MINOR PROJECT II

Report On

Remote Login Video Conferencing (Acked)

Submitted To: Mrs. Neetu Sardana Mam
Mrs. Arpita Mam

Submitted By:

Ankit Garg (10103623)
Rajat Jain (10103571)
Rajul Kukreja (10103509)
B6 Batch

ACKNOWLEDGEMENT

We take this opportunity to express our profound sense of gratitude and respect to all those who helped us throughout the duration of this project. Preparing a project is never a unilateral effort, its team effort that is responsible for a successful project. We wish to acknowledge the guidance and support of the professors and academics in bringing up a real picture of the concept for which the report is prepared.

We would like to make a special mention of support, help and encouragement we received from our project evaluators **Ms Neetu Sardana**, **Ms Vidhushi Wanchoo, Mrs Arpita mam** and **Mr Buddha Singh**.

Our special thanks to our supervisor **Ms. Neetu Sardana** who was so critical in this project and without him we would not have been able to do this.

We also thank all the staff members of JIIT for extending full support and making this whole experience enriching, informative and facilitating the project to reach its current state.

INDEX

ABSTRACT

To develop an all in one application allowing users within an organization to communicate and help each other without hassle of getting up and go to other side.

We aim to develop an application to let all users to connect once to server and afterwards:

a) **Text Chat** (NOTE: all others require user permission except broadcast)
   a. With all users(broadcast)
   b. With certain user who's online
   c. Make Custom Chat groups(one user can be in more than 1 group at a time, still chat is managed efficiently)

b) **Voice chat**: users are given flexibility to choose their preferred voice input and output device at start from all of available devices on their pc/laptop which is dynamically identified by program.

   They can even test devices for their proper functionalities before chat, afterwards:

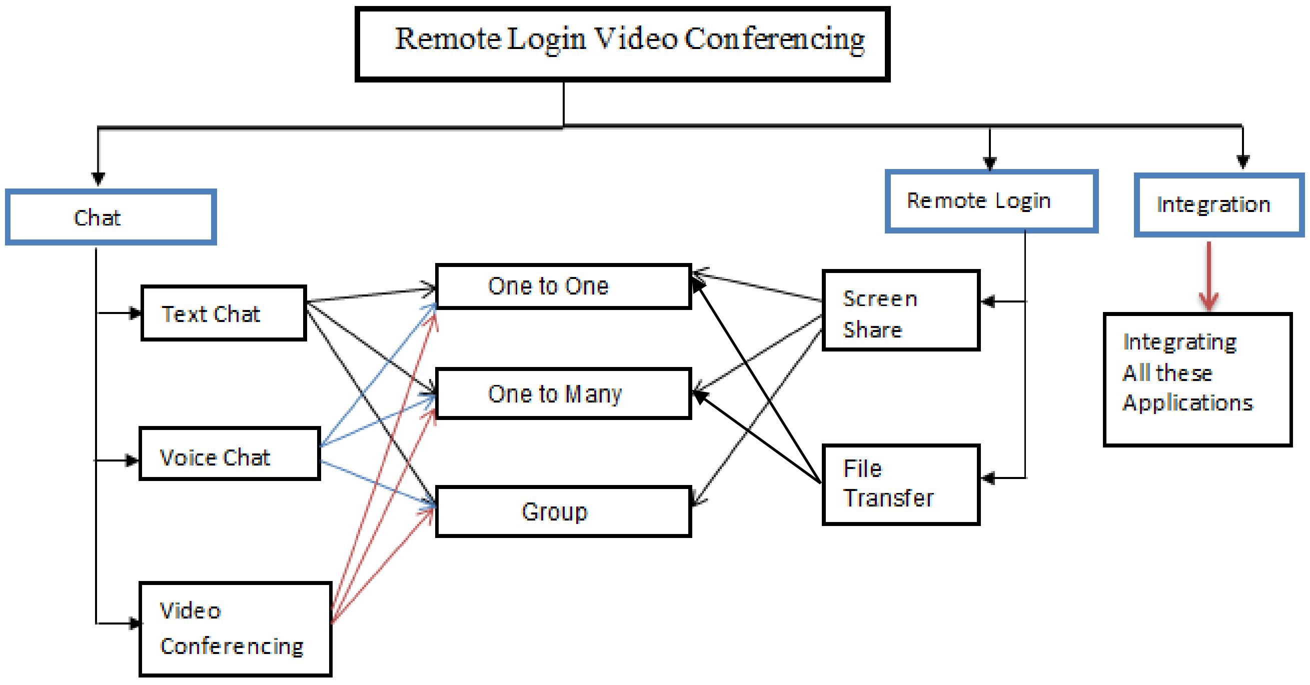
   a. Users can use existing server connection and find someone already online to request them to connect over voice chat while keeping text chat open(all of them).Here server is managing all requests whether they are accepted or rejected and notifying other client of response within certain time.

  b. As some users don’t like to be available online, they can exchange their IP addresses by any other means and setup chat themselves by 2 steps easy configure method.
  c. User can start to listen on their IP address while sending his voice to many users directly. He has to manually add IP address of each of recipients.

c) **File Transfer**: User can choose to send a file less than 5 Mb to selected online users, all other user gets prompt to accept or reject file and file is shared accordingly with option they choose.

d) **Video Conferencing:** As name suggests, a user can conference with multiple clients(up to 5 other),with video streaming in VC-1 format which is directly playable in any media device/player like VLC media player, User can Choose to stream their **webcam image** or **screen** with various size and quality configurable options.

Work Break Down Structure



Division of work among group members

Ankit Garg (10103623, B6)

Voice chat implementation, Interfacing/designing, Video/screen conferencing

Rajat Jain (10103571, B6)

Screen capture implementation, File transfer, Integration of application, Testing

Rajul Kukreja (10103509, B6)

Text chat implementation, Integration of application

Introduction

“We are stuck with technology when what we really want is just stuff that works and works properly”

“First we thought the PC was a calculator. Then we found out how to turn numbers into letters with ASCII — and we thought it was a typewriter. Then we discovered graphics, and we thought it was a television. With the World Wide Web, we've realized it's a brochure.”

Now the technology has so much advanced that we can do whatever we imagine.

So,

Aiming towards improvising and Combining certain application, we target an all in one application, allowing users within an organization to communicate and help each other without hassle of getting up and go to other side.

The Application includes:

1) Text Chat
2) Voice Calling
3) File transfer
4) Screen Share
5) Video Conferencing

All Applications will run over LAN / Wi-Fi

Background Study and Findings

Working on C#

Although C# is derived from the C programming language, it has features such as garbage collection that allow beginners to become proficient in C# more quickly than in C or C++. Similar to Java, it is object-oriented, comes with an extensive class library, and supports exception handling, multiple types of polymorphism, and separation of interfaces from implementations. Those features, combined with its powerful development tools, multi-platform support, and generics, make C# a good choice for many types of software development projects: rapid application development projects, projects implemented by individuals or large or small teams, Internet applications, and projects with strict reliability requirements. Testing frameworks such as NUnit make C# amenable to test-driven development and thus a good language for use with Extreme Programming (XP). Its strong typing helps to prevent many programming errors that are common in weakly typed languages.

Socket Programming in C#

There are several flavours to socket programming - like client side, server side, blocking or synchronous, non-blocking or asynchronous etc. With all these flavours in mind, we decided to work in Client Side, Server side with asynchronous or non-blocking. Reason of taking asynchronous is that Asynchronous is a form of processing that permits other processes to continue before the transmission has finished by sending/receiving in a separate thread.

Network programming in windows is possible with sockets. A socket is like a handle to a file. Socket programming resembles the file IO as does the Serial Communication. You can use sockets programming to have two applications communicate with each other.

The application are typically on the different computers but they can be on same computer. For the two applications to talk to each other, either on the same or different computers using sockets one application is generally a server that keeps listening to the incoming requests and the other application acts as a client or makes the connection to the server application. The server application can either accept or reject the connection. If the server accepts the connection, a dialog can begin with between the client and the server. Once the client is done with whatever it needs to do it can close the connection with the server, during the time client has an active connection it can send the data to the server and/or receive the data.

The complexity begins here. When either side (client or server) sends data the other side is supposed to read the data. But how will the other side know when data has arrived. There are two options - either the application needs to poll for the data at regular intervals or there needs to be some sort of mechanism that would enable application to get notifications and application can read the data at that time. Well, after all **Windows is an event driven system** and the notification system seems an obvious and best choice and it in fact is.

As the two applications that need to communicate with each other need to make a connection first. In order for the two application to make connections the two applications need to identify each other (or each other's computer). Computers on network have a unique identifier called I.P. address which is represented in dot-notation like 10.20.120.127 etc.

Requirement Analysis

1) *Hardware Requirements*

- RAM : Min 256MB
- Windows: XP and above
- Wi-Fi / LAN Connection
- Webcam
- Speaker
- Mic

2) *Software Requirements*

1. Developer End
    - Microsoft Visual Studio 2012
    - .NET Framework 4.0
    - Expression Encode 4 Service Pack 2
2. User End
    - Expression Encode 4 Service Pack 2
    - .NET Framework 4.0

3) Functional Requirements

Functional requirements define what the system or application will do - specifically in the context of an external interaction (with a user, or with another system).

Major functionalities of our project are as follows: -

a) **<u>Group chat</u>**: - Unlike normal text chat, Users are allowed to create/leave groups and view all group members separately.
b) **<u>Text chat will work side by side</u>**: - All groups will work simultaneously without affecting your video or voice chat.
c) **<u>Screen Sharing</u>**: - it's all to make meeting interesting we need to share screen so that we can tell online guests to see what other client is doing or giving remote presentations. Users also have an option to show screen around their mouse only.
d) **<u>File Sending</u>**: - we can share file or small presentations with online guests.
e) **<u>Voice Chat</u>**: - We can chat with any other friend who is online but we can make a voice call to only one at a time.
f) **<u>Video Chat</u>**: - We can make video confencing with meeting head selected guests (max 5 others allowed).
g) **<u>Prompt before task</u>**: - Every user has choice to join any video conference/text chat group/voice call or not. Also other important tasks like receiving file have prompt.
h) **<u>Full Screen</u>**: - if any user wants to see video conference on full screen mode he can easily by double clicking the desired user video.
i) **<u>Individual volume for all clients</u>**: - in video chat, user has option to change volume of individual client, not affecting others volume.

j) **<u>Individual Play/pause for all clients</u>**: - in video chat, user has option to stop receiving feed of individual client, not affecting others.or restart receiving from one already connected to.

k) **<u>Name of all clients displayed below player</u>**: - name of each client is displayed under respective player to easily identify them.

4) Non-Functional Requirements

Non-functional requirements are any requirements that don't describe the system's input/output behaviour. Note that we are still talking about requirements, not implementation details, so just because we're using the phrase "non-functional" doesn't mean that anything is fair game to put in that section.

The most common types of non-functional requirements you'll see relate to system operation (availability, continuity, DR), performance (throughput, latency, storage capacity), and security (authentication, authorization, auditing, privacy).

<u>PERFORMANCE: -</u> there is a no doubt that we are having around 10sec delay to video or screen sharing in broadcasting but it's will be same for all users so that delay will not make any difference to our objective of video connectivity.

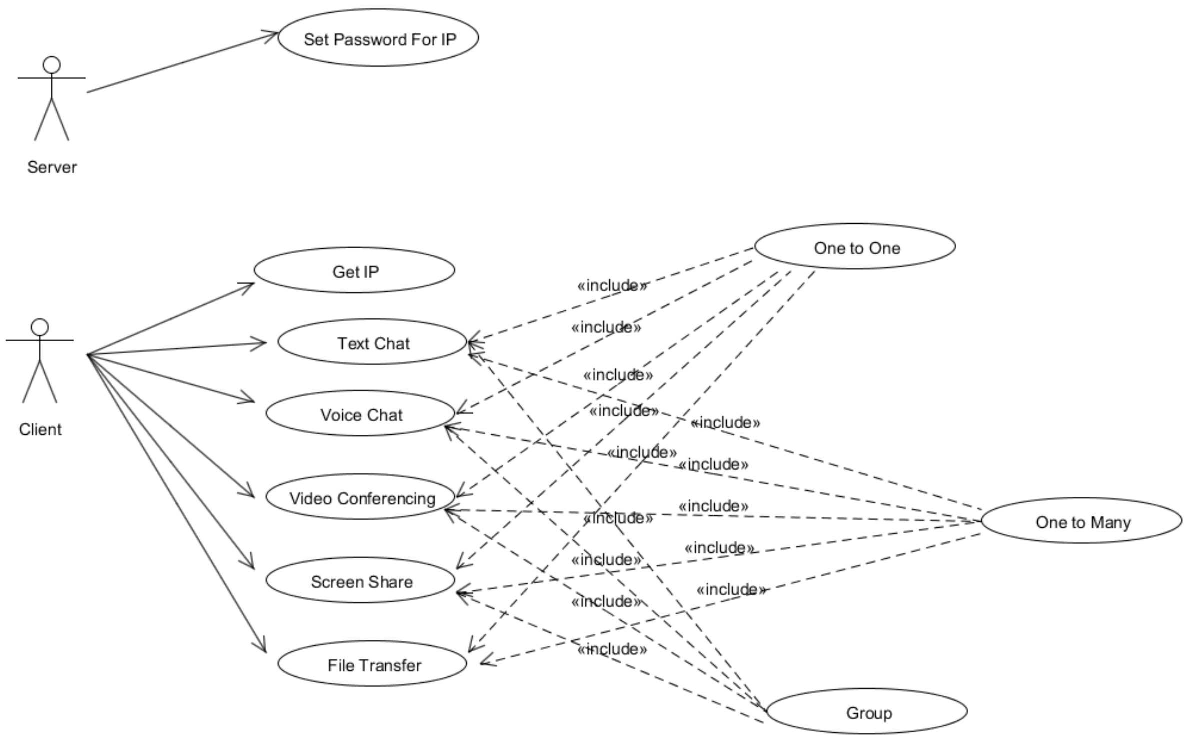
Main thing to note is that voice and video are perfectly synchronised because those 10 seconds are used by processor to encode video in VC-1 video format.

<u>SECURITY: -</u> we use encoding techniques like a-law and u-law for security purposes in voice chat and there is no other way to access them by any other way than our software.

THROUGHPUT: - it's 100% in case of text chat as we are using TCP (connection-oriented) protocol but in case of their slight lose in video and audio quality on transmission because of UDP (connection-less) protocol.

5) UML Diagram

Use case Diagrams



6) User Requirements

The user requirement(s) document (URD) or user requirement(s) specification is a document usually used in software engineering that specifies the requirements the user expects from software to be constructed in a software project.

1) Our basic objective was REMOTE VIDEO CONFERENCING which we have achieved.

2) It’s a user-friendly software, handy to use for any type of user.

3) It's compatible with all windows OS (used by most users).

4) Easily compatible with any hardware present on client computer.

Testing

Successfully running Text chat, File Transfer, Voice calling with good quality of sound and minimal delays and allowing users to encode their voice using either of 2 codecs (alaw/ulaw).Screen Share and Video Conferencing with small delay of 12 seconds with VC-1 Encoding.

The application works on all system successfully that have windows Operating system with .NET Framework 4.0 or above.

The .NET Framework (pronounced *dot net*) is a software framework developed by Microsoft that runs primarily on Microsoft Windows. It includes a large library and provides language interoperability (each language can use code written in other languages) across several programming languages. Programs written for the .NET Framework execute in a software environment (as contrasted to hardware environment), known as the Common Language Runtime (CLR), an application virtual machine that provides services such as security, memory management, and exception handling. The class library and the CLR together constitute the .NET Framework.

.NET Frame work:-

Version: 4.0 and above.

The Application also requires Expression Encoder 4, Service Pack 2.

Black Box Testing

1) Usability Testing

Test Case 1

Test Condition: - New group should be made in new tab.

User Input: - Select members and click group button.

Desired Output: - Group should be made.

Test Result: - Unsuccessful

Error: - Intercommunication between threads occurs.

Correction Made: - Delegate method is used which keep calling parent class until required parent class is found and then thread is created.

Now

Test Result: - Successful

Test Case 2

Test Condition: - Check Protocols

User Input: - Send Protocol for voice chat.

Desired Output: - Voice chat should start.

Test Result: - Unsuccessful

Error: - Can’t distinguish between voice chat and video chat.

Correction Made: - Prefix are given to Text Chat, voice chat, video chat, screen share

Now

Test Result: - Successful

Test Case 3

Test Condition: - Checking File send.

User Input: - User Send File.

Desired Output: - Other User should receive file.

Test Result: - Unsuccessful

Error: - We use synchronous coding so when we send file UI freezes till file is send and received.

Correction Made: - We use Asynchronous coding now

Now

Test Result: - Successful

Test Case 4

Test Condition: - Check Client to Server Message.

User Input: - Client sends Message to Server.

Desired Output: - Should receive uncorrupted message.

Test Result: - Unsuccessful

Error: - Sometime Server is receiving trash messages.

Correction Made: - Automatically resending the message.

Now

Test Result: - Successful

Test Case 5

Test Condition: - Check Voice chat and video chat together.

User Input: - Start video chat than voice chat.

Desired Output: - Should Start.

Test Result: - Unsuccessful

Error: - Client crashes, exclusive permission to audio device required by audio codec.

Correction Made: - At a time only one Permission is given that is you can either do video chat or voice chat

Now

Test Result: - Successful

Test Case 6

Test Condition: - Check Screen Share

User Input: - Start Screen Share (sending picture by picture)

Desired Output: - Should receive sound and images together

Test Result: - Unsuccessful

Error: - Images are received faster than voice

Correction Made: - We are sending after encoding using VC-1 Format so that voice and video are combined in a container before transmitting.

Now

Test Result: - Successful

2) **Performance Testing**

Test Case 1

Test Condition: - The time of entire process to take place.

User Input:-

Desired Output: - The time delay should be less.

Test Result: - Successful

Test Case 2

Test Condition: - The Quality of sound receive in voice chat.

*User Input:-*Transmit voice in voice chat

Desired Output: - Voice received clear and noise free.

Test Result: - Successful

Test Case 3

Test Condition: - The Quality of image and sound received in video/screen

*User Input:-*Transmit images and sound

Desired Output: - Voice received clear, noise free, clear image.

Test Result: - Successful

Test Case 4

Test Condition: - The File Transfer

User Input: - Sending small Files

Desired Output: -Receive File

Test Result: - Successful

Future Scope

FUTURE SCOPE OF APPLICATION:

This application can be easily implemented under various situations. We can add new features as and when we require. Reusability is possible as and when require in this application. There is flexibility in all the modules.

SOFTWARE SCOPE:

- **Reusability**: Reusability is possible as and when require in this application. We can update it next version. Reusable software reduces design, coding and testing cost by amortizing effort over several designs. Reducing the amount of code also simplifies understanding, which increases the likelihood that the code is correct. We follow up both types of reusability: Sharing of newly written code within a project and reuse of previously written code on new projects.
- **Understandability:** A method is understandable if someone other than the creator of the method can understand the code (as well as the creator after a time lapse). We use the method, with small and coherent helps to accomplish this.
- **Cost-effectiveness:** Its cost is under the budget and make within given time period. It is desirable to aim for a system with a minimum cost subject to the condition that it must satisfy the entire requirement.
- **Scope:** Scope of this document is to put down the requirements, clearly identifying the information needed by the user, the source of the information and outputs expected from the system.

Gantt Chart

Task	Start Date	Duration	End Date
Analysis	15-Feb-13	3 Days	17-Feb-13
Planning	18-Feb-13	2 Days	19-Feb-13
Installation of Software	20-Feb-13	1 Day	20-Feb-13
Text Chat(One to One)	21-Feb-13	5 Days	25-Feb-13
Voice Chat(One to One)	21-Feb-13	10 Days	02-Mar-13
Text Chat(One to Many)	26-Feb-13	5 Days	02-Mar-13
Voice Chat(One to Many)	03-Mar-13	15 Days	17-Mar-13
Text Chat(Group Chat)	03-Mar-13	10 Days	12-Mar-13
Voice Chat(Group Cast)	18-Mar-13	11 Days	28-Mar-13
File Transfer	13-Mar-13	5 Days	17-Mar-13
Screen Share(One to One)	29-Mar-13	6 Days	03-Apr-13
Screen Share(One to Many)	04-Apr-13	15 Days	18-Apr-13
Screen Share(Many to Many)	19-Apr-13	7 Days	25-Apr-13
Video Conferencing	26-Apr-13	4 Days	29-Apr-13
Integration	29-Mar-13	35 Days	02-May-13
Final Report	03-May-13	3 Days	05-May-13

References

[1] MSDN tutorials for asynchronous client/server

- http://msdn.microsoft.com/en-us/library/bew39x2a.aspx
- http://msdn.microsoft.com/en-us/library/fx6588te.aspx

[2] MSDN socket tutorial

- http://msdn.microsoft.com/en-us/library/b6xa24z5.aspx

[3] Audio I/O libraries

- http://www.codeproject.com/Articles/19854/Sending-and-playing-microphone-audio-over-network

[4] Video encoding Library

- http://www.microsoft.com/en-in/download/details.aspx?id=27870

[5] How to use a web cam in C# with .NET Framework 4.0 and Microsoft Expression Encoder 4

- http://www.codeproject.com/Articles/202464/How-to-use-a-WebCam-in-C-with-the-NET-Framework-4