**GROUP CHAT APPLICATION WITH ADVANCED FEATURES**

*Akila Santhoshi Vegesna, Nithin Tom George, Subhankar Banerjee*

**Abstract –** Chatting is an essential requirement in the current world. People in one part of the world look to communicate with people in other part of the world through various chat applications. This report will focus on a group chat application, that will enable people in a particular institution like schools, colleges or companies to communicate with their peers. It is a client-server application using Transmission Control Protocol (TCP) architecture that enables the group chat feature, so that people logged into the application can communicate with others using the application. By default, the application broadcasts the messages to all the clients connected to the server. Apart from that, there is the option of a public and private group chat where in, the users can communicate with certain set of people either in private or public. The safety of the messages being sent are ensured using the encryption and decryption followed by compression and decompression techniques. Further implementations include file transfer facility, voice recording and sending with playback option and Emoji characters to show human reactions in an attractive way. Thus in all, the application supports the essential requirements of the user with scope for further improvements that can be made in future.

**I. INTRODUCTION**

In the present world, chatting is an important requirement for all users. People wants to communicate with their friends who are in any part of the world. This report will focus on group chat application, that will ensure that fast communication is established between different users in a particular institution like schools, colleges or even companies. A centralized server will have the primary control in this group chat application. All the clients can connect to the server, using the server’s IP address and port number. Once the clients are connected, they are free to talk to any other client in that network using the Transmission Control Protocol(TCP) method. As TCP is a connection oriented protocol, we need to establish a connection between the client and server programs. We use the reliability feature of the TCP protocol to ensure safe sending of message packets without any loss of packets. In addition to it, messages sent are encrypted and compressed using the AES standard to provide additional safety to the data being sent. Further, we use the broadcast method as a default feature in our application to deliver the messages to the clients. There is also provision for public and private group chats, wherein users can chat with their peers in public group or if you want some privacy, can chat with your peers in private group. More flexibility is there for users in the group chat mode. One to one chat, one to a specific number of users chatting all are enabled in the group chat mode. As a client connected to the centralized server, you have the sole privilege to create as many public and private groups as you desire and transfer messages, files or voice according to the needs.

All types of files can be sent using this application, text, pdf, images, voice, video etc. Byte by byte sending method is used to send the files to the intended recipient. Since our communication mode is TCP and not UDP, there won’t be a case where we have packet loss and data not transferred fully. Another feature of this application is the voice recording facility. A user can record his or her voice and even has the playback option to re-record the voice if not satisfied with the recording. The features of Pulse Code Modulation(PCM) has been used for this functionality. Encoding and Sampling techniques are being used to covert the analog voice signal to digital signal that will enable the sending of voice signal over the TCP channel. To make our application similar to the existing chat applications, we have also included the Emoji characters, that will give the user to use our chat application just as any other chat application. In the section II and III, we will be discussing about Related works and Implementation respectively. In section IV and V, we will be discussing on the Working and the Functionalities respectively. Finally, in the last few sections we will be concentrating on the scope for Future Improvements, Conclusion and References.

**II. RELATEDWORKS**

The design and implementation of the application is done in Java, since it is easy to program in, it precludes the need to deal with low-level memory management and includes powerful libraries for sockets and threads. Implementing a chat server application is one of the most popular network programming projects. Excellent highly configurable applications are available both as open source as well as proprietary software. Some examples of open source chat applications implemented in java are Llama Chat, FreeCS, Chipchat and openCHAT. With little experience in network programming as well as short duration for the project, our intension was not to match or improve the existing implementation but to implement a basic version of our own.

**III. IMPLEMENTATION**

Some key aspects of how the application is developed is discussed below:

**1. Encryption and decryption:-** Encrypting the messages are done by first importing the library functions javax.crypto.Cipher, javax.crypto.NoSuchPaddingException, javax.crypto.spec.IvParameterSpec and javax.crypto.spec.SecretKeySpec. A cryptographic cipher is implemented using crypto.cipher. For forming the secret key SecretkeySpec and AES/CBC/PKCS5Padding transformation is used.

The same cipher text and secret key is used for decryption.

**2. Compression and decompression:-** ZLIB library is imported for having a general-purpose compression and de-compression. ZLIB is part of the PNG standard and not protected by any patents. Deflater is used to compress the message and inflator is used to decompress. The message is stored in a java byte array for this and java.nio.ByteBuffer is imported.

**3. Technology used**:- For both the front end and back end of the Graphical User Interface, Java is used. Java Swing and AWT are used to build the GUI. javax.swing.JFrame is imported for creating the JPanels and menu bar. At first when the client connects the log in window appears which is a LogWindow that inputs username and password as input. Username or password cannot be left empty. If it is then a messagebox showing error pops up. This is implemented using an ActionListener. Connect button is implemented using a JButton. After connecting to the server, this page’s visibility is closed. It directs to a new JFrame. Here the online users are shown in MainChat. A new room is created using RoomCreation. Messages are displayed in a JTextPane. Java.awt.Color, java.util.Random, javax.swing.text.Style, javax.swing.text.StyleConstants, javax.swing.text.StyledDocument are imported to give the messagebox color, size and styles.

**IV.WORKING**

Initially, the server program starts by binding the IP address and port number to its socket. Now the server is ready to receive connections from the client. As soon as we start the client program (the client has to give IP address and port number to which the server is bound for establishing TCP connection), connection is established between the server and the client and the login page is displayed for the user to login. Multiple client connections with the server is established with the help of POSIX threads. Therefore, each client runs a separate thread, during its interaction with the server. The login page requires a valid entry for username and password such that if any of the field is empty, the user will not be allowed to login. If any user tries to login with the same name from multiple clients, then also the user will not be allowed to login as there already exists a user with the same username. On successful entry of username and password, the user will be directed to the chat window page, where the user has all the privileges to chat with people online, send files (text, pdf, image, voice etc), voice record with playback facility and sending the recorded voice to the peers. Once the user is done with the chatting, then he can use the logout option to log the user out of the chat window and get redirected back to the login page.

USER 2

USER 1

USER 4

SERVER 1

USER 3

*Figure: Showing the centralized server with multiple client connections*

A logged in user by default can chat with all the online users using the broadcast facility enabled for this application. Since the application is used in institutions, if there is a global message to be sent to all the people in the institution, he or she can use this feature to communicate with all the people. In this way, the messages communicated are visible to all the users using this application. The messages sent will be encrypted and compressed as mentioned in the previous section and sent over the TCP connection. At the recipient side, the messages will be decompressed and decrypted so that the recipient can view the message. Along with this, the users also have the privilege to do a group chat either in public or in private. The following sections will explain in detail about public and private group chat.

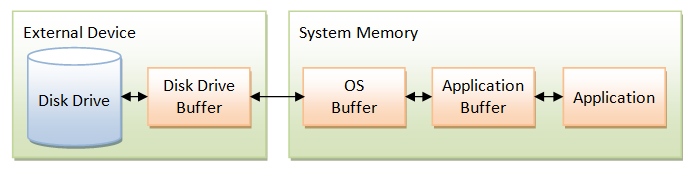
**1. Public Group Chat** :- In Public group chat, the user has the option to create a public group which will be visible to all users online. Any user who wants to join the group can join the group. In this way, there is the option to chat with specific people who have joined the group. At any time, if a user wants to leave the group then it is possible with the help of leave option provided in this application. Messages sent in the public group, is visible to only the users in the public group. Broadcasting technique is also adopted in this chat method. The owner of the public group always has the privilege to close the group when he or she desires. In that scenario, the public group will be deleted from all users who had joined the group initially. Another privilege for the owner is that, he has the right to remove a user from the group. Using this feature, the owner can control who all should be there in the group which he owns.

**2. Private Group Chat :-** In Private group chat, the user has the option to create a private group which only the owner of the group can see. The owner of the group can invite the friends according to his or her desire, so that he can chat with specific people of his choice. In this mode, one to one chat facility is available along with the broadcasting group chat feature to support all types of users. In this mode also, the user has the privilege to remove a user from the group if desired and can chat with specific people according to the liking. The main aim of this feature is to maintain the privacy for the users. At any time, if the owner of the group wants to delete the group, it is made possible by the delete group feature enabled in the application. Deletion of a group by the owner will result in the group getting deleted from all users who are members of the group.

An array list is used to store the details of the public and private rooms whereas a Linked list is used to store the message queues. Message queues are classified into two, one is the system message queue, where all the login, logout, room creation, file transfer etc messages are queued. Next is the text message queue, that will queue all the message communication between the clients. The Graphical User Interface for this has been implemented with the help of the libraries provided by Java Swing.

**V.FUNCTIONALITIES**

**1. File Sending :-** Inorder to perform the advanced Input and Output operation, JDK introduces the so called New I/O or (NIO) in java.nio.Buffer package and its auxiliary packages, to support high performance and intensive I/O operations. NIO is meant to complement the existing Standard I/O (in java.io package), not as a replacement.



Physical I/O operation is thousands times slower than memory access. Hence, a chunk of data is often cache or buffer to improve the throughput. As illustrated from the above diagram, many layers of cache exist between your Java application and physical disk.

* **Disk Buffer**: Disk Buffer is RAM that is built into the disk drive to store a block of data from the disk. The cost of transferring data from the disk surface to the disk buffer is by far the slowest and the most expensive operation, because it involves physical movement of the disk.
* **OS Buffer**: OS Buffer has its own buffering as it can cache more data and manage it more elegantly. This buffer can also be shared among the applications.
* **Application Buffer**: Application may optionally buffer its own data.

NIO data transfer is through the so-called *buffer*s implemented in java.nio.Buffer class. A Buffer is similar to an array, except that it is implemented much more efficiently by closely coupled with the underlying operating system. A Buffer is a contiguous, linear storage. Similar to an array, a Buffer has a fixed capacity.

**Data Transfer (Get/Put):** Each of the primitive buffers provides a set of get() and put() methods to read or write an element or an array of elements from or to the buffer. The position increases by the number of elements transferred.

ByteBuffer class is being used to provide additional getXxx() or putXxx() methods to parse raw bytes into other primitive types. It can also be used as the sources and targets of I/O operations. We are using NIO to read the data into a byte array. Files class provide a static method ‘readAllBytes’, which can be used for this purpose.

**2. Voice Recording with Playback and Sending :-** In voice recording, the user has the option to record his or her voice and then playback the message to test the quality of the recording. This has been done with the help of ‘AudioInputStreamClass’. We have also used methods from ‘AudioSystem’ class to manipulate the Audio Input Stream objects. The voice part of the code is done in a separate thread so that it doesn’t interfere with other functionalities. Once the user is done with the recording, the analog sound signal is converted to digital format(digitized) using the Pulse Code Modulation technique (amplitude of the analog signal is converted to digital value represented by group of pulses). This has been achieved using the encoding method in AudioFormat class, where in the audio signal input is encoded and sampled at a sample rate of 8000 and sample size of 8 to obtain the Pulse Code Modulation output. To make sure that we have recorded the whole audio, getFrameLength method in AudioInputStream class is used so that the encoding and sampling is done on the whole frame. As soon as digitization is done, the recorded audio is saved in a wave file at the sender side.

The playback option has been enabled using the ‘AudioSystem’ class, with the help of ‘getAudioInputStream’ method. When the user clicks on the ‘play’ button, the ‘.wav’ file created from the previous recording will be opened and read byte by byte as output to the user. Since the output is read as byte by byte, it is ensured that audio content has been received completely.

Once the user has recorded the voice and is fully satisfied with the recording after unlimited attempts, the user has the privilege to send this voice message to an online user. ‘File transfer’ method mentioned in the previous section has been made use for this purpose. The recorded ‘.wav’ file will be sent over the TCP connection, reliably to the recipient. The recipient user can either decline or accept the voice message and save it to the system. The file saved to the system should be a wave file in order for the file to be viewed as an audio file. The playback button provided at the recipient side, will enable the user to hear the sent voice message. This works in the same way as mentioned in the sender side, where byte by byte data is read from the file to the output user. Thus the application supports the voice message sending for any user to another online user connected to the same centralized server.

**3. Emoji Characters :-** The user has the privilege to insert Emoji characters into the message he or she sends. The Emoji’s will appear both in the sender side and receiver side. When a user enters an emoticon, that will automatically be converted to the corresponding hexadecimal Unicode. Then using the Unicode package, the respective Emoji’s will be replaced. The following paragraph explains in details more about Emoji’s.

Java.util.Map, java.util.regex.Pattern, java.util.HashMap are imported for the implementation of Emojis. The implementation is done using String myEmojiString = Emoji.replaceInText(myString), where mystring is the message from the user which will be having a symbolic notification of a smiley which when passed through the function gives the output myEmojiString i.e, the output string where they are replaced by corresponding Emojis. The supported emoticon symbols are ":-)", ":)",”:-(", ":(", ":-D", ":D", ";-)", ";)", ":-P", ":P", ":-p", ":p", ":-\*", ":\*", "<3", ":3", ">:[", ":'|", ":-[", ":'(", "=O", "xD", ":')", ":-/", ":/", ":-|", ":|", "\*\_\*". The font of the messages in the terminal had to be changed to "Segoe UI Emoji", so that Unicode Emojis are supported. A negative look-behind ensures that the match is not preceded by "[-\_a-zA-Z0-9)(;:\*<>=/]" and a negative look-ahead ensures that the match is not followed by one of these characters. A regular expression is constructed which ensures that the emoticon is not part of a longer string of special characters e.g. <:)))> or <http://> which both include basic emoticons.

**VI.FUTURE IMPROVEMENTS**

Further improvements can be made to our current application by adding the feature of a voice call and video call, where in the user can chat with his friends who are online. This requires the implementation of voice over IP using the Session Initiation Protocol(SIP). SIP is a protocol for signaling and controlling multimedia communication such as voice and video calls. Since the implementation of this protocol takes time, we will be considering it as a future advancement to our application.

**VII.CONCLUSION**

Thus, we have designed and developed a group chat application implementing the Transmission Control Layer Protocol(TCP) with features such as file sending, voice recording and sending with playback option and Emoji characters. Other features that were implemented were the public and private group chat option which the user has to chat with the peers. We have also ensured the security of the messages by encrypting and compressing the messages being sent over the TCP channel so that the receiver would need to decompress and decrypt to view the message. The GUI have been implemented with Java Swings and AWT, that gave us more functionalities while developing the front end of the application. With the future advancements such as voice and video call facility, our application would replicate the current existing chat applications but in a different manner as it concentrates on group chats.

**VIII.REFERENCES**

1. <http://www.guigarage.com/2015/01/support-emojis/>
2. <http://mrbool.com/file-transfer-between-2-computers-with-java/24516>
3. <http://www.rgagnon.com/javadetails/java-0542.html>
4. <http://www.codejava.net/coding/capture-and-record-sound-into-wav-file-with-java-sound-api>
5. <http://cs.lmu.edu/~ray/notes/javanetexamples/>
6. <http://pirate.shu.edu/~wachsmut/Teaching/CSAS2214/Virtual/Lectures/chat-client-server.html>
7. <http://stackoverflow.com/questions/32871843/clients-server-chat-with-aes-encryption-in-java>