SECURED NETWORK CONFERENCING

Thesis submitted in partial fulfillment of the requirements for the award of the Degree of

MASTER OF ENGINEERING IN COMPUTER SCIENCE AND ENGINEERING
OF BHARATHIAR UNIVERSITY

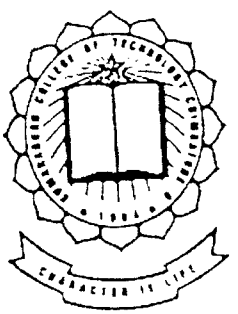
By

K.SOUNDARARAJAN
(Reg.No.0037K0012)

P. 631

Under the Guidance of

Mr. R.DINESH B.Tech., M.S.(Comp. Science, USA)
Senior Lecturer
Dept. of CS&E, KCT



DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

KUMARAGURU COLLEGE OF TECHNOLOGY

(Affiliated to Bharathiar University)

COIMBATORE – 641 006

2000 - 2001

CERTIFICATE

Department of Computer Science and Engineering

Certified that thesis is a bonafide report

of

the thesis work done by

K.SOUNDARARAJAN
(Reg.No.0037K0012)

at

KUMARAGURU COLLEGE OF TECHNOLOGY

COIMBATORE – 641 006

During the year – 2000 – 2001

R. Dinesh 18-12-2001

Guide
Mr.R. DINESH B.Tech., M.S.(Comp. Science,USA)
Department of Computer Science and Engineering
K.C.T., Coimbatore.

Head of the Department
Prof S.THANGASAMY

Place : Coimbatore

Date : 20-12-2001

Submitted for Viva – Voce examination held at
Kumaraguru College of Technology on 20-12-2001

R. Dinesh

Internal examiner (20/12/2001)

External Examiner

CERTIFICATE

This is to certify that this thesis work entitled **"SECURED NETWORK CONFERENCING"** being submitted by **K.SOUNDARARAJAN**, (Reg.No.0037K0012) for the award of degree of **MASTER OF ENGINEERING IN COMPUTER SCIENCE & Engg.** is a bonafide work carried under my guidance. The results embodied in this thesis have not been submitted to any other university or institute for the award of any degree or diploma.

R. J Dinesh 18/12/2

Mr.R.DINESH B.Tech., M.S.(Comp. Science, USA)

Senior Lecturer

Department of Computer Science and Engineering

Kumaraguru College of Technology

Coimbatore.

ACKNOWLEDGEMENT

It is the duty of the author to express his deep sense of gratitude to his **Parents** whose support and encouragement made to do this course **M.E., (COMPUTER SCIENCE&ENGINEERING)** in this prestigious institution.

The author wishes to take this opportunity to offer a respectful note of thanks to **Dr. K.K.PADMANABAN, Ph.D.,** principal of the college, for the excellent facilities made available to accomplish this project work.

The author would like to deem it a privilege to record his sincere thanks to **Prof.S.THANGASAMY, Ph.D.,** Head of the Department of Computer Science Engineering for his valuable suggestions and motivations during the entire period of this course.

The student would like to express his heartfelt gratitude to his guide **Mr.R.DINESH,B.Tech.,M.S.** (Comp. Science, USA) for his valuable guidance, suggestions, and consistent encouragement, which lead to the successful completion of the project.

K.SOUNDARARAJAN

SYNOPSIS

The project "Secured Network Conferencing" aims to perform Network conferencing. The main objective is to conduct a conference and the conference members can discuss the issues on the topic of the day and try to find the solution for the topic. The provision of security in conferencing will ensure that the conference message not be able to read by others. Security in the conference enable the conferencing forum to be secure, no unauthorized user can enter into the conference hall and hear to the forum.

The essence of a text conferencing system is that it must allow users to read text messages in a forum and add their views to the forum. A forum is a place where people come together to discuss on a topic. Many variations on this basic theme are possible such as text with graphics, structured dialogs, and gateways to other conferencing systems.

Audio conferencing in turn allow the participants to discuss on a topic which is similar to a conference hall but with a restriction-the facial expression of the participants will not be visible as they are using computers to exchange views. There will be a central convenor who organise the proceedings of the day.

The organiser of the conference starts the session and the members of the conference body join the meeting which is connected to the conference server. The registration form allows a member to access the server and can able to use the text conferencing or audio conferencing. Several controls are available to access the situation at any instant of the day.

Security in conferencing system involves authentication, privacy and preservation of the message. Cryptography is used in conferencing where encryption is being done at transmitter end and decryption is done at receiving end. Classic cipher algorithm is used.

The member register himself to the conference server with his ID. Separate GUI window is available for all users who are available in the conference. After registration key will be supplied to the user for encryption. The proceedings will be maintained as a log file.

CONTENTS

INTRODUCTION

1.INTRODUCTION

Computer network conferencing is just now starting to grow and take advantage of the modern technology that is available. Although there are some systems which have been around for some time (BRC - Bitnet Relay Chat and IRC - Internet Relay Chat), there has not been any real move to bring them together under a single protocol. This has led to various protocols and different systems coming to life. As these different systems continue to pop up, it is becoming more obvious that there is need of a standard in this area for developers to follow without the need of worrying about protocol clashes.

In any implementation of a conferencing program, there are likely to be two main components:

(1) a client program or interface which users enter commands into (hereafter referred to as a "client") and

2) a server program which acts as a multiplexor for various clients which connect to it. There are other expectations and requirements for both servers and clients which are mentioned in more detail later.

Secured Network conferencing is the conferencing system that has important features like security in the messages that are transferred and also that minimum affort is needed for the user to connect to the conference server. The user are in no need of knowing the server Ip address and the port number to which he will be communicating. In audio the links for the audio controls are set up when the user go for the audio conferencing. In secured Network conferencing an user can able to choose text or audio or both conferencing methodologies.

1.1 The current status of the problem taken up

Secured Network Conferencing is a conferencing system which is accomplished with text and audio modes. A typical conferencing system will contain a room(chat) which will enable the members to talk to the other members.

Text conferencing and audio conferencing are done in different environments as the have different controls to change from on form to another. But in secured network conferencing both the text as well as the audio can be used simultaneously almost. This conferencing system also finds the conferencing server automatically without supplying the servers IP address and the port number.

Conferencing system also enables to check the necessary hardware and point out the error in the hardware and the underlying software problems as exceptions. The provision of maintaining a log file will be helpful for the others what have been discussed when he joined the meeting late.

1.2 Relevance and importance of the topic

The project "Secured Network Conferencing " is the study and implementation of Network conferencing. The main objective is to conduct a conference and the conference members can discuss the issues on the topic of the day and try to find the solution for the topic. The provision of security in conferencing will ensure that the conference message not be able to read by others. Security in the conference enable the conferencing forum to be secure, no unauthorized user can enter into the conference hall and hear to the forum.

The essence of a text conferencing system is that it must allow users to read text messages in a forum and add their views to the forum. A forum is a place where people come together to discuss on a topic. Many variations on this basic theme are possible such as text with graphics, structured dialogs, and gateways to other conferencing systems.

Audio conferencing in turn allow the participants to discuss on a topic which is similar to a conference hall but with a restriction-the facial expression of the participants will not be visible as they are using computers to exchange views. There will be a central convenor who organise the proceedings of the day

The organiser of the conference starts the session and the members of the conference body join the meeting which is connected to the conference server. The registration form allows a member to access the server and can able to use the text conferencing or audio conferencing. Several controls are available to access the situation at any instant of the day.

Security in conferencing system involves authentication and privacy. Cryptography is used in conferencing where encryption is being done at transmitter end and decryption is done at receiving end. Classic cipher algorithm is used.The proceedings will be maintained as a log file.

LITERATURE SURVEY

2.Literature survey

2.1 Secure communication[7.1.1]

Secure communication is the most straightforward use of cryptography. Two people may communicate securely by encrypting the messages sent between them. This casn be done in such a way that a third party eavesdropping may never be able to decipher the messages. While secure communications has existed for centuries, the key management problem has prevented it from becoming commonplace. However , the danger of a code breaker who try to recapture the messages by number of trials.

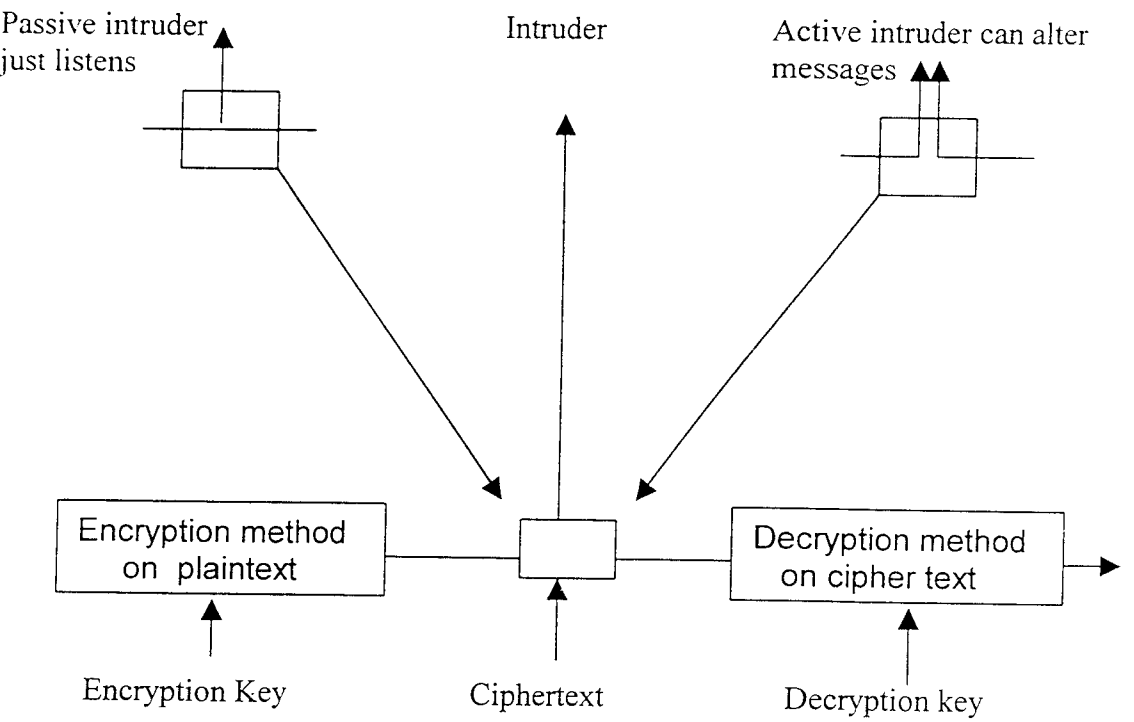


fig. 2a The Cryptography model

The messages to be encrypted, known as the plain text, are transformed by a function that is parametrized by a key. The output of the encryption process, known as the ciphertext or cryptogram, is then transmitted to the destination. We assume that the enemy, or the intruder, hears and accurately copies dowm the complete cipher text. However, unlike the intened recipient, he does not know what the key is and so cannot decrypt the cipher text easily. Fig(2a) Sometimes the intruder cannot only listen to the communication channel(passeive intruder), but can also record messages and play them later, inject his own messages, or modify legitimate messages before they get to the receiver. The art of breaking the ciphers is called Cryptanalysis.

2.1.2 Content preservation

Content preservation is one of the key issue to manage when cryptography is used in the communication. For example., consider I will meet you at 5'o clock is the plain text and it is encrypted to a cipher and transmitted. The receiver gets the message and decrypt the message which looks as I will meet you at 7'o clock. This type of problems can occur at both ends , encryption side as well as in the decryption end.

2.2 Cryptography[7.1.1]

As the field of cryptography has advanced, the dividing lines for what is and what is not cryptography have become blurred. Cryptography today might be summed up as the study of techniques and applications that depend on the existence of difficult problems. *Cryptanalysis* is the study of how to compromise (defeat) cryptographic mechanisms, and *cryptology* (from the Greek *kryptos logos*, meaning ``hidden word'') is the discipline of cryptography and cryptanalysis combined. To most people, cryptography is concerned with keeping communications private. Indeed, the protection of sensitive communications has been the emphasis of cryptography throughout much of its history. However, this is only one part of today's cryptography.

Encryption is the transformation of data into a form that is as close to impossible as possible to read without the appropriate knowledge (a key; see below). Its purpose is to ensure privacy by keeping information hidden from anyone for whom it is not intended, even those who have access to the encrypted data. *Decryption* is the reverse of encryption; it is the transformation of encrypted data back into an intelligible form. Encryption and decryption generally require the use of some secret information, referred to as a *key*.

For some encryption mechanisms, the same key is used for both encryption and decryption; for other mechanisms, the keys used for encryption and decryption are different. Today's cryptography is more than encryption and decryption. *Authentication* is as fundamentally a part of our lives as privacy. We use authentication throughout our everyday lives -- when we sign our name to some document for instance -- and, as we move to a world where our decisions and agreements are communicated electronically, we need to have electronic techniques for providing authentication.

While modern cryptography is growing increasingly diverse, cryptography is fundamentally based on problems that are difficult to solve. A problem may be difficult because its solution requires some secret knowledge, such as decrypting an encrypted message or signing some digital document. The problem may also be hard because it is intrinsically difficult to complete, such as finding a message that produces a given hash value.

2.3 Techniques in Cryptography[7.1.1]

There are two types of cryptosystems: *secret-key* and *public-key cryptography*. In secret-key cryptography, also referred to as symmetric cryptography, the same key is used for both encryption and decryption. The most popular secret-key cryptosystem in use today is the *Data Encryption Standard*.

In public-key cryptography, each user has a *public key* and a *private key*. The public key is made public while the private key remains secret. Encryption is performed with the public key while decryption is done with the private key.

2.3.1 Cryptography in secure communication

Cryptography is extremely useful; there is a multitude of applications, many of which are currently in use. A typical application of cryptography is a system built out of the basic techniques. Such systems can be of various levels of complexity. Some of the more simple applications are secure communication, identification, authentication, and secret sharing. More complicated applications include systems for electronic commerce, certification, secure electronic mail, key recovery, and secure computer access. In general, the less complex the application, the more quickly it becomes a reality. Identification and authentication schemes exist widely, while electronic commerce systems are just beginning to be established. However, there are exceptions to this rule: namely, the adoption rate may depend on the level of demand.

2.3.2 Stream Cipher

A stream cipher is a type of symmetric encryption algorithm. Stream ciphers can be designed to be exceptionally fast, much faster than any block cipher. While block ciphers operate on large blocks of data, stream ciphers typically operate on smaller units of plaintext, usually bits. The encryption of any particular plaintext with a block cipher will result in the same ciphertext when the same key is used. With a stream cipher, the transformation of these smaller plaintext units will vary, depending on when they are encountered during the encryption process. A stream cipher generates what is called a *keystream* (a sequence of bits used as a key). Encryption is accomplished by combining the keystream with the plaintext.

The generation of the keystream can be independent of the plaintext and ciphertext, yielding what is termed a *synchronous* stream cipher, or it can depend on the data and its encryption, in which case the stream cipher is said to be *self-synchronizing*.

2.3.3 Key Recovery

Key recovery is a technology that allows a key to be revealed under certain circumstances without the owner of the key revealing it. This is useful for two main reasons: first of all, if a user loses or accidentally deletes his or her key, key recovery could prevent a disaster. Secondly, if a law enforcement agency wishes to eavesdrop on a suspected criminal without the suspect's knowledge (akin to a wiretap), the agency must be able to recover the key. Key recovery techniques are in use in some instances; however, the use of key recovery as a law enforcement technique is somewhat controversial.

2.3.4 Cryptanalysis

Cryptanalysis is the flip-side of cryptography: it is the science of cracking codes, ecoding secrets, violating authentication schemes, and in general, breaking ryptographic protocols. In order to design a robust encryption algorithm or ryptographic protocol, one should use cryptanalysis to find and correct any weaknesses. his is precisely the reason why the most trusted encryption algorithms are ones that have een made available to public scrutiny. For example, DES has been exposed to public rutiny for years, and has therefore been well-trusted, while Skipjack was secret for a ng time and is less well-trusted. It is a basic tenet of cryptology that the security of an gorithm should not rely on its secrecy. Inevitably, the algorithm will be discovered and weaknesses (if any) will be exploited. The various techniques in cryptanalysis empting to compromise cryptosystems are referred to as attacks. Some attacks are neral, whereas others apply only to certain types of cryptosystems.

.5 Classic cipher Algorithm for Text Conferencing

Classic cipher algorithm is the Symmetric key cryptographic algorithm in which ccessful in block of messages. Since in text conferencing the users message will be a

block and it should be encrypted. So the Classic cipher algorithm will be quite useful in the realtime message transfers like conferencing.

The Algorithm is illustrated as follows.

Plain text	a	b	C	d	..	..			w	x	y	z
Cipher	Q	W	E	R					V	B	N	M

For example.,

If the plain text is attack then the encrypted message will be QZZQEA. This is mostly a monoalphabetic substitution, with the key being the 26 letter string corresponding to the full alphabet. This is a safer system when a block of messages transfers very quickly in the conferencing system, the intruder should make 26! Combinations to make out the correct message on a real time environments. When the integers are used along with the alphabets then it will again take about 10! combinations.

By using the Classic cipher for some words this will be hacked but on using it in block of message transfers this algorithm will be quite suitable.

2.3.6 Classic cipher algorithm for Audio Conferencing.

The analog audio signal is fed into the microphone and PCM encoding is done to convert the analog signal into a digital signal. On the basis of the sampling bit size the digital code is converted to the corresponding alphabet and then the classic cipher algorithm is applied as in the text conferencing. This process is usually done in the transmitter end. When the encrypted message arrives at the destination decryption is done and then it is converted to the audio code by the reverse process.

2.4Network Conferencing[7.2.2]

Meetings which are carried out over the netwok using a special software running on a remote machine. Communication among participants is reffered to as Network

conferencing. There are basically 2 types of network conferencing are available namely text and audio.

Conferences today are an integral part of the business world in many ways. A conference may be held to reassure staff about company problems or may be held by a few directions in an emergency situation where a carefully considered solution is needed. There will be a group of 2 or more, where each speaks and listens to others.

2.4.1 Talk/phone Vs Conferencing

To provide instantaneous communication between two users on Unix and other multiuser systems, interprocess communication is commonly used either over a network or other local methods. The diversity of Unix platforms has introduced as many problems as the presence of various operating systems on the network. Commonly, those on Unix based machines are unable to talk to those on VMS or VM machines. The occasion even arises where two Unix hosts are unable to talk to each other due to different talk protocols.

2.4.2 Textual Conferencing

When the communication is taking place between the participants through the text of messages is Textual conferencing. Everything that is typed on a keyboard is transmitted to the machines of all participants. The participants can share their views on the particular topic of discussion.

In text conferencing the participant types the messages what he wants to send to others in the conference through the keyboard. The user can also block the participants whomever he wants.

2.4.3 Audio Conferencing

When the speech of the participant is sent to the members of the conference then it is audio conferencing. Normally what we speak is analog signal, therefore it should be converted to digital code for the computer to understand. PCM encoding scheme is used to convert the analog audio signal.

The digital audio signal is then converted to the format which will support the underlying network protocol. At the destination the digital audio is captured and playback is completed by the speaker or head phone available to the destined participant.

2.4.4 User Privacy

Members of a conference may wish to exchange ideas privately without fear of others eavesdropping or interrupting the current conference. To facilitate this, there should be some support by the product to pass messages from one user/client directly to another. It is also reasonable for a user to want to be able to hide in one way or another from other users, effectively making themselves invisible to other users.

2.4.5 Message delivery

In routing between conference servers, the problem of routing messages is an important issue. If there was a server for the conference at each domain, this wouldn't be an issue, one could simply do some sort of lookup and find the server for it. This is not the case and unless such a server becomes a standard item for conference. Thus for a layer on the top of TCP/IP is needed to deliver messages between the servers for the conference.

2.4.6 Responsibilities of conference Servers

A conference server should be able to supply a list of attached users. The attached users should have the option of being able to say whether they wish to show up in all li ts. The server should provide some method to identify any known user and supply details to the person making the query based on the search key given.

Conference servers should not run in such a manner that they deliberately record the private conversations of users which are relying on the server in some way. It might seem that encrypting the messages before transmission to other servers in some way would solve this.

2.4.7 Error reporting

All errors that the server encounters in its running life should one way or another be reported to the operators which are responsible for this. This may include sending messages if an operator is online or logging it to an error message.

2.4.8 Advantages of Network Conferencing

When the meetings are to be conducted in real time then the participants should come from various places over long distances. The network conferencing enables the participants to use their computers connected to the network for the conferencing and makes them participate in the discussion with less effort. When the node is connected to the Conference server then he is a participant of the conference.

2.5 Networks and Multimedia[7.1.3, 7.3.1]

Over the last ten years, client/server computing has had a powerful impact on the way business deal with information technology. Client/server computing has enhanced users' productivity, revolutionized computer working and restructured the computer industry.

Today, another new technology is posed to impact business computing in an equally dramatic way. Networked multimedia computer applications will significantly affect users and network managers and have a tremendous impact on computing and network infrastructures.

2.5.1 Need for Security

When large volumes of data is transmitted through the network there is a severe problem of attacking the data which is sent through the physical media. However efforts are made to conceal the messages which is passed through the communication media.

Networked Multimedia consists of four tiers:

- The top tier consists of the multimedia content providers, such as the news industry, the television industry, and the entertainment industry.

- The second tier constitutes the multimedia application developers. Applications include distant Conferencing, workgroup collaboration and imaging.

- The third tier consists of the multimedia platform builders. Silicon graphics, Sun ,Intel, Apple and other hardware vendors are announcing or delivering the multimedia-capable computers. Of the computers sold in 1995-2001, 90% will be

multimedia capable. However there is a growing demand for the networked multimedia applications.

- The fourth multimedia industry tier is the network infrastructure. There are 2 very different networking environments that will use the multimedia applications. Networking of multimedia will initially occur in business over the private networking infrastructure.

2.5.2 Network Requirements

In general, there are three things that a network must provide in order to be able to support multimedia applications: bandwidth, consistent quality of service(QOS), and multipoint packet delivery.

The amount of bandwidth that an application requires varies greatly depending on the quality of the audio, Frequency and the bit size. The data rate of the telephone quality audio is 64Kbps and for the applications it is 100Kbps. Most of today's PC are on a shared LAN such as Ethernet or a token ring. So there will be around 10 to 20 systems connected for the shared application and use a data rate of 50kbps-100kbps. This bandwidth allows them to transmit the voice recorded over the mic to the clients in the network.

Audio conferencing, can be implemented over the intranet using the available 50-100Kbps bandwidth, as only the telephone quality sound is transmitted to the participants of the conference. However the transmission of the textual message have no limitation of bandwidth requirements.

2.5.3 Bit rate

Real time, interactive applications such as conferencing are sensitive to accumulated delay. For example, telephone networks are engineered to provide less than

400 milliseconds round trip latency. The round trip latency budget is consumed by the sending computer(Encryption), the network, and the receiving computer(Decryption). As a rule of thumb, the sending computer will take a few seconds to send a packet.

The network contributes to latency in several ways, including propagation delay, transmission delay, store and forward delay, and processing delay.

- Propagation delay is the length of time it takes information to travel the distance of the line. This is essentially controlled by the speed of light and is independent of the networking technology used.
- Transmission delay is the length of time it takes to put a packet on the media. Transmission delay is determined by the speed of the media and the size of the packet.
- Store and forward delay is the length of time it takes for an internetworking device such as a switch/bridge/router to receive a packet before it can send it.
- Processing delay consists of steps such as looking up a route and changing a header. When a packet comes in, the networking device(bridge, router or switch)needs to decide which interface it should be sent out on.

Voice communication requires low latency in order to maintain audio quality. Data networks that carry voice traffic is engineered to have low latency below 400ms.

PROPOSED LINE OF ATTACK

3.Proposed line of attack

Secured Network Conferencing aims to conduct a conference and the conference members can discuss the issues on the topic of the day and try to find the solution for the topic. The provision of security in conferencing will ensure that the conference message not be able to read by others. Security in the conference enable the conferencing forum to be secure, no unauthorized user can enter into the conference hall and hear to the forum.

The essence of a text conferencing system is that it must allow users to read text messages in a forum and add their views to the forum. A forum is a place where people come together to discuss on a topic. Many variations on this basic theme are possible such as text with graphics, structured dialogs, and gateways to other conferencing systems.

Audio conferencing in turn allow the participants to discuss on a topic which is similar to a conference hall but with a restriction-the facial expression of the participants will not be visible as they are using computers to exchange views. There will be a central convenor who organise the proceedings of the day

The organiser of the conference starts the session and the members of the conference body join the meeting which is connected to the conference server. The registration form allows a member to access the server and can able to use the text conferencing or audio conferencing. Several controls are available to access the situation at any instant of the day.

Security in conferencing system involves authentication and privacy. Cryptography is used in conferencing where encryption is being done at transmitter end and decryption is done at receiving end. Classic cipher algorithm is used. The proceedings are maintained as a log file.

Secured Network Conferencing is accomplished with text and audio modes. A typical conferencing system will contain a room(chat) which will enable the members to talk to the other members.

Text conferencing and audio conferencing are done in different environments as the have different controls to change from on form to another. But in secured network conferencing both the text as well as the audio can be used simultaneously almost. This conferencing system also finds the conferencing server automatically without supplying the servers IP address and the port number.

Conferencing system also enables to check the necessary hardware and point out the error in the hardware and the underlying software problems as exceptions. The provision of maintaining a log file will be helpful for the others what have been discussed when he joined the meeting late.

DETAILS OF PROPOSED METHODOLOGY

4.Details of proposed methodology

Secured network conferencing provides the conferencing modes by text channel and audio channel. The conference member must first register themselves for using text or audio or both. Audio can only be paired user but multiple user can hold for next chance. Text permit multiple users at any instant of time. The input devices are keyboard and microphone, the output devices are speaker and screen fig 4a.

4.1 Product Perspective

Secured network conferencing is a independent product of other applications. It is implemented in Java1.3. and posses the features supported by java. Various java packages are used namely, io, net, awt, swing. Java is meant for its platform independence and object oriented programming features. The package swing enables the user to have GUI design and various look and feel screens.

4,2 Product functions.

The product includes the functions like register, unregister, call, receive, block, hold and terminate.

1.Starting the NameServer

2.Start the Register form

3.submit the form

4. call a person

5. Accept /Reject a call from other member

6.Select the type of conference you want

7. If text conferencing enter the text in the textbox provided in the Message window ans also read the incoming message. The same textbox can be used as area to sent replies.

8.If audio conferencing, setup the audiolinks and put the hold control on such that the AudioTransmitter and the AudioReceiver are activated. On speech completion the audio links are disabled to increase the bandwidth.

9. Report the status of each user.

10.Terminate the proceedings.

4.2.1 NameServer

NameServer is the conferencing server where the conference is initiated. The main purpose of the conference server is to coordinate all activities initiated by the members of the conference. When the server is started then it will be in running state in order to receive a call from a member. It will create a group where all the users are combined. The nameserver accepts the connection request if he is an authorised user and sends the reply that he is a user of the conference system.

4.2.2 Register

This is a client side program which is first initiated by the user to become a member of the conference. The user is asked for the name and register himself for the conference. If the user is registered already, a message "already registered" is shown.

4.2.3 Unregister

Unregister is initiated by the member when he presses the exit button in the call window. The service is halted for the particular user and is informed to the NameServer. When the username is unregistered he will not be able to continue the communication. The Socket connections will be closed.

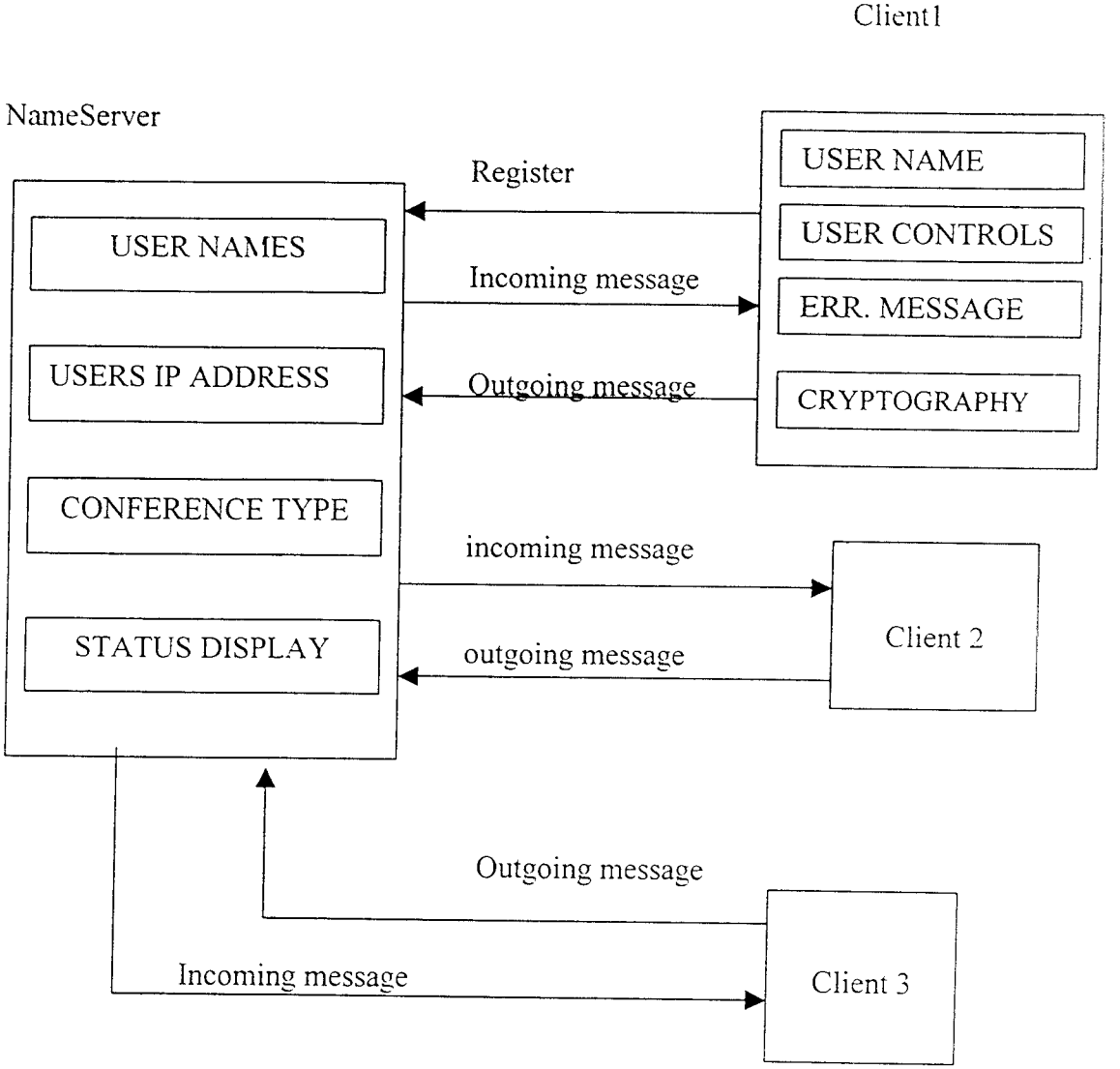


Fig 4a.Block diagram of the conferencing system

4.2.4 Form Submission

When the user completes in supping his name for the conference the form is submitted to the form is submitted to the NameServer inorder to enroll himself as the part of the discussion forum. The user is verified then he is transferred to the call control

center where he can be called or he can call somebody. If there exits another user with the same name the request is turned out and outputs a error message and asks for new name.

4.2.5 Call Blocking

Call blocking is the procedure to block a list of names where a particular user don't want to hear anymore. If a particular user is blocked, the messages sent by him will not be displayed. The user who blocked him has only right to make the blocked user to communicate with him when unblocking is done.

4.2.6 Receive

When a message is received the callers id is shown in front of the message. The message is prompted such that they can talk to each other. Separate window is maintained for showing the textual messages. This receive window is only meant for the textual conferencing only but the audio does not require any window as it is played back in speaker or headphone.

4.2.7 Hold

When hold is initiated by a user it will be displayed in the screens of all users. If the callee is in talk with others show the ID in the corresponding window. Otherwise, both audio and text communication are established.

4.2.8 Terminate

Terminate control can be initiated by any user who want to go out of the conference proceedings. This will remove the user ID in all other users window and gives the message that particular user has left the proceeding.

4.2.9 Active

When communication path is established when the I/O streams are captured by the user, Active status is displayed. The active state implies that the user can use audio or textual communication to the other user.

4.2.10 Caller Establish communication

Get response from callee, check if callee accepts the call, establish communication channel or do nothing according to the checking results.

4.3 Textual conferencing

Exchange textual messages through textual channel, can establish multiple textual channels by one channel per callee. Encryption is done at the sending end and decryption is done at the receiving end when the communication is started. Key maintenance is the key factor and it is initiated by the server. A unique key is available for the particular session.

The participant uses the keyboard as the input device and the monitor as the output device. For the text communication the user should have been registered for the service if he is a right person then he has to call the participant and then a conversation can be invoked.

In the implementation of the text conferencing socket should be created and the datastreams are to be captured. A socket is a communication endpoint where the messages are sent and received. TextArea is provided for the user to type the messages to send for others in the conference. After the user presses enter then encryption algorithm is activated and the cipher text is sent to the participants. After receiving the cipher message decryption is done before displaying the message. The decrypted message is displayed with the user name where it came from actually.

Format to display the received message:

Username>>message

When the message is displayed the participant can tell his views or simply ignore and listen to other members.

4.4 Audio conferencing

Exchange of audio message through audio channel until the channel is terminated by either caller or callee, can establish only one channel among all the callee. Here every phase of the audio conferencing is important and different. When the user selects option to go for audio then the product fetches the information about the audio links. When the necessary hardware is found then audio capture and playback will be opened from the AudioTransmitter and AudioReceiver. For better service not too many audio links are opened. When the talk completes member can come out of the audio control and the callee will be notified and it automatically goes from audio conference to the text mode.

Every activity of the user will be displayed in the conference server with the status of each member in the forum.

4.4.1Audio properties.

The properties of the audio is very necessary in the audio communication when the user goes for audio conferencing. The audio used in Audio conferencing will have PCM signed encoding ,44100HZ,16bits,2channels,4 frames,44100 frame rate ,big endian.

4.4.2 AudioTransmitter

In AudioTransmitter socket connections are established and then the microphone connections are open by setting up the audio links. When the message is delivered the analog signals are captured and then it is converted to the PCM code. The code is

converted into the format which will be supported by the underlying language for the encryption technique and then it is converted into a bytestream. The audio message is then transferred in the form of bytestream in the network.

4.4.3 AudioReceiver

When the message arrives to the receiver then it is decrypted and then its format is changed in order to playback the received message. The received message is played in the headphone or through the speaker. When the user does not want to use the audio the respective control is disabled.

4.5 General constraints

Java 1.3 is required to run this product, with Pentium Serial PC with average speed, multimedia equipments(speaker/headphone and microphone). Network with high transmitting speed will be a minimum requirement.

4.6 Assumptions and dependencies

Assuming too many user to use the text channel to communicate with one person. for this would decrease response speed and quality. Every user must use different IP address.

RESULTS OBTAINED

5.Results Obtained

On executing Secured Network Conferencing program the following results are obtained.

5.1 NameServer

For any client server program there will a server process and the client process. In secured Network conferencing the server process is NameServer and the client program starts with the Registration of the conference member to the Nameserver.

C:\jdk1.3>java NameServer
Checking to see if the NameServer is already installed
Created group InetAddress
Created datagramSocket
Trying to contact the NameServer .
NameServer is not installed
Attemptin to install the NameServer
Created server socket
My IP address is 111.111.111.126
Created group InetAddress
Created multicast socket and joined the group
Created datagramSocket
Waiting for connection
MulticastListener is running
Received new multicast packet
Sent my IP address to the client
Connection received from: MM3
Got I/O streams

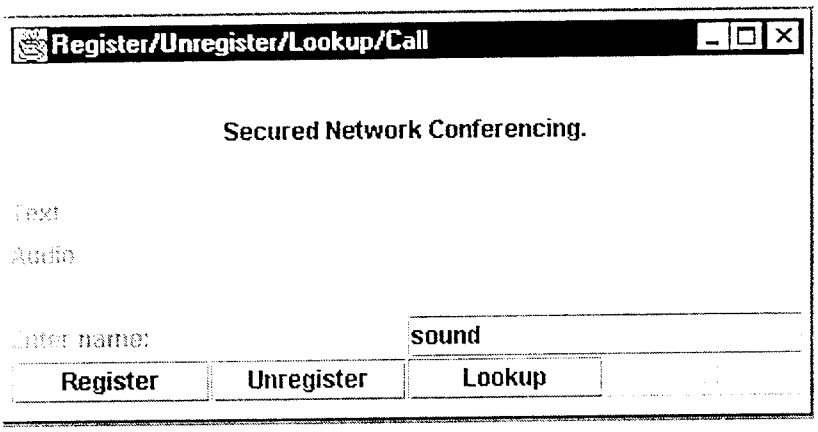
5.2 Register/Unregister/Lookup/Call

Register is the client program through which the members can contact the NameServer. For a valid user it will send the acknowledgement message in a window.

UnRegister is the process through which the user can remove his name when he wants to quit the forum

Lookup is the area through which the user can identify himself and the status of his conferencing.

Call button is to call a person who is in conference for the discussion



5.3 Call Control Center

The status of each user in the conference is displayed in a separate window. The call control center consists of various controls.

The callee name can be given in the textbox where the user can call another member to discuss on the issue.

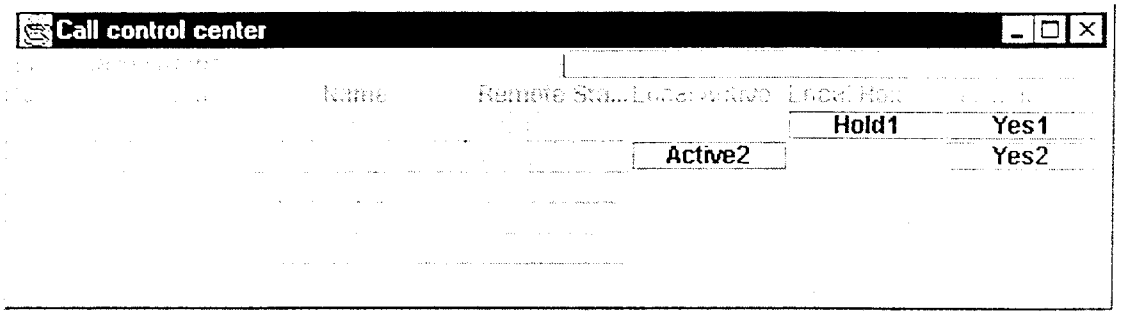
No. represents the total number of participants available in the conference.

Type represents the flow of message. ie to or from .

Name gives the members name who is active in the conference discussion.

Active command will initiate the audio conferencing and hold represents the text conferencing.

When the user wants to leave the conference, the control is Termintate.



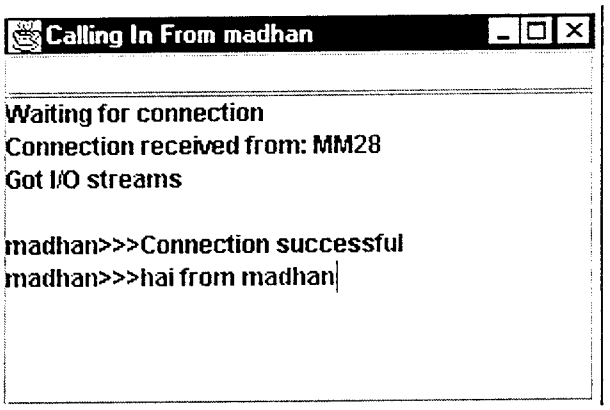
5.4 Calling in from user1

Calling in from the user1 is the window which displays the messages which are coming from the user 1. When the recipient receives the message he can also send the replies to the calling user with the help of the textbox provided at the starting of the calling window.

The window also contains the message that the name of the node which the calling member is attached.

The format for the message to be displayed in the screen is

Caller name>>>received message.



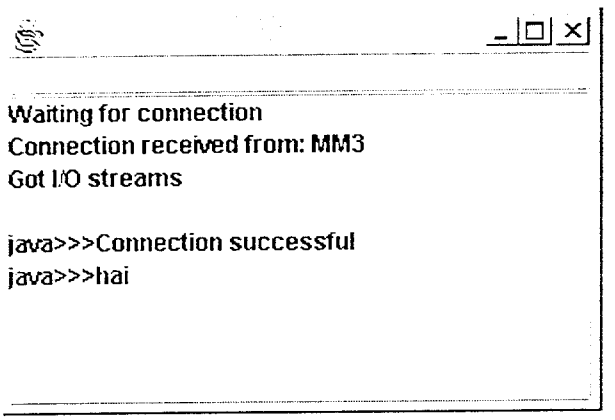
5.5 Calling in from user2

Calling in from the user1 is the window which displays the messages which are coming from the user2. When the recipient receives the message he can also send the replies to the calling user with the help of the textbox provided at the starting of the calling window.

The window also contains the message that the name of the node which the calling member is attached.

The format for the message to be displayed in the screen is

Caller name>>>received message.



CONCLUSION

6.Conclusion

Computer network conferencing is just now starting to grow and take advantage of the modern technology that is available. Although there are some systems which have been around for some time(BRC-Bitnet Relay Chat and IRC- Internet Relay Chat), there has not been any real move to bring the users to communicate with each other. The project also brings down the overhead to choose the IP address of the server and the port address also with the overhead through choosing the hardware setup.

In Secured network conferencing, RFC 1324 is followed for the network conferencing part and for cryptography Classic cipher is used to implement the project successfully.

Future outlook

New cryptographic techniques can be devised and they can be implemented with the conferencing system and the audio can be compressed to increase the data rate. The conferencing system can also be implemented with stenography.

REFERENCES

7.References

7.1 Reference books

1.Andrew S Tanenbaum *Computer Networks*, Printice Hall Thrid Edition 1999

2.William Stallings *Data Communications*, Printice Hall Second edition 1998

3.Sch96] B. Schneier. *Applied Cryptography*. JonWiley & Sons, New York, 2nd edition, 1996.

7.2 Reference journals

1.M.R. Macedonia and D.P. Brutzman. audio and video across the internet. *IEEE Computer*, 27(4):30–36, April 1994.

2. D Reed Network Working Group Request for Comments: 1324 A Discussion on Computer Network Conferencing. RFC No.1324. 1992

7.3 Web sites

1.www.jjtc.com/security/ on 12.08.2001

2.www.java.sun.com on 22.08.2001

APPENDICES

8.Appendices

8.1 Source code

NameServer.java

```
import java.io.*;
import java.net.*;
import java.awt.*;
import java.awt.event.*;
import java.util.*;


public class NameServer
  {
  DataOutputStream socketOutput;
  DataInputStream socketInput;
  DatagramPacket receivePacket, sendPacket;
  MulticastSocket multicastConnection;
  DatagramSocket datagramSocket;
  ServerSocket server;
  Socket connection;
  InetAddress group;
  String myName;
  Vector serviceList, providerList;
  final String locateNameServerRequest = "Need Name Server IP address";

  NameServer() {}

  class MulticastListener extends Thread
    {
    public MulticastListener(String name)
      { super(name);
      }

    public void run()
      {
      System.out.println("MulticastListener is running");
      byte[] buf = new byte[1000];
      receivePacket = new DatagramPacket(buf,buf.length);
      while (true)
        {
        try {
          multicastConnection.receive(receivePacket);
          }
```

```
        catch (IOException ioe)
            {System.out.println("Error in receiving multicast " +
                        "message");
             System.exit(-1);
            }
        System.out.println("Received new multicast packet");

        String clientRequest = new String(receivePacket.getData(),
                        0,receivePacket.getLength());
        if (clientRequest.equalsIgnoreCase(locateNameServerRequest))
          {
          sendPacket = new DatagramPacket(myName.getBytes(),
                        myName.length(),
                        receivePacket.getAddress(),
                        receivePacket.getPort());
          try {
              datagramSocket.send(sendPacket);
             }
          catch (IOException ioe)
              {System.out.println("Error in receiving " +
                  "multicast message");
               System.exit(-1);
              }
          System.out.println("Sent my IP address to the client");
          }
        }
      }
  }


private String addServiceProvider(String serviceName, String providerName)
  {
   if (providerList.contains(providerName))
     return ("Error: Already registered for another service");
   else {
        providerList.addElement(providerName);
        serviceList.addElement(serviceName);
        return ("Registered for this service");
       }
  }

private String removeServiceProvider(String serviceName,
                        String providerName)
  { if (!providerList.contains(providerName))
      return ("Error: Not registered for any service");
    else { int providerIndex = providerList.indexOf(providerName);
```

```
        if (!serviceName.equalsIgnoreCase(
            (String) serviceList.elementAt(providerIndex)))
           return ("Error: Not registered for this service");
        else {providerList.removeElementAt(providerIndex);
            serviceList.removeElementAt(providerIndex);
            return ("Unregistered for this service");
            }
      }
  }

private String getServiceProvider(String serviceName)
  { if (!serviceList.contains(serviceName))
      return ("Error: No such service");
    else {int serviceIndex = serviceList.indexOf(serviceName);
        return ((String) providerList.elementAt(serviceIndex));
       }
  }

public void runNameServer()
  {

   try {
      System.out.println("Attemptin to install the NameServer");
      try {
         server = new ServerSocket(5000,100);
        }
      catch (IOException ioe)
          { System.out.println("Unable to create server socket");
            System.exit(-1);
          }
      System.out.println("Created server socket");

      try {
         myName = InetAddress.getLocalHost().getHostAddress();
        }
      catch (UnknownHostException e)
          {System.out.println("Unable to get IP address of " +
                        "local host");
           System.exit(-1);
          }
      System.out.println("My IP address is " + myName);

      try {
         group = InetAddress.getByName("228.5.6.7");
        }
      catch (UnknownHostException uhe)
```

```
    {System.out.println("Unable to form group 228.5.6.7");
     System.exit(-1);
    }
System.out.println("Created group InetAddress");

try {
    multicastConnection = new MulticastSocket(6789);
    multicastConnection.joinGroup(group);
   }
catch (IOException ioe)
    {System.out.println("Unable to form multicast connection");
     System.exit(-1);
    }
System.out.println("Created multicast socket and joined " +
            "the group");

try {
    datagramSocket = new DatagramSocket(5001);
   }
catch (SocketException se)
    { System.out.println("Unable to create datagramSocket");
      System.exit(-1);
    }
System.out.println("Created datagramSocket");

MulticastListener multicastListener =
    new MulticastListener("Multicast Listener");
multicastListener.start();

serviceList = new Vector();
providerList = new Vector();

while(true)
   {
    System.out.println("Waiting for connection\n");
    connection = server.accept();

    String clientName = connection.getInetAddress().getHostName();
    System.out.println("Connection received from: " + clientName);

    socketOutput =
      new DataOutputStream(connection.getOutputStream());
    socketOutput.flush();
    socketInput =
      new DataInputStream(connection.getInputStream());
    System.out.println("Got I/O streams");
```

```
        String command,serviceName,response;
        do
          {
          command = socketInput.readUTF();
          serviceName = socketInput.readUTF();
          System.out.println("Received command = " + command +
                        ", serviceName = " + serviceName);

          if (command.equalsIgnoreCase("Register"))
            response = addServiceProvider(serviceName,clientName);
          else if (command.equalsIgnoreCase("Unregister"))
                response = removeServiceProvider(serviceName,
                                        clientName);
          else if (command.equalsIgnoreCase("Lookup"))
                response = getServiceProvider(serviceName);
          else if (command.equalsIgnoreCase("CloseConnection"))
                break;
          else response = "Error: Bad command";
          System.out.println("Response is:\n   " + response);

          socketOutput.writeUTF(response);
          socketOutput.flush();
          }
        while (!command.equalsIgnoreCase("CloseConnection"));

        socketOutput.close();
        socketInput.close();
        connection.close();
       }
     }
   catch (Exception e)
       {System.out.println("Bad problem somewhere");
        System.exit(0);
       }

  }


public boolean nameServerIsAlreadyInstalled()
  {
  boolean alreadyInstalled = true;

  try {

     System.out.println("Checking to see if the NameServer is " +
```

```
                "already installed");

try {
    group = InetAddress.getByName("228.5.6.7");
   }
catch (UnknownHostException uhe)
     {System.out.println("Unable to form group 228.5.6.7");
      System.exit(-1);
     }
System.out.println("Created group InetAddress");

try {
    datagramSocket = new DatagramSocket();
   }
catch (SocketException se)
     { System.out.println("Unable to create datagramSocket");
       System.exit(-1);
     }
System.out.println("Created datagramSocket");

sendPacket =
      new DatagramPacket(locateNameServerRequest.getBytes(),
                  locateNameServerRequest.length(),
                  group,6789);
try {
    datagramSocket.send(sendPacket);
   }
catch (IOException ioe)
     {System.out.println("Error in receiving multicast message");
      System.exit(-1);
     }
System.out.println("Trying to contact the NameServer");

byte[] buf = new byte[1000];
receivePacket = new DatagramPacket(buf,buf.length);
datagramSocket.setSoTimeout(5000);
try {
    datagramSocket.receive(receivePacket);
   }
catch (InterruptedIOException iioe)
     {System.out.println("NameServer is not installed");
      alreadyInstalled = false;
     }
catch (IOException ioe)
     {System.out.println("Error in receiving multicast message");
      System.exit(-1);
```

```
            }
        if (alreadyInstalled)
          System.out.println("NameServer is already installed");
        datagramSocket.close();
       }
    catch (Exception e)
        {System.out.println("Bad problem somewhere");
         System.exit(0);
        }

    return (alreadyInstalled);

   } // end nameServerIsAlreadyInstalled()


 public static void main(String args[])
    {
    NameServer app = new NameServer();
    if (!app.nameServerIsAlreadyInstalled())
       app.runNameServer();
    }
}
```

AudioTransmitterr.java

```
import java.io.*;
import java.net.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.sound.sampled.*;

class AudioTransmitter extends Thread
{
   private boolean active;
   private DatagramPacket sendPacket;
   private DatagramSocket outputDatagramSocket;
   private String calleeAddress;

   public AudioTransmitter(String calleeA )
   {
        calleeAddress = calleeA;
        active = true;
        try
        {
        outputDatagramSocket = new DatagramSocket();
      }
      catch (SocketException se)
      {
        System.out.println("Unable to create outputDatagramSocket");
        System.exit(-1);
      }
      System.out.println("Created outputDatagramSocket");
      }

      public void run()
      {
        try
            {
            System.out.println("Opening microphone connection");
            TargetDataLine line;
            AudioFormat.Encoding encoding = AudioFormat.Encoding.PCM_SIGNED;
            //AudioFormat(AudioFormat.Encoding encoding, float sampleRate,
            //          int sampleSizeInBits, int channels, int frameSize,
            //    float frameRate, boolean bigEndian)
```

```
AudioFormat format = new AudioFormat(encoding,44100.16.2.4.44100.true);
//AudioFormat format = new AudioFormat(encoding,16000.16.1.2.
//16000.true);
DataLine.Info info = new DataLine.Info(TargetDataLine.class.format);
if (!AudioSystem.isLineSupported(info))
                {
                System.out.println("Line matching " + info + " not supported.");
                return;
            }
            try
            {
                line = (TargetDataLine) AudioSystem.getLine(info);
                line.open(format, line.getBufferSize());
            }
            catch (LineUnavailableException ex)
            {
                System.out.println("Unable to open the line: " + ex);
                return;
            }
            catch (SecurityException ex)
            {
                System.out.println("Security exception : " + ex.toString());
                return;
            }
            catch (Exception ex)
            {
                System.out.println("Exception : " +ex.toString());
                return;
            }
            System.out.println("Audio Transmitter's Sound connection has been
done");
            int frameSizeInBytes = format.getFrameSize();
            int bufferLengthInBytes = 512*frameSizeInBytes;
            byte data [] = new byte[bufferLengthInBytes];
            line.start();

            while (active)
            {
                int numBytesRead = line.read(data,0,bufferLengthInBytes);
                if (numBytesRead <= 0) break;
                sendPacket = new DatagramPacket(data,data.length,

    InetAddress.getByName(calleeAddress),5050);
                outputDatagramSocket.send(sendPacket);
            }
```

```
                    line.stop();
                    line.close();
                    line = null;
                }
                catch (EOFException eof)
                {
                    System.out.println("Server terminated connection");
                }
                catch (IOException io)
                {
                    io.printStackTrace();
                }
            } // end run

        public void close()
        {
           active = false;
        }

        public void setActive()
        {
           active = true;
        }

} // end class Transmitter
```

AudioReceiver.java

```
import java.io.*;
import java.net.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.sound.sampled.*;

class AudioReceiver extends Thread
{
        private boolean active;
        DatagramPacket receivePacket;
        DatagramSocket inputDatagramSocket;

        AudioReceiver(DatagramSocket input)
        {
                active = true;
                inputDatagramSocket = input;
        }

        public void run()
        {
                        System.out.println("Setting up audio links");
                        try
                        {
                                SourceDataLine line = null;
                                AudioFormat.Encoding encoding
=AudioFormat.Encoding.PCM_SIGNED;
                                //AudioFormat(AudioFormat.Encoding encoding, float
sampleRate,
                                //      int sampleSizeInBits, int channels, int frameSize,
                                //      float frameRate, boolean bigEndian)
                                AudioFormat format = new AudioFormat(encoding,44100,

        16,2,4,44100,true);
                                //AudioFormat format = new
AudioFormat(encoding,16000,
                                //16,1,2,16000,true);
                                System.out.println("Trying to create info");
                                DataLine.Info info =new
DataLine.Info(SourceDataLine.class, format);
                                System.out.println("Format and info created");
```

```
                        if (!AudioSystem.isLineSupported(info))
                        {
                                System.out.println("Line matching " + info + " not
supported");
                                System.exit(0);
                        }
                        try
                        {
                                line = (SourceDataLine)
AudioSystem.getLine(info);
                                line.open(format,20480);
                        }
                        catch (LineUnavailableException ex)
                        {
                                System.out.println("Line is unavailable");
                                System.exit(0);
                        }
                        catch (Exception e)
                        {
                                System.out.println("Error in opening line");
                                System.exit(0);
                        }
                        System.out.println("Audio Receiver Created and opened
line.  " + "Starting playback ...");
                        int bytesPerFrame = format.getFrameSize();
                        int numBytes = 512*bytesPerFrame;
                        byte audioBytes[] = new byte[numBytes];
                        try
                        {
                                int numBytesRead = 0;
                                line.start();
                                try{inputDatagramSocket.setSoTimeout(1000);}
                                catch (Exception e)
                                        {       System.out.println(e+" Here");}

                                  while (active)
                                  {
                                  try{
                                        receivePacket = new
DatagramPacket(audioBytes,numBytes);}
                                        catch (Exception e)
                                        {       System.out.println(e);}

                                        try
                                        {
```

```
        inputDatagramSocket.receive(receivePacket);
                                                numBytesRead =
receivePacket.getLength();

        line.write(audioBytes,0,numBytesRead);
                                            }
                                            catch (SocketException se)
                                            {
                                                System.out.println("Error in reading
packet from inputDatagramSocket");
                                                System.exit(-1);
                                            }
                                            catch (InterruptedIOException iioe)
                                            {       System.out.println(
                                                "error of interrupted io and
audioreceiver is stopped");

        //break;//inputDatagramSocket.close();
                                            }
                                        }

                                    System.out.println("Playback has been completed");
                                    line.drain();
                                    line.stop();
                                    line.close();
                                    line = null;
                                }
                                catch (Exception ex)
                                {
                                    System.out.println("Unexpected error during
playback");
                                }
                            }
                            catch (Exception e)
                            {
                                System.out.println("Bad problem somewhere");
                            }
                    } // end run()

    public void close()
    {
      active = false;
    }

    public void setActive()
```

```
    {
      active = true;
    }

} // end class Receiver



import java.io.*;
import java.net.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.util.*;

public class Control extends JFrame implements Runnable
{
        private JLabel enterLabel, no, type, name, lActive;
   private JLabel lHold, remoteAudioStatus, terminate;
        private JPanel panel[];
        private static JTextField noField[], nameField,
                        ttextField[], names[], remoteAudioNow[];
        private static JButton localActive[], localHold[],terminates[];
        private static Vector typeThread, nameThread, localTextThread,
                        localActiveThread, remoteActiveThread,    addressThread;
        public static boolean hasOutCall = false;
        public static int count = 0;
        public static String calleeName, calleeNames,calleeAddress,callerName;
        public String nameServerAddress;
        private int audioBusyIndex;
        public DatagramSocket inputDatagramSocket;
        public Action audioAction;
        public AudioTransmitter audioOut;
        public AudioReceiver audioIn          ;
   public Blocklist   app1 ;

 public Control(String callerN)
 {
    super( "Call control center" );

    callerName = callerN;
    typeThread = new Vector(1);
    nameThread = new Vector(1);
    localTextThread = new Vector(1);
    localActiveThread = new Vector(1);
    remoteActiveThread = new Vector(1);
```

```
    addressThread = new Vector(1);

    audioBusyIndex = -1;

        try {
         inputDatagramSocket = new DatagramSocket(5050);
        }
      catch (SocketException soe)
      {
        System.out.println("Some I/O error in creating InetAddress for name server in
Control");
                System.exit(-1);
      }

    Container c = getContentPane();
    setLocation(200, 300);
    c.setLayout( new GridLayout (8, 1) );

    app1 = new Blocklist(callerName, "ouc");

    panel = new JPanel[8];
    enterLabel = new JLabel("Enter Callee Name:");
    nameField = new JTextField();
    nameField.addActionListener(
                new ActionListener()
                  { public void actionPerformed(ActionEvent e)
                        {
                        if (e.getActionCommand().equals(callerName))
                        {
                                JOptionPane.showMessageDialog(null,"Loop call is not
permitted");
                                nameField.setText("");
                                return;
                        }
                        for (int i = 0; i < nameThread.size(); i++)
                         if
(e.getActionCommand().equals((String)nameThread.elementAt(i)))
                        {
                                JOptionPane.showMessageDialog(null,"Already in talk
with this user");
                                nameField.setText("");
                                return;
                        }
                        if (app1.isInList(e.getActionCommand()) )
                        {
```

```
                    JOptionPane.showMessageDialog(null,"The name is in the
block list");
                    nameField.setText("");
                }

                else
                {
                if (!hasOutCall)
                {
                    setCalleeName(e.getActionCommand());
                    hasOutCall = true;
                    run();
                }
                else
                    JOptionPane.showMessageDialog(null,
                "To provide better service, only one"
                +" call-out is allowed");
                  nameField.setText("");
                  }}
              }
          );

    panel[0] = new JPanel ();
    panel[0].setLayout( new GridLayout(1,2) );
    panel[0].add(enterLabel);
    panel[0].add(nameField);

    no = new JLabel ("No.");
    type = new JLabel("Type");
    name = new JLabel("Name");
    remoteAudioStatus = new JLabel("Remote Status");
    lActive = new JLabel("Local Active");
    lHold = new JLabel("Local Hold");
    terminate = new JLabel("Terminate");
    panel[1] = new JPanel ();
    panel[1].setLayout( new GridLayout(1,7) );
    panel[1].add(no);
    panel[1].add(type);
    panel[1].add(name);
    panel[1].add(remoteAudioStatus);
    panel[1].add(lActive);
    panel[1].add(lHold);
    panel[1].add(terminate);

    noField = new JTextField[6];
    ttextField = new JTextField[6];
```

```
names = new JTextField[6];
localActive = new JButton[6];
localHold= new JButton[6];
remoteAudioNow= new JTextField[6];
terminates= new JButton[6];

RadioHandler radioHandler = new RadioHandler();
ButtonHandler handler = new ButtonHandler();

for(int i = 2; i<=7; i++)
{
    String x = "" + (i-1);
    noField[i-2] = new JTextField(x);
    ttextField[i-2] = new JTextField();
    names[i-2]= new JTextField();
    remoteAudioNow[i-2] = new JTextField();
    noField[i-2].setEnabled(false);
    ttextField[i-2].setEnabled(false);
    names[i-2].setEnabled(false);
    remoteAudioNow[i-2].setEnabled(false);

    localActive[i-2] = new JButton("Active"+(i-1));
    localActive[i-2].addActionListener(radioHandler);
    localHold[i-2]= new JButton("Hold"+(i-1));
    localHold[i-2].addActionListener(radioHandler);

    terminates[i-2]= new JButton("Yes"+(i-1));
    terminates[i-2].setEnabled(true);
    terminates[i-2].addActionListener(handler);

    panel[i]=new JPanel();
    panel[i].setLayout(new GridLayout(1,7));
    panel[i].add(noField[i-2]);
    panel[i].add(ttextField[i-2]);
    panel[i].add(names[i-2]);
    panel[i].add(remoteAudioNow[i-2]);
    panel[i].add(localActive[i-2]);
    panel[i].add(localHold[i-2]);
    panel[i].add(terminates[i-2]);
}

for (int i = 0; i<8; i++)
{
    c.add(panel[i]);
}
```

```
    addWindowListener(
                new WindowAdapter() {
                        public void windowClosing  (WindowEvent e)
                        {
                                for (int i = 0 ; i<typeThread.size();i++)
                                  if ( (
                (String)typeThread.elementAt(i)).equals("To") )
                ((Transmitter) localTextThread.elementAt(i)).close();
                else
                ((Receiver) localTextThread.elementAt(i)).close();
                Register unregister = new Register(true);

                //System.exit(0);
                        }
                        }
                );

    setSize( 550, 150 );
    show();
}

   private class RadioHandler implements ActionListener
   {
        public void actionPerformed( ActionEvent e)
        {
                int changeIndex = -1 ;
                System.out.println(e.getActionCommand());

                        for (int i = 0; i< typeThread.size(); i++)
                                if (( e.getActionCommand().equals("Active"+(i+1)))
                                 || (e.getActionCommand().equals("Hold"+(i+1))))
                        {
                                changeIndex = i;
                        }

                if (changeIndex == -1)
                {
                        return;
                }

                if (audioBusyIndex != changeIndex && audioBusyIndex != -1
                        && ( e.getActionCommand().equals("Active"+(changeIndex+1))))
                {
                        return;
                }
```

```
            if (e.getActionCommand().equals("Hold"+(changeIndex+1)) )
            {
                    localHold[changeIndex].setEnabled(false);
                    localActive[changeIndex].setEnabled(true);
                    if (((String)typeThread.elementAt(changeIndex)).equals("To"))

    ((Transmitter)localTextThread.elementAt(changeIndex)).sendData(
                            "SYSTEMCOMMAND:Hold");
                    else

    ((Receiver)localTextThread.elementAt(changeIndex)).sendData(
                            "SYSTEMCOMMAND:Hold");

                    if ( audioBusyIndex == changeIndex)
                    {
                            audioAction.close();
                            audioBusyIndex = -1;
                    }
                    localActiveThread.setElementAt("Hold",changeIndex);
            }
            else
            {
                    localHold[changeIndex].setEnabled(true);
                    localActive[changeIndex].setEnabled(false);
                    if (((String)typeThread.elementAt(changeIndex)).equals("To"))

    ((Transmitter)localTextThread.elementAt(changeIndex)).sendData(
                            "SYSTEMCOMMAND:Active");
                    else

    ((Receiver)localTextThread.elementAt(changeIndex)).sendData(
                            "SYSTEMCOMMAND:Active");


                    localActiveThread.setElementAt("Active",changeIndex);
                    if (remoteActiveThread.elementAt(changeIndex).equals("Active"))
                    {
                                    Action oneAct = new Action(changeIndex,
                                            (String)
addressThread.elementAt(changeIndex));
                                    audioBusyIndex = changeIndex;
                                    oneAct.start();
                            }
            }
            refresh();
    }
```

```
}

private class ButtonHandler implements ActionListener
{
    public void actionPerformed( ActionEvent e)
    {
        int removeIndex = -1 ;
        System.out.println(e.getActionCommand());

        for (int i = 0; i< typeThread.size(); i++)
            if (e.getActionCommand().equals("Yes"+(i+1)))
            {
            removeIndex = i;
            }

        if (removeIndex == -1)
        {
            return;
        }

        if ( (
        (String)typeThread.elementAt(removeIndex)).equals("To") )
        {
            ((Transmitter) localTextThread.elementAt(removeIndex)).close();
            hasOutCall = false;

        }
        else
        {
            ((Receiver) localTextThread.elementAt(removeIndex)).close();

        }
    }
}

    public void remove(String removeType, String removeName,
                        JFrame threadName)
    {
        int removeIndex = -1 ;

            for (int i = 0; i< typeThread.size(); i++)
            if ((
            (String)typeThread.elementAt(i)).equals(removeType)
            && (
            (String)nameThread.elementAt(i)).equals(removeName)
            && (
```

```
                (JFrame)localTextThread.elementAt(i)).equals(threadName))
                {
                        removeIndex = i;
                        if (removeType.equals("To"))
                                hasOutCall = false;
                }

        if (removeIndex == -1)
        {
                return;
        }

        if ( audioBusyIndex == removeIndex)
        {
                audioAction.close();
                audioBusyIndex = -1;
        }



        typeThread.removeElementAt(removeIndex);
        nameThread.removeElementAt(removeIndex);
        localTextThread.removeElementAt(removeIndex);
        localActiveThread.removeElementAt(removeIndex);
        remoteActiveThread.removeElementAt(removeIndex);
        addressThread.removeElementAt(removeIndex);

        refresh();
}

public static void addCount()
{
        count++;
}
public static void minusCount()
{
        count--;
}

public static int getCount()
{
        return nameThread.size();
}

public static void setType(String callType)
{
```

```
        typeThread.addElement(callType);
}

public static void setThreadName(String callName)
{
        nameThread.addElement(callName);
}

public static void setThread(JFrame threadName)
{
        localTextThread.addElement(threadName);
}

public static void setLocalActiveThread(String audioA)
{
        String temp = "" + audioA;
        localActiveThread.addElement(temp);
}

public static void setRemoteActiveThread(String audioA)
{
        String temp = "" + audioA;
        remoteActiveThread.addElement(temp);
}

public static void setAddressThread(String address)
{
        addressThread.addElement(address);
}

public static void setCalleeName(String name)
{
        calleeName = name;
}

public int getAudioBusyIndex()
{
        return audioBusyIndex;
}

public String getAudioActive(String calleeName, String type)
{
        int index = -1;

        for (int i = 0 ; i<typeThread.size();i++)
        {
```

```
            if ( ((String)typeThread.elementAt(i)).equals(type) &&
                 ((String)nameThread.elementAt(i)).equals(calleeName))
                 index = i;
          }
          System.out.println("The operating index is "+index);
          System.out.print("\blocal active status is"
+(String)localActiveThread.elementAt(index));
          return ((String)localActiveThread.elementAt(index));
    }

    public void setAHT(String calleeN, String calleeAddress,
                 String type, String aht)
    {
          int index = -1;
          for (int i = 0 ; i<typeThread.size();i++)
            if ( ((String)typeThread.elementAt(i)).equals(type) &&
                 ((String)nameThread.elementAt(i)).equals(calleeN))
            index = i;

          if (index == -1 ) return;

          if (aht.equals("Active"))
          {
                 remoteActiveThread.setElementAt("Active", index);
                 remoteAudioNow[index].setText("Active");
                 Action oneAct = new Action(index, calleeAddress);
                 oneAct.start();
                 return;
          }

          if (aht.equals("Hold"))
          {
                 remoteActiveThread.setElementAt("Hold", index);
                 remoteAudioNow[index].setText("Hold");
                 Action oneAct = new Action(index, calleeAddress);
                 oneAct.start();
                 return;
          }

          if (aht.equals("Terminate"))
          {
                 terminate(index);
                 return;
          }
    }
```

```
class Action extends Thread
{
        int index;
        String calleeAddress;

        Action (int index, String calleeAddress)
        {       this.index = index;
                this.calleeAddress = calleeAddress;
        }

        public void run()
        {
                if (audioBusyIndex == index)
                {
                        audioBusyIndex = -1;
                        hasOutCall = false;
                        close();
                }

                if( ((String)localActiveThread.elementAt(index)).equals("Active")
                 && ((String)remoteActiveThread.elementAt(index)).equals("Active") )
                        {
                                audioBusyIndex = index;
                                System.out.println("AudioBusyIndex is " +
audioBusyIndex + "----------------------------");
                                audioAction = this;
                                audioOut = new AudioTransmitter(calleeAddress );
                        audioIn = new AudioReceiver(inputDatagramSocket) ;
                                count ++;
                                audioOut.start();
                                audioIn.start();
                        }
        }

        public void close()
        {
                audioIn.close();
                audioOut.close();
        }
}

        public void terminate(int index)
        {
                if( ((String)localActiveThread.elementAt(index)).equals("Active")
                 &&((String)remoteActiveThread.elementAt(index)).equals("Active") )
                 {
```

```
                audioAction.close();
                System.out.println("reseting the audiobusyindex to -1");
                audioBusyIndex = -1;
        }

        if ( (
        (String)typeThread.elementAt(index)).equals("To") )
        {
                hasOutCall = false;
        }

        typeThread.removeElementAt(index);
        nameThread.removeElementAt(index);
        localTextThread.removeElementAt(index);
        localActiveThread.removeElementAt(index);
        remoteActiveThread.removeElementAt(index);
        addressThread.removeElementAt(index);

        refresh();
}

public void setValues(String nameServerA, String calleeN,
String calleeA, String callerN)
{
        nameServerAddress = nameServerA;
        calleeNames = calleeN;
        calleeAddress = calleeA;
        callerName = callerN;
}

public void refresh()
{
        System.out.println("\nNow is refreshing\nCount is "+ nameThread.size()+
                "the size of vector is "+typeThread.size());
        int index=0;

        for( int i = 0 ; i<6; i++)
        {
                ttextField[i].setText("");
                names[i].setText("");
                remoteAudioNow[i].setText("");
                localActive[i].setEnabled(false);
                localHold[i].setEnabled(false);
                terminates[i].setEnabled(false);
        }
```

```
int k = 0;
while(k< 6 && k<typeThread.size())
{
        if (localTextThread.elementAt(k) == null)
        {
                typeThread.removeElementAt(k);
                nameThread.removeElementAt(k);
                localTextThread.removeElementAt(k);

                localActiveThread.removeElementAt(k);
                remoteActiveThread.removeElementAt(k);
                addressThread.removeElementAt(k);
        }
        else
        k++;
}

if( !typeThread.isEmpty())
{
        for (int i = 0; i< typeThread.size(); i++)
        {
        ttextField[i].setText((String)typeThread.elementAt(i));
        names[i].setText((String)nameThread.elementAt(i));

        if (((String) localActiveThread.elementAt(i)).equals("Active"))
        {
                localActive[i].setEnabled(false);
                localHold[i].setEnabled(true);
        }
        else
        {
                localActive[i].setEnabled(true);
                localHold[i].setEnabled(false);
        }

        if (((String) remoteActiveThread.elementAt(i)).equals("Active"))
                remoteAudioNow[i].setText("Active");
        else
                remoteAudioNow[i].setText("Hold");

        terminates[i].setEnabled(true);
        }
}
this.repaint();

for (int i = 0; i< typeThread.size(); i++)
```

```
                {System.out.println("local type status is "
+(String)typeThread.elementAt(i));
                }
                for (int i = 0; i< nameThread.size(); i++)
                {System.out.println("local name status is "
+(String)nameThread.elementAt(i));
                        }
                for (int i = 0; i< localTextThread.size(); i++)
                {System.out.println("local text status is " +i);
                        }
                for (int i = 0; i< remoteActiveThread.size(); i++)
                {System.out.println("remote active status is "
+(String)remoteActiveThread.elementAt(i));
                        }
                for (int i = 0; i< localActiveThread.size(); i++)
                {System.out.println("local active status is "
+(String)localActiveThread.elementAt(i));
                        }
                for (int i = 0; i< addressThread.size(); i++)
                {System.out.println("address status is"
+(String)addressThread.elementAt(i));
                        }

                System.out.println("The end of refresing\n");
                }

        public void run()
        {
        Lch newlch = new Lch(this);
        newlch.start();
        System.out.println("calleename is "+ calleeName+calleeAddress+callerName);
        newlch.setValues(nameServerAddress,calleeName, calleeAddress, callerName);
   }
}
```

8.2 RFC 1324

A Discussion on Computer Network Conferencing

Status of this Memo

This memo provides information for the Internet community. It does not specify an Internet standard. Distribution of this memo is unlimited.

Abstract

This memo is intended to make more people aware of the present developments in the Computer Conferencing field as well as put forward ideas on what should be done to formalize this work so that there is a common standard for programmers and others who are involved in this field to work with. It is also the intention of this memo to stimulate the computer community and generate some useful discussion about the merits of this field.

Introduction

Computer network conferencing is just now starting to grow and take advantage of the modern technology that is available. Although there are some systems which have been around for some time (BRC - Bitnet Relay Chat and IRC - Internet Relay Chat), there has not been any real move to bring them together under a single protocol. This has led to various protocols and different systems coming to life. As these different systems continue to pop up, it is becoming more obvious that there is need of a standard in this area for developers to follow without the need of worrying about protocol clashes.

In any implementation of a conferencing program, there are likely to be two main components: (1) a client program or interface which users enter commands into (hereafter referred to as a "client") and 2) a server program which acts as a multiplexor for various clients which connect to it. There are other expectations and requirements for both servers and clients which are mentioned in more detail later.

8.1.0 NETWORK CONFERENCING TODAY

8.1.1 Conferencing in general today

Conferences today are an integral part of the business world in many ways. A conference may be held to reassure staff about company problems (boost moral) or may be held by a few directors in an emergency situation where a carefully considered solution is needed. Conferences also form the cornerstone of workshops held where various groups of people, who attend, are to be briefed on new developments.
In nearly all of these situations, there will be a group of 2 or more, where each speaks and listens to others. There exist PABXs and other features of the telephone system

which provide for conferencing between people around the globe at a cost effective rate. The only place which really lacks any formal form of conferencing is the internet, although many unofficial conferencing systems already exist, spanning the globe or providing local forums.

8.1.2 Talk/phone vs. conferencing

To provide instantaneous communication between two users on unix and other multiuser systems, interprocess communication is commonly used either over a network or other local methods. The diversity of unix platforms has introduced as many problems as the presence of various operating systems on the net. Commonly, those on Unix based machines are unable to talk to those on VMS or VM machines. The occasion even arises where two Unix hosts are unable to talk to each other due to different talk protocols.

8.1.3 Advantages of realtime computer conferencing

By providing a standard for computer conferencing, it should eliminate the problem of who is using what computer. This will mean that someone from a VMS or VM machine can talk with one or more people without having to worry whether their counterpart has an account on a compatible machine for their choice of communication. Electronic mail (email) has already reached this position with most modern mailers on the internet being compliant with RFC822. It is therefore not unreasonable to expect this of realtime conferencing which is to talk as USENet is to email; although of those four (4), only email and news have been covered by RFCs.

USENet is a vast resource and immensely useful for many people around the globe. It does, however suffer from a high noise to signal ratio. It would be unwise to expect much difference in performance from conferencing.

By providing the means for realtime computer conferencing, it opens up a whole new area of usefulness to computers. For both students and staff alike, it opens up new possibilities. In educational institutions where there is a high level of project work with groups of more than 2, it means that students can work from home or other remote places and discuss their project with their fellow students in a manner which would be similar to all students having a conventional meeting or conference. This same situation also applies to staff members. For those who have previously relied on email between fellow researchers in many remote institutions, computer conferencing brings the world together, onto the researchers screen where they can trade ideas and code in real time. Traditionally to achieve these goals, the phone would have been used and a teleconference setup and it will probably remain so for many years to come with video phones too. However, with phone conferencing, when people talk over each other, the quality of the discussion is degraded.

8.2.0 Goals for what a protocol should provide In producing a protocol for conferencing over computer networks, the following problems must be considered:

8.2.1 State Information problems

The number of users who are a part of the conference may fluctuate continuously by a large amount over any given period of time. The protocol should endevour to make disruptions such as these as smooth as possible but at the same time, keep the realtime feel in the conference. It is not acceptable to buffer a user who quits for any given time but at the same time, if a server has network problems with connecting to another one, it may be wise to find some way around the continual stream of state messages that are passed - or - at least a way to reduce the number.

8.2.2 Network barriers

Members of a conference may be on physical networks which cannot directly communicate with each other, such as those used from a host on a commercial network talking via a bridge to someone from a network directly connected to a network directly accessible from theirs. So in this case, the users involved have no need to directly use the bridge (as required by unix talk) since the server on the gateway host provides a way for messages to be passed in and out of the unreachable sections. In this case also, there is a minimum security risk to the network which is otherwise unreachable.

8.2.3 User needs

8.2.3.1 User privacy

Members of a conference may wish to exchange ideas privately without fear of others eavesdropping or interrupting the current conference. To facilitate this, there should be some support by the protocol to pass messages from one user/client directly to another.

It is also reasonable for a user to want to be able to hide in one way or another from other users, effectively making themselves invisible to other users.

8.2.3.2 Realtime Expectations

Users will expect conferencing to be real time, giving the thereby demanding that the protocol supply a quick, efficient, reliable and accurate delivery of a message. Only when these requirements are met can a conference system hope to be of any use to its users.

8.2.4 Message Delivery

8.2.4.1 Deficiencies in using IP only

In routing between conference servers, the problem of routing messages is an important issue. If there was a server for the conference at each domain, this wouldn't be an issue, one could simply do some sort of lookup and find the server for it. This is not the case and unless such a server becomes a standard item for unix machines, it is not reasonable

to expect it to ever be so. Thus the need for a layer on top of TCP/IP is needed to deliver messages between the servers for the conference.

8.2.4.2 Flexibility

The routing protocol used should not be inflexible and should allow for routes to change over time in much the same way as RIP does now. However, there is no need for a special routing protocol such as RIP since this is already part of IP's functionality. Routing information should be updated automatically when the server receives information via that route whether it creates or destroys a route.

8.2.4.3 Building a flexible transport protocol on top of existing ones

If such a conferencing service is built upon TCP/IP, it is therefore possible to build an abstract routing model which has no relation tothe TCP/IP model. However, it is not wise to ignore the presence of either TCP or IP since by integrating them into the protocol, it is easier to use their strengths. If the protocol relies too heavily on TCP/IP features, it will also inherit some of its weaknesses. These maybe taken for granted, but it is worth keeping them in mind when designing a protocol to be both reliable, efficient and useful.

8.2.5 Network Structure

8.2.5.1 Size

The potential userbase of a conferencing system using the internet should not be underestimated. It is therefore desirable that the conferencing system should be as distributed as possible, and as little state information kept as possible. If the IRC network is taken as a guide, with 800 users on 140 servers in some 200 channels, the server was using over 1MB of memory. Due to the nature of conferencing and the server being run as a daemon, this memory was hardly ever swapped out. For this reason, servers should aim to only be authoritive about required users, channels and servers and keep up to date information on these.

There is also no requirement that a global conferencing system be built, although it is an ideal arena to show the strengths of the network. It also goes without saying that it shows up a lot of its weaknesses too.

Any protocol which is developed should operate equally well and efficiently on both a large scale network and on a small scale network.

8.3.0 Usage

If past usage is any guide, then a networ k based conferencing system will be largely used by mostly students. This is not as unreasonable as it may sound since students and student accounts easily form the largest body on the internet. To encourage staff or

other adults into this field, it might be prudent to reduce the amount of noise and interfearance a bored student (or staff member!) can generate.

Realtime conferencing via computer networks is, however, a very attractive toy to many students. It puts them in touch with the world at no extra charge to them. They are able to construct their own character and mask or hide their real self. This is a field which has already been researched and is an interesting topic to pursue.

8.4.0 Setting it up

8.4.1 Installation

The installation and setup of most network utilities/servers is not something that is commonly discussed. It is, however, a point worth considering here after observations made on the setup and installation of systems such as IRC. If the setup is too easy and requires little work, it is not unreasonable to expect students to "install" it in their own accounts to provide themselves and friends with this service. There is little that can really be done about this except to force servers to listen and connect only to a certain priveledged port(s). This need, however, requires root intervention or aid and it is doubtful whether a service such as this should require such steps.

This problem is not often encountered with other network services since they either require large amounts of disk space to be done properly (news) or require the co-operation of other servers before they work in a full serving role (DNS and use of name servers is a good example of this). Of the two, the latter is a good solution if it can be implemented fairly and well.

8.4.2 Controlling growth

Is it possible to reasonably control the growth and connectivity of a large realtime conferencing network? Should it be compared to other facilities such as USENet which is commonly available and very widespread with no real central control over whogetsnews?

8.5.0 Finding the *right* protocol

This section deals with points which are central issues when deciding upon a protocol. There are many points to consider when developing a realtime protocol which is going to provide a service to many users simultaneously.

8.5.1 Name for protocol

Although names such as IRC and ICB have been used in the past to describe the implementation provided, this document is aimed at stimulating a protocol which is much more general and useful than these. A better name would reflect this. Depending on what network it is implemented on, the Network Conferencing Protocol (NCP) or the Internet Conferencing Protocol (ICP) are two suitable names.

8.5.2 Responsibilities of conference servers

8.5.2.1 Message passing

A conferencing server should pass on all messages not destined for itself or its users to the destination as quickly and efficiently as possible. To this end, the server should not be required to do extensive parsing of the incoming message, but rather, look at the header and decide from there whether to send it on in the typical gateway/relay fashion or parse it and pass it to one or more of its users.

8.5.2.2 Who is on?

Any conference server should be able to supply (on request) a list of attached user(s). The attached user(s) should have the option of being able to say whether they wish to show up in such lists.

8.5.2.3 Who is who?

All servers should provide *some* method to identify any known user and supply details to the person making the query based on the search key given.

8.5.2.4 Conference security

Conference servers should not run in such a manner that they deliberately record the private conversation(s) of users which are relying on the server in some way. It might seem that encrypting the message before transmission to other servers in some way would solve this, but this is better left as an option which is implemented in clients and thus leaves it to the users to decide how secure they want their conference to be.

8.5.2.5 Error reporting

All errors that the server encounters in its running life should one way or another be reported to the operator(s) which are responsible for this. This may include sending messages if an operator is online or logging it to an error file.

8.5.2.6 Network Friendliness

It is quite easy for any network based application to "abuse" the network it is running on. Also in a relay situation, it is quite possible that a server will become bogged down trying to keep up with just one connection and reduces the performance on an overall scale to all users relying on it. It is therefore recommended that user connections be subject to some sort of monitoring and flood control to stop them dumping large amounts of spurious data and causing the server to slow down.

The server should also aim to maximise the packet size of all packets written out to the network. Not only does this make the packet/bytes statistics look nice, but also

increases the efficiency of the server by reducing the time it spends in the system state waiting/doing IO operations such as read/write. The cost here is a fractional decrease in the real-time efficiency of the server.

8.5.2.7 To ASCII or not to ASCII

Given that most of the widely used Internet protocols such as SMTP, NNTP and FTP are all based on commands which are given via ASCII strings, there seems no reason why a conferencing protocol should be any different. The gains from going to binary are marginal and debugging/testing is not as easy as with ASCII. However, it is not unreasonable for some part of the protocol to be done in binary.

8.5.2.8 Queries or messages to a server and replies

For implementation of server queries, it is quite acceptable to use ASCII messages which are made up of words. (Any string of characters which doesn't start with a number). Replies should be some sort of numeric. This is a follow on from from 5.2.7 where all of FTP, NNTP and SMTP work in this manner. By reserving numerics *just* for server replies, there can be no confusion about whether the message is going to or from a server.

8.5.3 Responsibilities of clients

This section discusses the obligations of clients which are connected to a conference server.

8.5.3.1 Providing accurate information

Expecting accurate information is foolish, it matters not for most of the internet, but those that we do wish to trace wont give such information. A client is expected to provide accurate and valid information to the server it connects to so that confusion about who is who is not a problem. Optionally, the server may decide to not trust the information from the client and use some authenticatio scheme that is open to it for such.

8.5.3.2 Client as servers

If a client is acting as a server and accepting direct connections from other clients, the client should provide information about users as discussed in 5.2.3. It is not necessary that a client be able to handle complex methods of communication such as channels and their advanced forms, but they should at least provide users with being able to send messages to other users.

An example of this type of program might be Xtv where one or more users can connect to another Xtv client program using Xtv clients. In the case of X windows and perhaps in other areas, one it to ask the destination user to run a program in a similar manner to unix talk.

8.5.4 How complex should the protocol be?

8.5.4.1 User identification

When a user signs onto a system that has an implementation of a conferencing protocol, they are usually asked or given some sort of unique key by which they are later able to be referenced by. In a large system, it may be such that any key which has meaning to the user(s) will not be sufficient and that collisions will occur with such. It is therefore suggested that a server generate an identifier for each new user it has. This identifier must not only be unique in space, but also time. It is not reasonable for the user to ever have to be aware of what this identifier is, it should only be known by servers which *need* to know. A similar system to that used by NNTP/SMTP is a fair implementation of such a scheme.

8.5.4.2 Trees and cycles

Due to the structure of the network being cyclic or forming loops, it is quite natural to want to emulate this within the protocol that is available for users. This has several advantages over trees, mainly the average path between any 2 nodes being shorter. A cyclic structure also poses many problems in getting messages delivered and keeping the connected users and servers up to date. The main problem with using the tree model is that a break in one part of the tree needs to be communicated to all other parts of the tree to keep some sort of realism about it. The problem here is that such communications happen quite often and a lot of bandwidth is needlessly generated. By implementing a protocol which supports a cyclic graph of its connectivity, breakages are less damaging except when it is a leaf or branch that breaks off.

8.5.5 Protocol summary

It is not expected that any protocol that meets the above demands will be either easy to arrive at or easy to implement. Some of the above requirements may seem to be exotic, unnecessary or not worth the effort. After viewing previous conferencing programs and how they work, many short comings can be seen in taking shortcuts.

8.6.0 Security Considerations

Security issues are not discussed in this memo.