media Framework(JMF) API Guide

[17] http://java.sun.com/j2se/1.4.2/docs/guide/security/jce/JCERefGuide.html, Java Cryptographic Extension (JCE) reference guide

[18] Wainhouse Research, "Traversing Firewalls and NATs With Voice and Video Over IP"

[19] Loic Oria, "Approaches to Multicast over Firewalls: an Analysis", August 1999

[20] Ahmet Uyar, Wenjun Wu, Hasan Bulut, Geoffrey Fox, "Service-Oriented Architecture for Building a Scalable Videoconferencing System", March 25 2006

[21] Ahmet Uyar, Shrideep Pallichara,Geoffrey Fox, "Towards an Architecture for Audio/Video Conferencing in Distributed Brokering System", 2003