1. ABSTRACT

1. Abstract:

In any college or institute, the education, study or planning for some events can be improved by increasing interaction between students with teachers or teacher with another teacher. The students may have different queries which they want to discuss with the teachers but the teacher may not available or free at that time. For some discussion or planning something the teacher may need to discuss with one another and can serve well to the students or the organization.

But as of now, the teachers or students need to be physically present in front of each other, who may not be available to others when needed. Since meeting physically is not possible all the times because of some reasons; the reasons may be they are distant to each other, their cabins may be far, they are not available at the right place, etc., it is sometimes not possible to find someone in the whole institute for that we need to communicate with them or someone who gave information about them.

This problem can be solved by providing a software solution to them which will help them to communicate with each other without leaving the computer or cabins, more specifically a video conferencing or video calling system.

For this, we need an efficient and effective way for communication, which will be reliable and cost effective.

With the growing population, the resource utilization increases i.e., internet. Generally, for video conferencing, we must need a fast and reliable connection. Beside this, there are many efficient ways for the conferencing, which gives a reliable connection without the use of internet and it is cost effective too. Since in any institute, all the PC's are connected via LAN, then it is not difficult to communicate with the peers.

2. INTRODUCTION

2. Introduction:

The first thing which comes in every mind when asked to communicate with someone using technology is the Internet. But accessing internet needs a cost to be paid to the ISP (Internet Service Provider), but moreover after paying this cost it is not necessary that the communication will be smooth and reliable or the internet access speed will be efficient.

Here, we are providing a software solution for teachers and students to communicate not only smoothly, efficiently but also economically.

The BIT Communication System uses an Intranet or LAN (Local Area Network) or WLAN (Wireless Local Area Network) connection present in the organization to make the user able to communicate with each other without the use of Internet and stops us paying any ISP Fee involved in communication. All it need is a LAN connection for transferring data.

2.1. About the Project:

This project is based on a simple client to client or peer to peer communication system. This is developed using **Microsoft Visual Studio** and the program is written in **Microsoft VB.NET** which uses UDP (User Datagram Protocol) to transfer the data packets from one computer to another computer via LAN or WLAN.

Since this project is developed using VB.NET, it provides a machine independent platform thus consist of a stand-alone application which can run over any Windows Operating System having .NET Framework installed in it. The application provides reliable and effective communication between the PC's that are connected in the same Local Area Network.

The application uses port to port calling by using local IP addresses and a same specified port at both the side to establish the connection or link.

The user at one end can communicate with the user at other end by accessing its port i.e., the user at one end can send the audio and the video stream to the specified port of the user at other end which will be easily fetched by the application at that end.

The data can be transferred both by guided and unguided media.

2.2 User Datagram Protocol (UDP):

UDP (User Datagram Protocol) is an alternative communications protocol to Transmission Control Protocol (TCP) used primarily for establishing low-latency and loss tolerating connections between applications on the Internet. Both UDP and TCP run on top of the Internet Protocol (IP) and are sometimes referred to as UDP/IP or TCP/IP. Both protocols send short packets of data, called datagrams.

UDP provides two services not provided by the IP layer. It provides port numbers to help distinguish different user requests and, optionally, a checksum capability to verify that the data arrived intact.

TCP has emerged as the dominant protocol used for the bulk of Internet connectivity owing to services for breaking large data sets into individual packets, checking for and resending lost packets and reassembling packets into the correct sequence. But these additional services come at a cost in terms of additional data overhead, and delays called latency.

In contrast, UDP just sends the packets, which means that it has much lower bandwidth overhead and latency. But packets can be lost or received out of order as a result, owing to the different paths individual packets traverse between sender and receiver.

UDP is an ideal protocol for network applications in which perceived latency is critical such as gaming, voice and video communications, which can suffer some data loss without adversely affecting perceived quality. In some cases, forward error correction techniques are used to improve audio and video quality in spite of some loss.

UDP can also be used in applications that require lossless data transmission when the application is configured to manage the process of retransmitting lost packets and correctly arranging received packets. This approach can help to improve the data transfer rate of large files compared with TCP.

In the Open Systems Interconnection (OSI) communication model, UDP, like TCP, is in layer 4, the Transport Layer. UDP works in conjunction with higher level protocols to help manage data transmission services including Trivial File Transfer Protocol (TFTP), Real Time Streaming Protocol (RTSP), Simple Network Protocol (SNP) and Domain Name System (DNS) lookups.

2.3 WORKING:

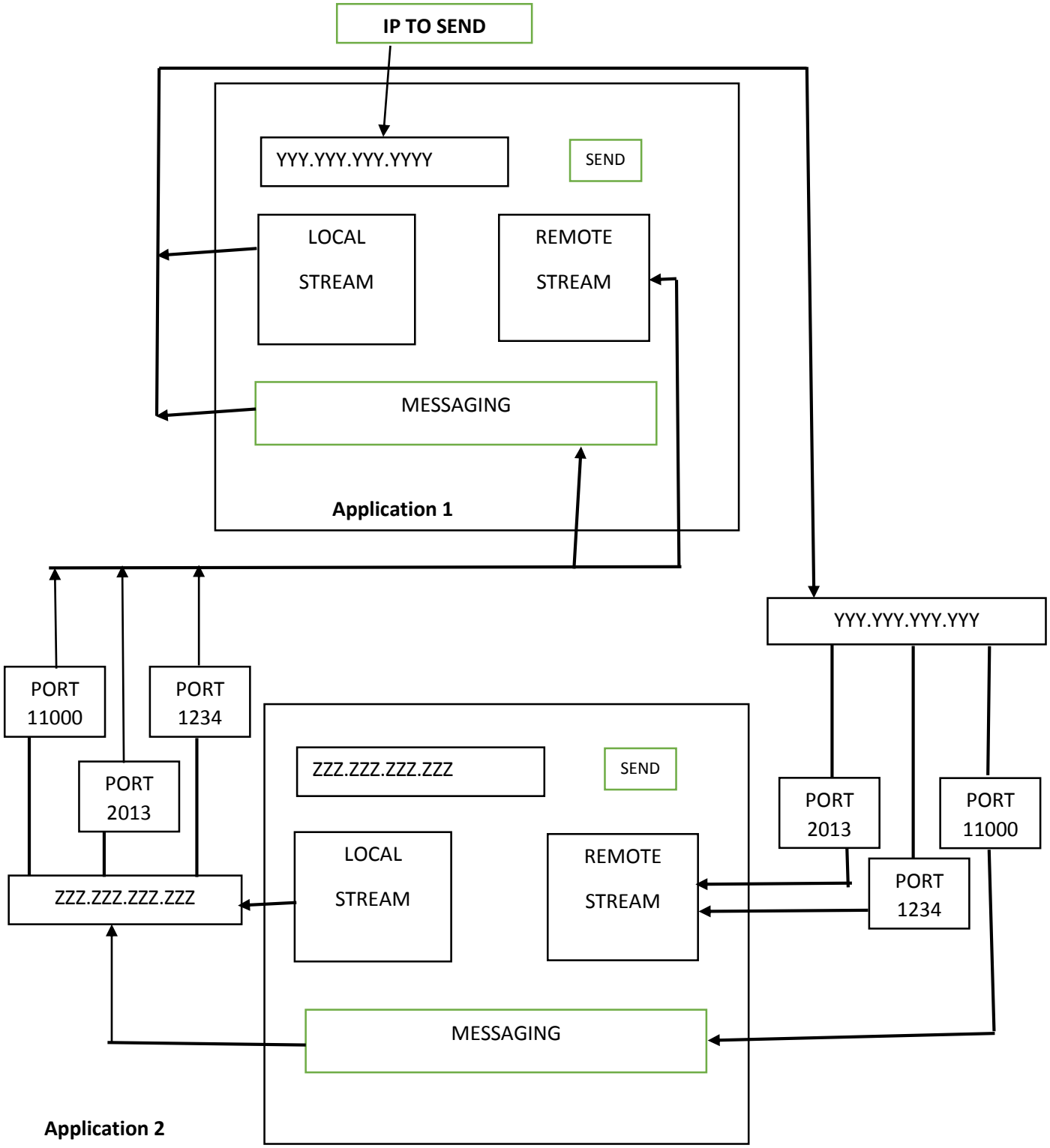


FIG 1: WORKING OF DIFFERENT ELEMENTS

The above figure shows the working of project. Application 1 is running on the machine with local IP YYY.YYY.YYY.YYY and Application 2 is running on the machine with local IP Address ZZZ.ZZZ.ZZZ.ZZZ.

As the application starts, the local stream (i.e., the audio and video stream), from the microphone and webcam of the local PC automatically gets started and displayed in the picture box named local stream in the figure.

When the user on Application 1 clicks the **CALL** button the local WEBCAM and MICROPHONE stream of Application 1 will be sent to the IP Address(IP: YYY.YYY.YYY.YYY) provided by the user in the text field of the Application 1.

The video stream will be sent to the port number 2013 of the destination machine (hence with the IP Address YYY.YYY.YYY.YYY) and audio stream will be sent to the port number 1234 of the destination machine (here again with the IP Address YYY.YYY.YYY.YYY).

The Application at the other end will be set to listen the video stream from its port 2013 and audio stream from port 1234, and as soon as the stream is received it will be displayed in the picture box written Remote Stream(as shown in fig 1).

When the user on Application 2 receives the stream they will get to know that who is calling. After seeing the notification the user have to put or click their name (who is trying to connect) from the list box appears on the application or can manually enter the IP Address in the text box, so that IP Address or Computer name must appear on the text box (besides the send button) and then by clicking on the **CALL** button, the video and audio stream of Application 2 will also be sent to the same ports of the machine with application 1(here with IP Address ZZZ.ZZZ.ZZZ.ZZZ) and will be displayed there in the picture box written remote stream as shown in the figure.

After the connection establishes the message stream will be sent to port number 11000, of both the application i.e., for the IP Address YYY.YYY.YYY.YYY and the IP Address ZZZ.ZZZ.ZZZ.ZZZ.

The application consists of several libraries.

2.4 LIBRARY USED IN THIS PROJECT:

1. **TouchlessLib.dll (TOUCHLESS LIBRARY)**:

Touchless is an SDK that allows users to create and experience multi-touch applications. Touchless started as Mike Wasserman's college project at Columbia University. The main idea: to offer users a new and cheap way of experiencing multi-touch capabilities, without the need of expensive hardware or software. All the user needs is a camera, which will track colored markers defined by the user.

What's in this project?

- WebCamLib.dll - Interfaces with DirectShow to grab webcam frames
- TouchlessLib.dll - Contains the functionality of Touchless SDK
- **TouchlessMgr**
    - Add functionality to save and load marker configuration files (reduce repeat training of the same marker, possibly provide autoconfig files for standard markers... will variant lighting allow for this?)
    - Implement additional marker data such as ColorAverage, ColorSpace, Axis and Roundness.
    - Add flood fill algorithm so we can add a marker with a few points in the Bitmap.
    - Refine the marker tracked colors as we find colors around the marker.
    - The representative color doesn't always match the perceived color of the marker.
    - Provide subsequent examples of a marker appearance
    - Have TouchlessMgr actually expose a way to get a list of the current markers
    - Make a better exception for camera start failure
    - Validate the PixelFormat of incoming images.
    - Create a utility function to retrieve ImageData in a consistent manner; we have a bit of code duplication right now.
    - Make a public interface for demo classes to implement, then allow the user to just

invoke start and stop of a demo class on the library

- Standardize error handling and exception generation across the project.

2. WebCamLib.dll (WEBCAM LIBRARY):

The WebCamLib.dll is a file which comes out of the touchless SDK. Touchless is real handy code to grab screen shots off a webcam.
But one can get several errors while using it as we got.

How to fix WebCamLib.dll errors?

Method 1: Solving the DLL Error by copying the WebCamLib.dll file to Windows System Folder. Extract the file (WebCamLib.dll) and paste it into the "**C:\Windows\System32**". If you are using 64 bit operating system copy the file into the "**C:\Windows\sysWOW64".**

Method 2: Copying the WebCamLib.dll file to the software File Folder.

Method 3: Doing a clean Install of the software that is giving the WebCamLib.dll Error.

Method 4: Solving the WebCamLib.dll error using the Windows System File Checker.

2.5 SUGGESTED CRITERIA:

1. **Multi-platform product:**

   For peer to peer videoconference this means that there are client distributions for various operating systems. For one to one videoconferences this means that both client and server are provided under various operating systems.

2. **Services offered:**

   A spectrum of communication services offered by the videoconference system i.e., audio, video and messages transmission.

3. **Classification of users involved in a conference:**

   Publisher (sender) and Subscriber (receiver)

4. **GUI configurability:**

   Ability to arrange view and control items of videoconference system on a desktop for individual user convenience.

5. **HW, SW requirements:**

   LAN and WLAN Connection over PC's, webcam and a microphone are min. hardware configuration required.

6. **New conference creation / setting:**

   New conference establishment requires only the IP Address or Computer name to Connect.

7. **Conference joining:**

   Complexity of the process that need to be done by user in order to join a conference. If there's notification arises to join someone, the user have to enter the IP Address of the sender to connect to each other.

8. **Additional network requirements:**

   Network adjustments that have to be done prior to begin of conference transmission itself.

3. DEVELOPMENT

DEVELOPMENT

3.1 Software Engineering Principles:

Software Engineering is the sub discipline of Computer Science that attempts to apply engineering principles to the creation, operation, modification and maintenance of the software components of various systems. As with much of Computer Science, the subject of Software Engineering is at an very early stage in its development. It is much more of an art than a science, and at present has little in common which classical engineering.

- What is engineering?

  Engineering is the application of well-understood scientific methods to the construction, operation, modification and maintenance of useful devices and systems.

- What is software?

  Software is comprises the aspects of a system not reduced to tangible devices, e.g. computer programs and documentation. It is distinguished from hardware, which consists of tangible devices, and often exists as collections of states of hardware devices. The boundary between hardware and software can be blurry, as with firmware and microcode.

- What is Software Engineering?

  Someday, Software Engineering may well be concerned with the application of well-understood scientific methods to the construction, operation, modification and maintenance of software. Today, however, Software Engineering is concerned with finding ways in which to produce working software for predictable costs in predictable time. When the problems involved are very simple or when only one person is involved, implementing software to meet their own needs, there isn't much to be said, and we are a long way from having any scientific principles for the production of software. Therefore, the major focus of software engineering today is on well-tested heuristics for the production of software to solve complex problems when many people are involved in the process, as users, as analysts, as programmers, as managers, etc. Therefore most of the issues in Software Engineering are concerned with interactions among people, rather than with the production of software.

In this project we have used the principles of Software Engineering.

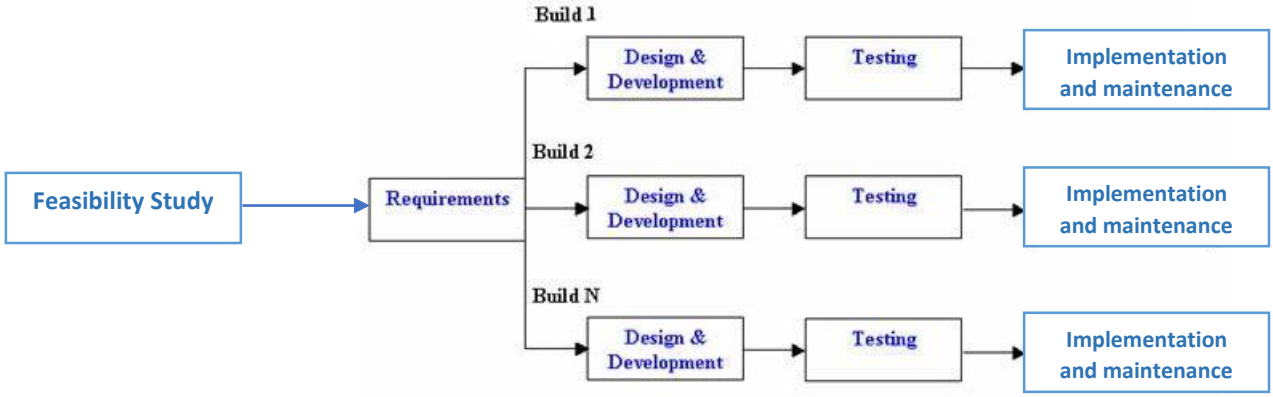
More specifically the **Incremental Model** of software development lifecycle.

3.2 Incremental Model:

In incremental model the whole requirement is divided into various builds. Multiple development cycles take place here, making the life cycle a **"multi-waterfall" cycle**. Cycles are divided up into smaller, more easily managed modules. Incremental model is a type of software development model like V-model, Agile model etc.

In this model, each module passes through the requirements, design, implementation and **testing** phases. A working version of software is produced during the first module, so you have working software early on during the **software life cycle**. Each subsequent release of the module adds function to the previous release. The process continues till the complete system is achieved.

3.2.1 Diagram of Incremental Model:



Incremental Life Cycle Model

3.2.2 Advantages of Incremental model:

- Generates working software quickly and early during the software life cycle.
- This model is more flexible – less costly to change scope and requirements.
- It is easier to test and debug during a smaller iteration.

- In this model customer can respond to each built.
- Lowers initial delivery cost.
- Easier to manage risk because risky pieces are identified and handled during it'd iteration.

3.2.3 Disadvantages of Incremental model:

- Needs good planning and design.
- Needs a clear and complete definition of the whole system before it can be broken down and built incrementally.

3.3 Feasibility Study:

The main aim of feasibility study is to determine whether it would be financially and technically feasible to develop the product.

- At first project managers or team leaders try to have a rough understanding of what is required to be done by visiting the client side. They study different input data to the system and output data to be produced by the system. They study what kind of processing is needed to be done on these data and they look at the various constraints on the behavior of the system.

- After they have an overall understanding of the problem they investigate the different solutions that are possible. Then they examine each of the solutions in terms of what kind of resources required, what would be the cost of development and what would be the development time for each solution.

- Based on this analysis they pick the best solution and determine whether the solution is feasible financially and technically. They check whether the customer budget would meet the cost of the product and whether they have sufficient technical expertise in the area of development.

After feasibility study we came up with some alternatives:

1.At first we tried to use **WebRTC (Web Real Time Communication)** which is the browser supported application for video conferencing with **HTML5**, **JavaScript**, **node.js** and **websockets.**

WebRTC:

WebRTC (**Web Real-Time Communication**) is a collection of communications protocols and application programming interfaces that enable real-time communication over peer-to-peer connections. This allows web browsers to not only request resources from backend servers, but also real-time information from browsers of other users.

This enables applications such as video conferencing, file transfer, chat, or desktop sharing without the need of either internal or external plugins.

WebRTC is being standardized by the World Wide Web Consortium (W3C) and the Internet Engineering Task Force (IETF). The reference implementation is released as free software under the terms of a BSD license. OpenWebRTC provides another free implementation based on the multimedia framework GStreamer.

WebRTC uses Real-Time Protocol to transfer audio and video.

HTML5:

HTML5 is a markup language used for structuring and presenting content on the World Wide Web. It is the fifth and current version of the HTML standard.

It was published in October 2014 by the World Wide Web Consortium (W3C) to improve the language with support for the latest multimedia, while keeping it both easily readable by humans and consistently understood by computers and devices such as web browsers, parsers, etc. HTML5 is intended to subsume not only HTML 4, but also XHTML 1 and DOM Level 2 HTML.

HTML5 includes detailed processing models to encourage more interoperable implementations; it extends, improves and rationalizes the markup available for documents, and introduces markup and application programming interfaces (APIs) for complex web applications.

For the same reasons, HTML5 is also a candidate for cross-platform mobile applications, because it includes features designed with low-powered devices in mind.

Many new syntactic features are included. To natively include and handle multimedia and graphical content, the

new <video>, <audio> and <canvas> elements were added, and support for scalable vector graphics (SVG) content and MathML for mathematical formulas. To enrich the semantic content of documents, new page structure elements such as <main>, <section>, <article>, <header>, <footer>, <aside>, <nav> and <figure>, are added. New attributes are introduced, some elements and attributes have been removed, and others such as <a>, <cite> and <menu> have been changed, redefined or standardized.

The APIs and Document Object Model (DOM) are now fundamental parts of the HTML5 specification and HTML5 also better defines the processing for any invalid documents.

JavaScript:

JavaScript ("JS" for short) is a full-fledged dynamic programming language that, when applied to an HTML document, can provide dynamic interactivity on websites. It was invented by Brendan Eich, co-founder of the Mozilla project, the Mozilla Foundation, and the Mozilla Corporation.
JavaScript is incredibly versatile. You can start small, with carousels, image galleries, fluctuating layouts, and responses to button clicks. With more experience you'll be able to create games, animated 2D and 3D graphics, comprehensive database-driven apps, and much more!

JavaScript itself is fairly compact yet very flexible. Developers have written a large variety of tools on top of the core JavaScript language, unlocking a vast amount of extra functionality with minimum effort. These include:

- Browser Application Programming Interfaces (APIs) — APIs built into web browsers, providing functionality like dynamically creating HTML and setting CSS styles, collecting and manipulating a video stream from the user's webcam, or generating 3D graphics and audio samples.
- Third-party APIs to allow developers to incorporate functionality in their sites from other content providers, such as Twitter or Facebook.
- Third-party frameworks and libraries you can apply to your HTML to allow you to rapidly build up sites and applications.

Node.js:

Node.js is a server-side platform built on Google Chrome's JavaScript Engine (V8 Engine). Node.js was developed by Ryan Dahl in 2009 and its latest version is v0.10.36.

Node.js is an open source, cross-platform runtime environment for developing server-side and networking applications. Node.js applications are written in JavaScript, and can be run within the Node.js runtime on OS X, Microsoft Windows, and Linux.

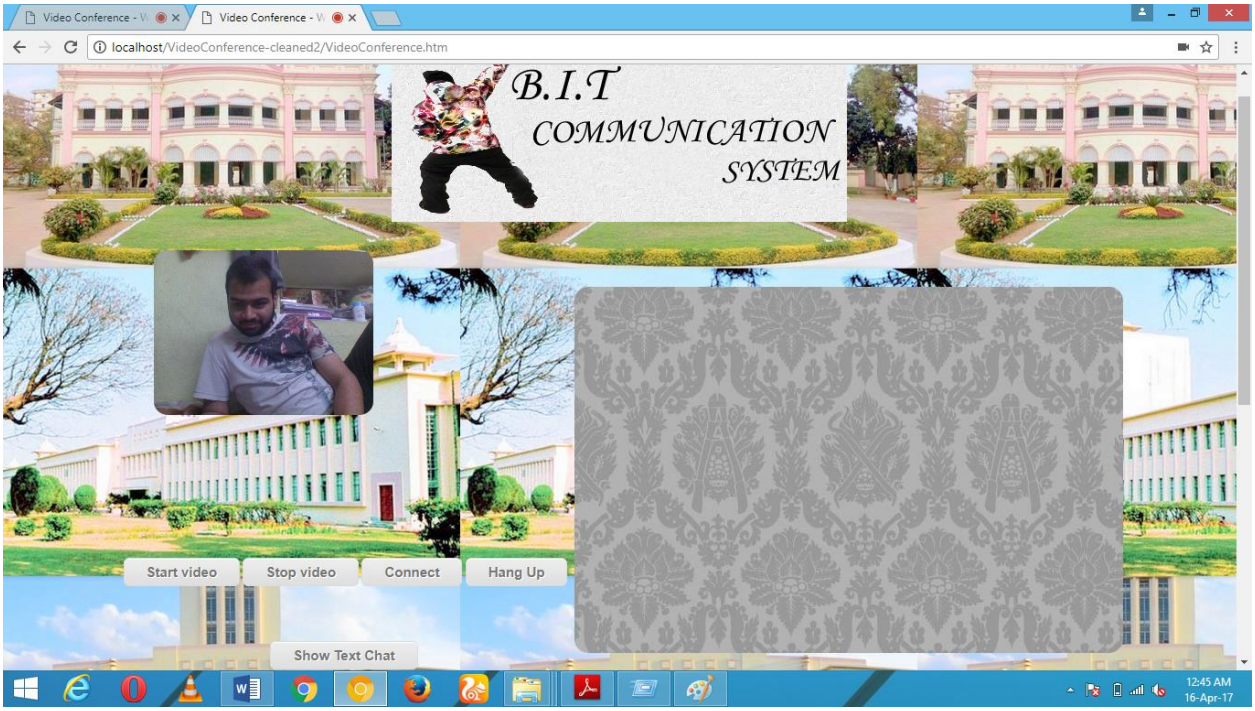
Node.js also provides a rich library of various JavaScript modules which simplifies the development of web applications using Node.js to a great extent.

WEBSOCKET PROTOCOL:

The WebSocket protocol was designed to work well with the existing Web infrastructure. As part of this design principle, the protocol specification defines that the WebSocket connection starts its life as an HTTP connection, guaranteeing full backwards compatibility with the pre-WebSocket world. The protocol switch from HTTP to WebSocket is referred to as a theWebSocket handshake.

The browser sends a request to the server, indicating that it wants to switch protocols from HTTP to WebSocket.

We also developed its prototype as shown below:



Limitations found during feasibility study of this prototype:

- But we didn't find it efficient as for using this application.
- We always needed to run a local web server which increased the software requirement.
  A Good software or a video chatting software should work as a standalone software which can't be achieved using WebRTC
- The websockets were also needed to be activated manually by using node.js, which made it not so user friendly.The application should be designed in such a way that everyone can use it easily.

2.The second alternative which matched our requirements was **VB.Net** with **User Datagram Protocol:**

VB.Net:

Visual Basic .NET (VB.NET) is an object-oriented computer programming language implemented on the .NET Framework. Although it is an evolution of classic Visual Basic language, it is not backwards-compatible with VB6, and any code written in the old version does not compile under VB.NET.

Like all other .NET languages, VB.NET has complete support for object-oriented concepts. Everything in VB.NET is an object, including all of the primitive types (Short, Integer, Long, String, Boolean, etc.) and user-defined types, events, and even assemblies. All objects inherits from the base class Object.

VB.NET is implemented by Microsoft's .NET framework. Therefore, it has full access to all the libraries in the .Net Framework. It's also possible to run VB.NET programs on Mono, the open-source alternative to .NET, not only under Windows, but even Linux or Mac OSX.

The following reasons make VB.Net a widely used professional language:

- Modern, general purpose.
- Object oriented.
- Component oriented.
- Easy to learn.
- Structured language.
- It produces efficient programs.
- It can be compiled on a variety of computer platforms.
- Part of .Net Framework.

Strong Programming Features VB.Net:

VB.Net has numerous strong programming features that make it endearing to multitude of programmers worldwide. Let us mention some of these features:

- Boolean Conditions
- Automatic Garbage Collection
- Standard Library
- Assembly Versioning
- Properties and Events
- Delegates and Events Management
- Easy-to-use Generics
- Indexers
- Conditional Compilation
- Simple Multithreading

User Datagram Protocol:

UDP (User Datagram Protocol) is an alternative communications protocol to Transmission Control Protocol (TCP) used primarily for establishing low-latency and loss tolerating connections between applications on the Internet. Both UDP and TCP run on top of the Internet Protocol (IP) and are sometimes referred to as UDP/IP or TCP/IP. Both protocols send short packets of data, called datagrams.

UDP provides two services not provided by the IP layer. It provides port numbers to help distinguish different user requests and, optionally, a checksum capability to verify that the data arrived intact.

TCP has emerged as the dominant protocolused for the bulk of Internet connectivity owing to services for breaking large data sets into individual packets, checking for and resending lost packets and

reassembling packets into the correct sequence. But these additional services come at a cost in terms of additional data overhead, and delays called latency.

In contrast, UDP just sends the packets, which means that it has much lower bandwidthoverhead and latency. But packets can be lost or received out of order as a result, owing to the different paths individual packets traverse between sender and receiver.

UDP is an ideal protocol for network applications in which perceived latency is critical such as gaming, voice and video communications, which can suffer some data loss without adversely affecting perceived quality. In some cases, forward error correction techniques are used to improve audio and video quality in spite of some loss.

UDP can also be used in applications that require lossless data transmission when the application is configured to manage the process of retransmitting lost packets and correctly arranging received packets. This approach can help to improve the data transfer rate of large files compared with TCP.

In the Open Systems Interconnection (OSI) communication model, UDP, like TCP, is in layer 4, the Transport Layer. UDP works in conjunction with higher level protocols to help manage data transmission services including Trivial File Transfer Protocol (TFTP), Real Time Streaming Protocol (RTSP), Simple Network Protocol (SNP) and Domain Name System (DNS) lookups.

We have opted for this combination i.e. of VB.NET and User Datagram Protocol because VB.Net provides a large set of libraries and tools for rapid application development and with its GUI development environment (Visual Studio) it becomes much easier to develop the interface of the application and UDP provides a faster way to transmit data over LAN as it doesn't have additional features like error control nor it adds additional bits to packets like checksum all it provides is speed which is very essential for video transmission. UDP is approximately 3 times faster than TCP.

3.4 Requirements Specification:

A **software requirements specification** (SRS) is a description of a software system to be developed. It lays out functional and non-functional requirements, and may include a set of use cases that describe user interactions that the software must provide.

Software requirements specification establishes the basis for an agreement between customers and contractors or suppliers (in market-driven projects, these roles may be played by the marketing and development divisions) on what the software product is to do as well as what it is not expected to do. Software requirements specification permits a rigorous assessment of requirements before design can begin and reduces later redesign. It should also provide a realistic basis for estimating product costs, risks, and schedules. Used appropriately, software requirements specifications can help prevent software project failure.

The software requirements specification document enlists enough and necessary requirements that are required for the project development. To derive the requirements we need to have clear and thorough understanding of the products to be developed or being developed. This is achieved and refined with detailed and continuous communications with the project team and customer till the completion of the software.

The SRS may be one of a contract deliverable Data Item Descriptions or have other forms of organizationally-mandated content.

Here's the SRS Document with all the details and the functions needed:

9. **Multi-platform product:**

   It is not necessary that everyone who wants to talk to you or wants to use the application to communicate should be using same operating system.
   the product which we were trying to build was to be Multi-platform product which we somehow have achieved.

10. **Video Transmission:**
    The software which we had to build should contain Video Transmission System.

11. Audio Transmission:

The software which we had to build should contain Audio Transmission System.

12. Message Transmission:

The software which we had to build should contain Message Transmission System.

13. Number of users involved in a conference:

Minimum 2 users should be able to communicate using the project.

Publisher (sender) and Subscriber (receiver).

14. Easy Connection Mechanism:

The Desired software should use an easy connection mechanism.

15. GUI configurability:

Ability to arrange view and control items of videoconference system on a desktop for individual user convenience.

16. HW, SW requirements:

LAN and WLAN Connection over PC's, webcam and a microphone are minimum hardware configuration required.

17. Conference joining:

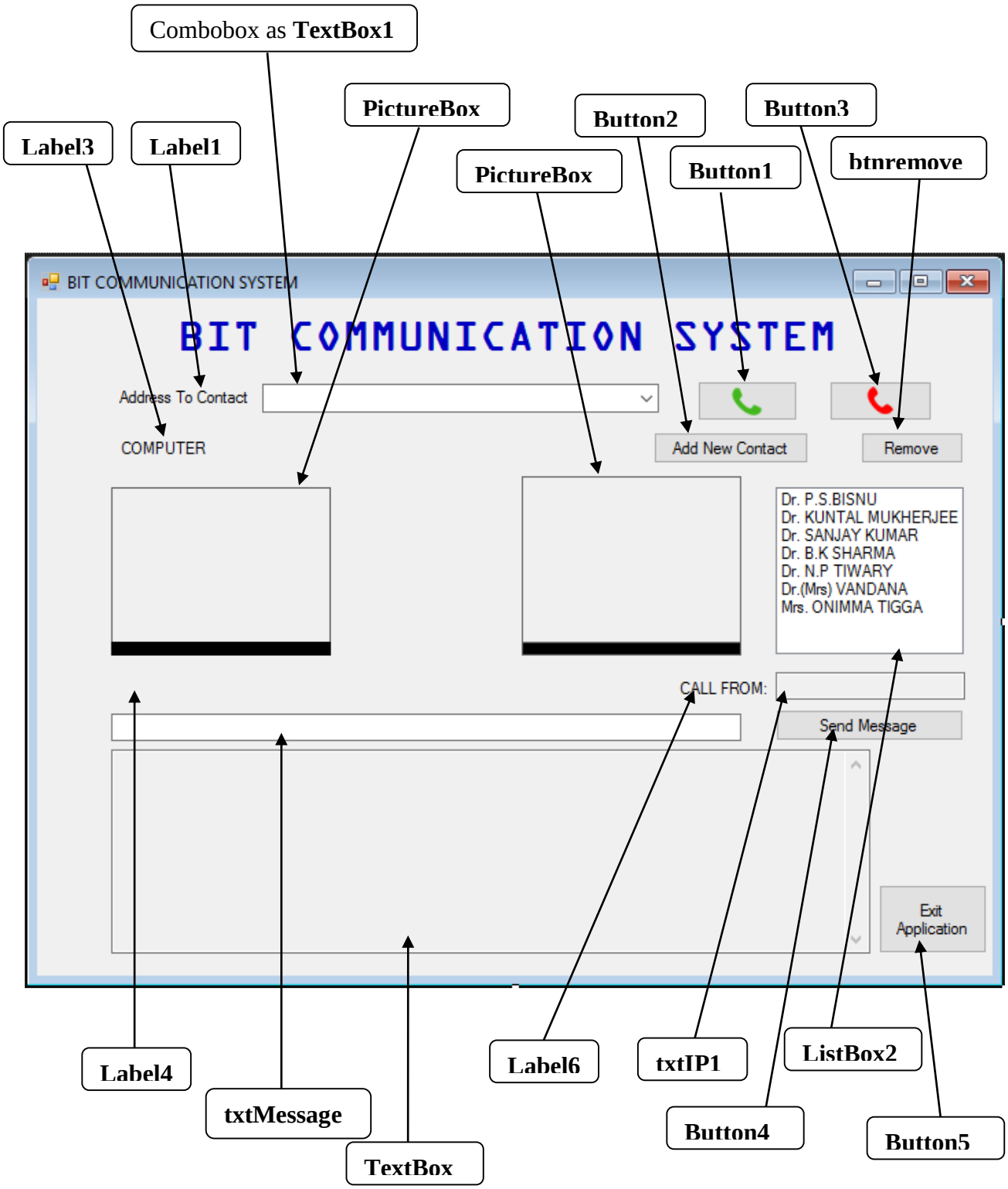
Complexity of the process that need to be done by user in order to join a conference. If there's notification arises to join someone, the user have to enter the IP Address of the sender to connect to each other.

3.5 DESIGN AND CODING:

3.5.1 DESIGN PHASE:

Next step is to bring down whole knowledge of requirements and analysis on the desk and design the software product. The inputs from users and information gathered in requirement gathering phase are the inputs of this step. The output of this step comes in the form of two designs; logical design and

physical design. Engineers produce meta-data and data dictionaries, logical diagrams, data-flow diagrams and in some cases pseudo codes.



Label1:

It is indicating the Combo box as **TextBox1,** to enter the IP Address of the system whose notification is coming to be connected or the user wants to connet to other user.

Combobox as **TextBox1:**

This combo box consist of the list of IP address of the other user connected in LAN or the user can enter IP Address manually.

Button1:

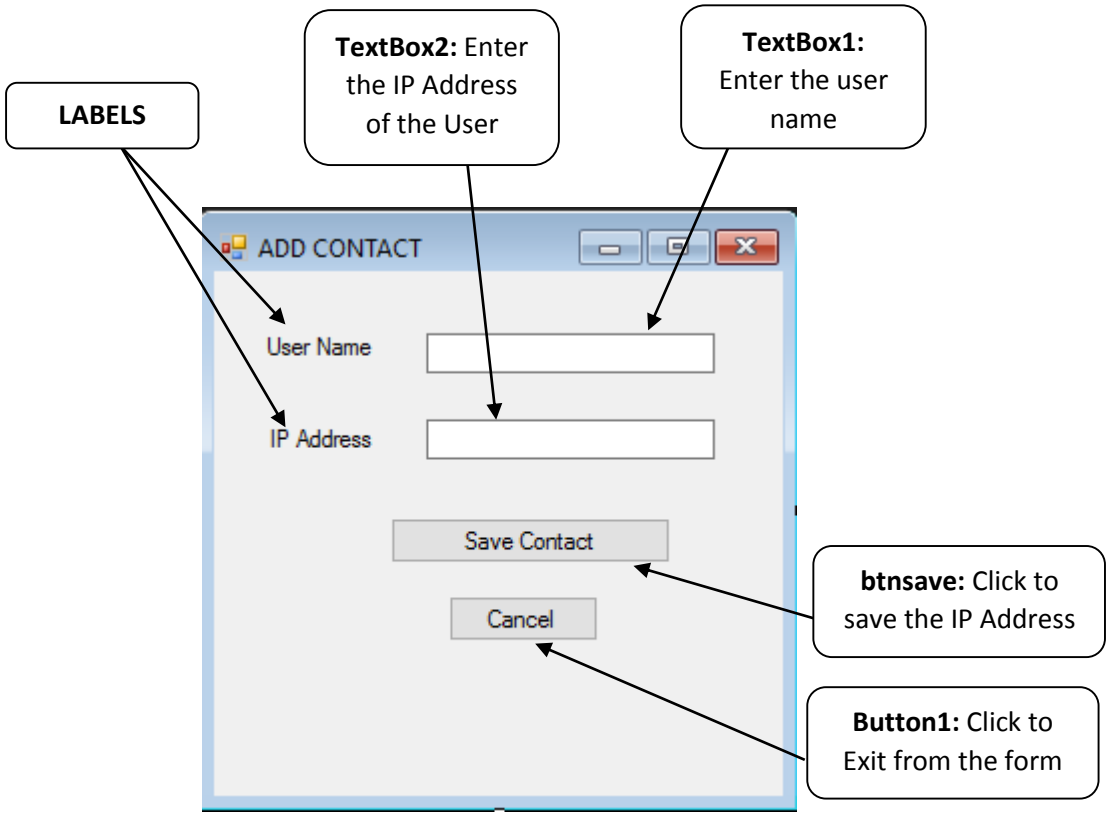
This button is programmed to connect with the remote stream. When the IP Address is entered in the TextBox1, the user need to click send button to establish connection, when the button is clicked by the sender, the receiver will receive notification at the notification label, the receiver also need to enter or select the name from the list box for the appearance of IP in the combo box then the user need to click on send button, then the connection will be established.

Button3:

This button is to **END the CALL** , when the conversation is over the user need to click the End Call button, to disconnect the ports.

Button2:

This button is created to ADD CONTACTS in the ListBox1. When it is clicked a new form is echoed over the screen, which is shown below:



Label3:

This label shows the local computer name.

PictureBox1:

It is the local video stream. When the user opens the application, the local stream starts running i.e., the webcam gets started and there's the local video showing.

PictureBox2:

It is the remote video stream. When the PC's are connected, there's the stream video of the remote user is displayed.

ListBox2:

It consists of the lists of the teacher referring IP Address of their PC's.

When the user gets the notification to connect, the user can simply select the name of the sender from the list box to get connected.

btnremove:

This button is used to remove the selected item from the list box.

Label4:

This label shows the notification for:

- Connecting user
- Peer connected
- Peer Disconnected, etc

Label6:

It indicates the text box where the connected PC's IP are shown.

txtIP1:

It displays the connected PC IP Address.

txtMessage:

It is the text box to write the message to be sent.

Button4:

This button is to send the message written in txtMessage textbox.

TextBox2:

The sent message with its notification will be displayed in this text box.

Button5:

This button is used to exit from the application.

3.5.2 CODING PHASE:

This step is also known as programming phase. The implementation of software design starts in terms of writing program code in the suitable programming language and developing error-free executable programs efficiently.

In designing phase we have already taken all the major decision regarding the system, now it's time to develop the physical system. We will consider designing phase output as input for coding phase. The basic role of this phase is to convert designing into code using the programming language decided in designing phase. The well-developed code in this phase can help to reduce the efforts required in testing and maintenance. But even we make any silly mistake; it may lead us to put extra efforts in testing and maintenance.

If we see it in a business perspective, the cost for testing efforts and maintenance is much higher than coding. So it always makes sense to spend time on coding phase. Here all developers write their own code and merged with other developer's code to make sure that all modules developed by different developers interact with each other as per expectations. This is one of the longest phases in software development life cycle.

As this project is not big that's why we have selected the "Incremental model" among several Software Engineering Models. In incremental model different builds are made so we made several builds and by incrementing functions in each and every build and testing them, thus we didn't require the project to be divided it into modules. And we chose VB.Net for programming because of its stand-alone features.

The explanation of code is as follows:

First we have imported several library functions as follows:

```
1   Imports System.Net.Sockets
2   Imports System.Threading
3   Imports TouchlessLib
4   Imports System.Net
5   Imports System.Text
6   Imports System.IO
7   Imports Microsoft.VisualBasic
8   Imports System.Runtime.InteropServices
9   Imports System.Net.Sockets.Socket
10  Imports System
```

Imports System.net.sockets: The namespace provides a managed implementation of the windows Sockets interface for developers who need to tightly control access to the network.

Imports System.Threading: The System.Threading namespace provides classes and interfaces that enable multithreaded programming. In addition to classes for synchronizing

thread activities and access to data this namespace includes a ThreadPool class that allows you to use a pool of system-supplied threads, and a Timer class that executes callback methods on thread pool threads.

Imports TouchlessLib: The TouchlessLib namespace is used to capture the images from webcam.

Imports System.Net: The System.Net namespace provides a simple programming interface for many of the protocols used on network today.Classes in the System.Net namespace can be used to develop windows store apps or desktop apps.

Import.System.Text: The System.Text namespace contains classes that represent ASCII and Unicode character encodings; abstract base classes for converting blocks of characters to and from blocks of bytes; and a helper class that manipulates and formats String objects without creating intermediate instances of String.

Import System.IO:The System.IO namespace contains types that allow reading and writing to files and data streams, and types that provide basic file and directory support.

Imports Microsoft.VisualBasics: The Visual Basic Runtime provides the underlying implementation for global Visual Basic functions and language features such as Len, IsDate, and CStr. And though the new Visual Basic Runtime provides similar facilities as its predecessors, **it is entirely managed code (developed in Visual Basic .NET) that executes on the common language runtime**. Furthermore, the Visual Basic Runtime is part of the .NET Framework, so it is never something separate that your application has to carry or deploy.

Import.System.Runtime.InteropService: The System.Runtime.InteropServices namespace provides a wide variety of members that support COM interop and platform invoke services.

```
15  Public Class Form1
16      Inherits System.Windows.Forms.Form
17      Dim subscriber As New UdpClient()
18      Dim publisher As New UdpClient()
19      Dim mycomputername As String = Environment.MachineName
20      Dim mycomputerIP() As System.Net.IPAddress = System.Net.Dns.GetHostAddresses(mycomputername)
21
22      Dim Touchless As New TouchlessLib.TouchlessMgr
23      Dim Camera1 As TouchlessLib.Camera = Touchless.Cameras.ElementAt(0)
24      Dim picsize As Size = New Size(160, 120)
25
26      Public receivingUdpClient As UdpClient
27      Public RemoteIpEndPoint As New System.Net.IPEndPoint(System.Net.IPAddress.Any, 0)
28      Public ThreadReceive As System.Threading.Thread
29
30      Dim SocketNO As Integer
31      Dim GLOIP As IPAddress
32      Dim GLOINTPORT As Integer
33      Dim bytCommand As Byte() = New Byte() {}
34      Dim udpClient As New UdpClient
35
```

Here we are declaring the class.

```
17      Dim subscriber As New UdpClient()
18      Dim publisher As New UdpClient()
```

Here we defining the UDP by declaring subscriber and publisher as udpClient ()

```
19      Dim mycomputername As String = Environment.MachineName
20      Dim mycomputerIP() As System.Net.IPAddress = System.Net.Dns.GetHostAddresses(mycomputername)
```

Here the name of this computer is established at system startup when the name is read from the registry. If this computer is a node in a cluster, the name of the node is returned. Using System.Net.Dns.GetHostAddresses, weget the host name of the local computer.

```
22      Dim Touchless As New TouchlessLib.TouchlessMgr
23      Dim Camera1 As TouchlessLib.Camera = Touchless.Cameras.ElementAt(0)
24      Dim picsize As Size = New Size(160, 120)
```

Here we first define the Touchlesslib and then by using Touchlesslib.Cameras we obtain a list of cameras of our system and by doing Cameras.ElementAt (0) we are explicitly selecting the first from the list.

```
26      Public receivingUdpClient As UdpClient
27      Public RemoteIpEndPoint As New System.Net.IPEndPoint(System.Net.IPAddress.Any, 0)
28      Public ThreadReceive As System.Threading.Thread
```

IPEndPoint class contains the host and local or remote port information needed by an application to connect to a service on a host. Here we combining the host's IP address and port number of a service, the IPEndPoint class forms a connection point to a service.

```
30      Dim SocketNO As Integer
31      Dim GLOIP As IPAddress
32      Dim GLOINTPORT As Integer
33      Dim bytCommand As Byte() = New Byte() {}
34      Dim udpClient As New UdpClient
```

In line number 30, we have declared **SockectNo** as type integer, which will accept the value of port number for text communication.

In line number 31, we have declared GLOIP which will accept receivers IP Address.

```
40      Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
41
42
43          publisher.Client.Blocking = False
44          subscriber.Client.ReceiveTimeout = 100
45          subscriber.Client.Blocking = False
46          subscriber.ExclusiveAddressUse = False
47          publisher.ExclusiveAddressUse = False
48          TextBox1.Text = ""
49
50          Label3.Text = "This Computer Name: " & Environment.MachineName
51
52          Touchless.CurrentCamera = Camera1
53          Touchless.CurrentCamera.CaptureWidth = picsize.Width
54          Touchless.CurrentCamera.CaptureHeight = picsize.Height
55
56
57          AxVideoChatReceiver2.ReceiveAudioStream = True
58          AxVideoChatReceiver2.ReceiveVideoStream = False
59
60          AxVideoChatReceiver2.Listen(Convert.ToString(mycomputerIP), 1234)
61          subscriber.Client.Bind(New Net.IPEndPoint(Net.IPAddress.Any, 2013))
62          'if nt wrking then in btn1_click
63          Try
64              SocketNO = "11000"
65              receivingUdpClient = New System.Net.Sockets.UdpClient(SocketNO)
66              ThreadReceive = New System.Threading.Thread(AddressOf ReceiveMessages)
67              ThreadReceive.Start()
68
69          Catch x As Exception
70              Console.WriteLine(x.Message)
71              TextBox2.Text = TextBox2.Text & vbCrLf & x.Message
72          End Try
73
74
75      End Sub
```

Here Form1_Load method is declared and define which handles the event of loading form 1. This method/ function contains all the elements to start the transmission services.

```
76      Public Sub ReceiveMessages()
77          Try
78              Dim receiveBytes As [Byte]() = receivingUdpClient.Receive(RemoteIpEndPoint)
79              txtIP1.Text = RemoteIpEndPoint.Address.ToString
80              TextBox1.Text = RemoteIpEndPoint.Address.ToString
81              Dim BitDet As BitArray
82              BitDet = New BitArray(receiveBytes)
83
84              Dim strReturnData As String = System.Text.Encoding.Unicode.GetString(receiveBytes)
85
86              TextBox2.Text = TextBox2.Text + vbCrLf + "Recieved Message:"
87              If (Encoding.ASCII.GetChars(receiveBytes) = "End call") Then
88
89
90                  MyBase.Close()
91                  Application.Restart()
92
93              End If
94              TextBox2.Text = TextBox2.Text & Encoding.ASCII.GetChars(receiveBytes) + ""
95              TextBox2.Text = TextBox2.Text & vbCrLf
96
97              TextBox2.Text = TextBox2.Text & vbCrLf
98              NewInitialize()
99          Catch e As Exception
100             Console.WriteLine(e.Message)
101         End Try
102     End Sub
```

Here the Receivemessages() thread is defined which allows the user to start receiving the text messages from the other end.

```
103     Public Sub NewInitialize()
104         Console.WriteLine("Thread *Thread Receive* reinitialized")
105         ThreadReceive = New System.Threading.Thread(AddressOf ReceiveMessages)
106         ThreadReceive.Start()
107     End Sub
```

Here the NewInitialize() method is defined to initialize the message receiving thread.

```
108     Private Sub Timer1_Tick(ByVal sender As Object, ByVal e As EventArgs) Handles Timer1.Tick
109
110         Try
111
112             Dim bitmapz As Bitmap = New Bitmap(picsize.Width, picsize.Height)
113             bitmapz = Touchless.CurrentCamera.GetCurrentImage
114             PictureBox1.Image = bitmapz
115
116             Dim sendbytes() As Byte
117             bytesfromimage(sendbytes, bitmapz)
118
119             publisher.Send(sendbytes, sendbytes.Length)
120         Catch ex As Exception
121         End Try
122
123
124         Try
125
126             Dim ep As System.Net.IPEndPoint = New System.Net.IPEndPoint(System.Net.IPAddress.Any, 0)
127
128             Dim rcvbytes() As Byte = subscriber.Receive(ep)
129             Dim bitmapz As Bitmap = New Bitmap(picsize.Width, picsize.Height)
130             imagefrombytes(rcvbytes, bitmapz)
131             PictureBox2.Image = bitmapz
132
133         Catch ex As Exception
134         End Try
135
136     End Sub
```

Here the actions are defined which are to be performed every time the timer ticks.

In line number 112 a bitmap is declared which is to be used to draw the receiving bytes into image in the picturebox.

```
140     Private Sub imagefrombytes(ByRef bytez() As Byte, ByRef piccolor As Bitmap)
141         Dim rect As New Rectangle(0, 0, piccolor.Width, piccolor.Height)
142         Dim bmpData As System.Drawing.Imaging.BitmapData = piccolor.LockBits(rect,
143             Drawing.Imaging.ImageLockMode.ReadWrite, Imaging.PixelFormat.Format32bppRgb)
144         Dim ptr As IntPtr = bmpData.Scan0
145         Dim bytes As Integer = bmpData.Stride * piccolor.Height
146         Dim rgbValues(bytes - 1) As Byte
147         System.Runtime.InteropServices.Marshal.Copy(ptr, rgbValues, 0, bytes)
148         Dim secondcounter As Integer
149         Dim tempred As Integer
150         Dim tempblue As Integer
151         Dim tempgreen As Integer
152         Dim tempalpha As Integer
153         secondcounter = 0
154         While secondcounter < rgbValues.Length
155             tempblue = rgbValues(secondcounter)
156             tempgreen = rgbValues(secondcounter + 1)
157             tempred = rgbValues(secondcounter + 2)
158             tempalpha = rgbValues(secondcounter + 3)
159             tempalpha = 255
160
161             tempred = bytez(((secondcounter * 0.25) * 3) + 0)
162             tempgreen = bytez(((secondcounter * 0.25) * 3) + 1)
163             tempblue = bytez(((secondcounter * 0.25) * 3) + 2)
```

```
164
165             rgbValues(secondcounter) = tempblue
166             rgbValues(secondcounter + 1) = tempgreen
167             rgbValues(secondcounter + 2) = tempred
168             rgbValues(secondcounter + 3) = tempalpha
169
170             secondcounter = secondcounter + 4
171         End While
172         System.Runtime.InteropServices.Marshal.Copy(rgbValues, 0, ptr, bytes)
173         piccolor.UnlockBits(bmpData)
174         Label5.Text = ""
175     End Sub
```

Here in this part the images are formed using the received bytes from remote stream and drawn on the bitmap.

```
176     Private Sub bytesfromimage(ByRef bytez() As Byte, ByRef piccolor As Bitmap)
177         Dim rect As New Rectangle(0, 0, piccolor.Width, piccolor.Height)
178         Dim bmpData As System.Drawing.Imaging.BitmapData = piccolor.LockBits(rect,
179             Drawing.Imaging.ImageLockMode.ReadWrite, Imaging.PixelFormat.Format32bppRgb)
180         Dim ptr As IntPtr = bmpData.Scan0
181         Dim bytes As Integer = bmpData.Stride * piccolor.Height
182         Dim rgbValues(bytes - 1) As Byte
183         System.Runtime.InteropServices.Marshal.Copy(ptr, rgbValues, 0, bytes)
184
185         Dim secondcounter As Integer
186         Dim tempred As Integer
187         Dim tempblue As Integer
188         Dim tempgreen As Integer
189         Dim tempalpha As Integer
190         secondcounter = 0
191         Dim bytelist As List(Of Byte) = New List(Of Byte)
192
193         While secondcounter < rgbValues.Length
194             tempblue = rgbValues(secondcounter)
195             tempgreen = rgbValues(secondcounter + 1)
196             tempred = rgbValues(secondcounter + 2)
197             tempalpha = rgbValues(secondcounter + 3)
198             tempalpha = 255
199
200             bytelist.Add(tempred)
201             bytelist.Add(tempgreen)
202             bytelist.Add(tempblue)
203
204             rgbValues(secondcounter) = tempblue
205             rgbValues(secondcounter + 1) = tempgreen
206             rgbValues(secondcounter + 2) = tempred
207             rgbValues(secondcounter + 3) = tempalpha
208
209             secondcounter = secondcounter + 4
210         End While
```

```
211
212
213         System.Runtime.InteropServices.Marshal.Copy(rgbValues, 0, ptr, bytes)
214
215         piccolor.UnlockBits(bmpData)
216
217         Dim bytearray(bytelist.Count - 1) As Byte
218         For i = 0 To bytelist.Count - 1
219             bytearray(i) = bytelist(i)
220         Next
221         bytez = bytearray
222
223     End Sub
```

Here in the above section the images are being converted to bytes ready to be transferred.

```
225     Private Sub Button1_Click(ByVal sender As Object, ByVal e As EventArgs) Handles Button1.Click
226
227
228         publisher.Connect(TextBox1.Text, 2013)
229
230         AxVideoChatSender1.VideoDevice = 0
231         AxVideoChatSender1.AudioDevice = 0
232         AxVideoChatSender1.VideoFormat = 0
233         AxVideoChatSender1.FrameRate = 15
234         AxVideoChatSender1.VideoBitrate = 128000
235         AxVideoChatSender1.AudioComplexity = 0
236         AxVideoChatSender1.AudioQuality = 8
237         AxVideoChatSender1.SendAudioStream = True
238         AxVideoChatSender1.SendVideoStream = False
239
240         AxVideoChatSender1.Connect(TextBox1.Text, 1234)
241
242         AxVideoChatReceiver1.ReceiveAudioStream = True
243         AxVideoChatReceiver1.ReceiveVideoStream = False
244
245         AxVideoChatReceiver2.ReceiveAudioStream = True
246         AxVideoChatReceiver2.ReceiveVideoStream = False
247
248         AxVideoChatReceiver2.Listen(Convert.ToString(mycomputerIP), 1234)
249         If (ListBox1.SelectedIndex >= 0) Then
250             Label5.Text = "Waiting for " + ListBox1.SelectedItem + " to accept the call."
251         Else
252             Label5.Text = "Waiting for " + TextBox1.Text + " to accept the call."
253
254         End If
255         TextBox1.Enabled = False
256
```

```
256
257         Dim pRet As Integer
258
259
260         Try
261             GLOIP = IPAddress.Parse(TextBox1.Text)
262             GLOINTPORT = "11000"
263             udpClient.Connect(GLOIP, GLOINTPORT)
264             If (TextBox2.Text = "") Then
265                 bytCommand = Encoding.ASCII.GetBytes("ANSWER THE CALL..")
266                 Label6.Text = "CALLING:"
267                 txtIP1.Text = TextBox1.Text
268
269             Else
270                 bytCommand = Encoding.ASCII.GetBytes("CALL ANSWERED!,YOU ARE NOW CONNECTED")
271                 Label6.Text = "Connected:"
272                 TextBox2.Text = TextBox2.Text + "YOU ANSWERED THE CALL"
273             End If
274             pRet = udpClient.Send(bytCommand, bytCommand.Length)
275
276             Console.WriteLine(Encoding.ASCII.GetString(bytCommand))
277
278         Catch ex As Exception
279             Console.WriteLine(ex.Message)
280             TextBox2.Text = TextBox2.Text & vbCrLf & ex.Message
281         End Try
282
283
284
285     End Sub
```

The above section contains code for the  Button which can act as a call initiator as well as the call accept button.

```
307     Private Sub btnremove_Click(sender As Object, e As EventArgs) Handles btnremove.Click
308         ListBox1.Items.Remove(ListBox1.SelectedItem)
309         TextBox1.Text = " "
310     End Sub
```

This section contains the code of remove button thus handles the click event of the same button.

```
312 Private Sub ListBox1_SelectedIndexChanged(sender As Object, e As EventArgs) Handles ListBox1.SelectedIndexChan
313     If (ListBox1.SelectedIndex = 0) Then
314         TextBox1.Text = "169.254.65.100"
315     End If
316     If (ListBox1.SelectedIndex = 1) Then
317         TextBox1.Text = "169.254.222.209"
318     End If
319     If (ListBox1.SelectedIndex = 2) Then
320         TextBox1.Text = "169.254.193.89"
321     End If
322     If (ListBox1.SelectedIndex = 3) Then
323         TextBox1.Text = "192.168.1.1"
324     End If
325     If (ListBox1.SelectedIndex = 4) Then
326         TextBox1.Text = "192.0.0.1"
327     End If
328     If (ListBox1.SelectedIndex = 5) Then
329         TextBox1.Text = "169.0.0.2"
330     End If
331     If (ListBox1.SelectedIndex = 6) Then
332         TextBox1.Text = "This Index is Empty"
333     End If
334 End Sub
```

This section of code contains the event of the changing selection from the listbox.

```
336     Private Sub Button3_Click(sender As Object, e As EventArgs) Handles Button3.Click
337         Dim pRet As Integer
338
339         Try
340             GLOIP = IPAddress.Parse(TextBox1.Text)
341             GLOINTPORT = "11000"
342             udpClient.Connect(GLOIP, GLOINTPORT)
343             bytCommand = Encoding.ASCII.GetBytes("End call")
344             pRet = udpClient.Send(bytCommand, bytCommand.Length)
345
346             Console.WriteLine(Encoding.ASCII.GetString(bytCommand))
347             TextBox2.Text = TextBox2.Text + vbCrLf + "Sent Message: "
348             TextBox2.Text = TextBox2.Text & Encoding.ASCII.GetChars(bytCommand) & ""
349
350             TextBox2.Text = TextBox2.Text + vbCrLf
351         Catch ex As Exception
352             Console.WriteLine(ex.Message)
353             TextBox2.Text = TextBox2.Text & vbCrLf & ex.Message
354         End Try
355         Try
356             ThreadReceive.Abort()
357             receivingUdpClient.Close()
358         Catch ex As Exception
359             Console.WriteLine(ex.Message)
360         End Try
361         AxVideoChatSender1.Disconnect()
362         AxVideoChatReceiver1.Disconnect()
363         AxVideoChatReceiver2.Disconnect()
364
365
366         publisher.Client.Close()
367         subscriber.Client.Close()
368
369         udpClient.Client.Close()
370
371         Application.Restart()
372     End Sub
```

The above code is of the  button which when clicked, restarts the current application as well as remote application.

```
Private Sub Button4_Click(sender As Object, e As EventArgs) Handles Button4.Click
    Dim pRet As Integer

    Try
        GLOIP = IPAddress.Parse(TextBox1.Text)
        GLOINTPORT = "11000"
        udpClient.Connect(GLOIP, GLOINTPORT)
        bytCommand = Encoding.ASCII.GetBytes(txtMessage.Text)
        pRet = udpClient.Send(bytCommand, bytCommand.Length)

        Console.WriteLine(Encoding.ASCII.GetString(bytCommand))
        TextBox2.Text = TextBox2.Text + vbCrLf + "Sent Message: "
        TextBox2.Text = TextBox2.Text & Encoding.ASCII.GetChars(bytCommand) & ""

        TextBox2.Text = TextBox2.Text + vbCrLf
    Catch ex As Exception
        Console.WriteLine(ex.Message)
        TextBox2.Text = TextBox2.Text & vbCrLf & ex.Message
    End Try
    txtMessage.Text = ""
End Sub
```

The above section is of send message button (button 4) which when clicked, sends the message from txtMessage text box to the remote application.

```
Private Sub TextBox2_TextChanged(sender As Object, e As EventArgs) Handles TextBox2.TextChanged
    TextBox2.SelectionStart = TextBox2.TextLength
    TextBox2.ScrollToCaret()


End Sub
```

This part helps the user to automatically navigate to the newly received messages.

```
Private Sub Form1_Closing(ByVal sender As Object, ByVal e As System.ComponentModel.CancelEventArgs) Handles My
    Try
        ThreadReceive.Abort()
        receivingUdpClient.Close()
    Catch ex As Exception
        Console.WriteLine(ex.Message)
    End Try
    AxVideoChatSender1.Disconnect()
    AxVideoChatReceiver1.Disconnect()
    AxVideoChatReceiver2.Disconnect()


    publisher.Client.Close()
    subscriber.Client.Close()

    udpClient.Client.Close()
    Application.Restart()


End Sub
```

This is the code which will be executed on form closing.

```
Private Sub Button5_Click(sender As Object, e As EventArgs) Handles Button5.Click
    Try
        ThreadReceive.Abort()
        receivingUdpClient.Close()
    Catch ex As Exception
        Console.WriteLine(ex.Message)
    End Try
    AxVideoChatSender1.Disconnect()
    AxVideoChatReceiver1.Disconnect()
    AxVideoChatReceiver2.Disconnect()
```

```
448         publisher.Client.Close()
449         subscriber.Client.Close()
450
451         udpClient.Client.Close()
452         Application.Exit()
453
454     End Sub
455     Class
```

This section contains the code for exit application button also in this section the class is closed.

3.6 TESTING:

Testing is a process of executing a program with the intent of finding an error. We can test our project “BIT COMMUNICATION SYSTEM” using various methods but the main objective is that when:

1. The form open’s the sender’s webcams, microphone and computer name is displayed in the combo box.
2. A list box indicates the list of member present in the network.
3. User need to select items/name from the list box and press send button to connect to the other user.
4. On successful connection the video and audio stream start running in the pc.

As we haven’t divided this project into modules so we don’t have to perform unit testing, the only testing we have to do is the overall system testing after each new build is built.

3.7 MAINTAINANCE AND IMPLEMENTATION:

3.7.1 IMPLEMENTATION:

In the **implementation phase**, the team builds the components either from scratch or by composition. Given the architecture document from the design phase and the requirement document from the analysis phase, the team should build exactly what has been requested, though there is still room for innovation and flexibility. For example, a component may be narrowly designed for this particular system, or the component may be made more general to satisfy a reusability guideline. The architecture document should give guidance. Sometimes, this guidance is found in the requirement document.

The implementation phase deals with issues of quality, performance, baselines, libraries, and debugging. The end deliverable is the product itself.

3.7.2 MAINTAINANCE:

Software maintenance is a part of Software Development Life Cycle. Its main purpose is to modify and update software application after delivery to correct faults and to improve performance. Software is a model of the real world. When the real world changes, the software requires alteration wherever possible.

Software maintenance is a vast activity which includes optimization, error correction, and deletion of discarded features and enhancement of existing features. Since these changes are necessary, a mechanism must be created for estimation, controlling and making modifications. The essential part of software maintenance requires preparation of an accurate plan during the development cycle.

In this project application is needed to maintain, specially for the IP Addresses of the connected users, i.e. when any teacher joins the college then the application is needed to update with their PC IP Address and their names in the applications list box. Similarly in case, if any teacher left the college then the application is needed to update, by removing their user names and IP Address from the application.

4. SYSTEM SPECIFICATION

4.1 Hardware Requirement:

In hardware requirement we require all those components which will provide us the platform for the development of the project. The minimum hardware required for the development of this project is as follows-

- **RAM: Minimum 512 MB**
- **Hard Disk: Minimum 160 GB**
- **Processor: Pentium 4 and above**
- **Webcam**
- **Microphone**
- **Ethernet Card**
- **LAN Cable or WLAN**

These are the minimum requirements. Other enhancements are according to needs.

4.2 Software Requirement:

Software's can be defined as programs which run on our computer. It provides the relationship between the human and a computer. It is very important to run software to function the computer. Various software's are needed in this project for its development. Which are as follows-

- **Operating System: Windows XP and higher**
- **.NET FRAMEWORK**

5. SOFTWARE ARCHITECTURE

5.1 SOCKET OVERVIEW:

A socket is an object that represents a low level point to the IP stack. This socket can be opened or closed or one of a set number of intermediate states. A socket can send and receive data down disconnection. Data is generally sent in blocks of few kilobytes at a time for efficiency; each of these blocks are called a **packet**.

All packets that travel over a medium uses the **Internet Protocol (IP).** This means that the source IP Address, destination address must be included in the packet. Most packets also contain a port number. A port is simply a number between 1 and 65,535 that is used to differentiate higher protocols. Ports are important when it comes to programming your own network applications because no two applications can use the same port.

Packets that contain port number comes in two flavor: UDP and TCP/IP. UDP has the lower latency than TCP/IP, especially on startup. Where data integrity is not of the utmost concerned, UDP can prove easier to use than TCP, but it should never be used where data integrity is more important than performance; however, data sent by UDP can sometimes arrive in the wrong order and be effectively useless to the receiver. TCP/IP is more unacceptable than a slow download; however, it is unsuited to internet radio, where the odd sound out of place is more acceptable than long gaps of silence.

5.2 UDP Protocol:

In this Project we have efficiently used the UDP (User Datagram Protocol).

The User Datagram Protocol is an unreliable, connectionless oriented protocol that uses an IP Address for the destination host and a port number to identify the destination application.

The UDP port number is distinct from any physical port on a computer such as a COM port or an I/O port Address. The UDP port is a 16-bit address that exists only for the purpose of passing certain type of datagram information to the correct location above the transport layer of the protocol stack.

A UDP datagram header consists of four fields of 2 bytes each:

1. Source Port Number
2. Destination Port Number
3. Datagram Size

5.3 Reasons for Using UDP

The Best protocols which can be used for this projects are UDP (User Datagram Protocol) And TCP (Transmission Control Protocol) thus we had to make choice between these two.

As we all know that video calling needs high speed data transmission else there will be a lag in communication and **UDP is approximately 3 times faster than TCP** that's why we chose UDP.

Here are some differences between **UDP** and **TCP**:

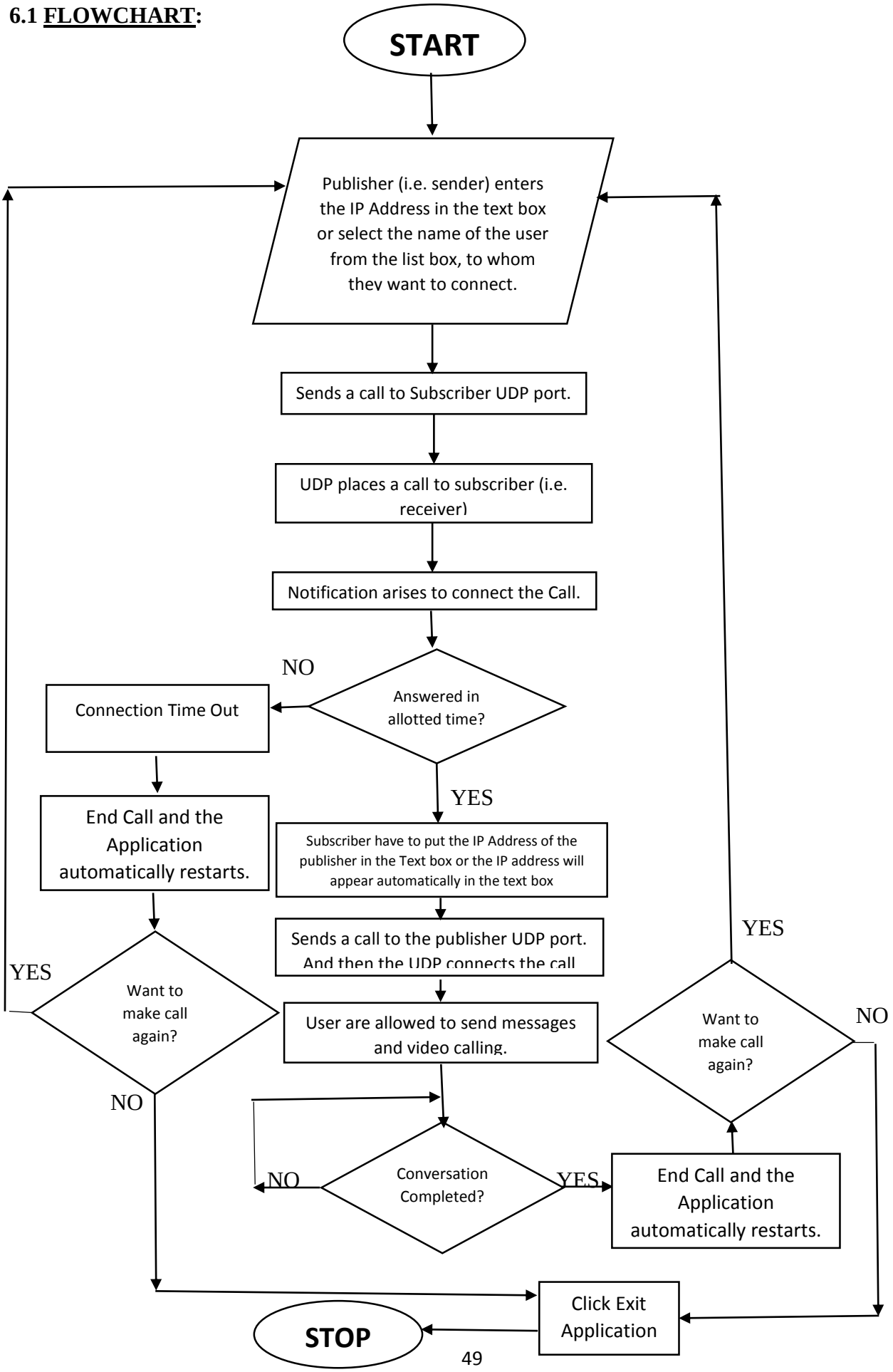
	TCP	**UDP**
Acronym for	Transmission Control Protocol	User Datagram Protocol or Universal Datagram Protocol
Connection	TCP is a connection-oriented protocol.	UDP is a connectionless protocol.
Function	As a message makes its way across the internet from one computer to another. This is connection based.	UDP is also a protocol used in message transport or transfer. This is not connection based which means that one program can send a load of packets to another and that would be the end of the relationship.
Usage	TCP is suited for applications that require high reliability, and transmission time is relatively less critical	UDP is suitable for applications that need fast, efficient transmission, such as games. UDP's stateless nature is also useful for servers that answer small queries from huge numbers of clients.
Use by other protocols	HTTP, HTTPs, FTP, SMTP, Telnet	DNS, DHCP, TFTP, SNMP, RIP, VOIP.

Ordering of data packets	TCP rearranges data packets in the order specified.	UDP has no inherent order as all packets are independent of each other. If ordering is required, it has to be managed by the application layer.
Speed of transfer	The speed for TCP is slower than UDP.	UDP is faster because error recovery is not attempted. It is a "best effort" protocol.
Reliability	There is absolute guarantee that the data transferred remains intact and arrives in the same order in which it was sent.	There is no guarantee that the messages or packets sent would reach at all.
Header Size	TCP header size is 20 bytes	UDP Header size is 8 bytes.
Common Header Fields	Source port, Destination port, Check Sum	Source port, Destination port, Check Sum
Streaming of data	Data is read as a byte stream, no distinguishing indications are transmitted to signal message (segment) boundaries.	Packets are sent individually and are checked for integrity only if they arrive. Packets have definite boundaries which are honored upon receipt, meaning a read operation at the receiver socket will yield an entire message as it was originally sent.
Weight	TCP is heavy-weight. TCP requires three packets to set up a socket connection, before any user data can be sent. TCP handles reliability and congestion control.	UDP is lightweight. There is no ordering of messages, no tracking connections, etc. It is a small transport layer designed on top of IP.

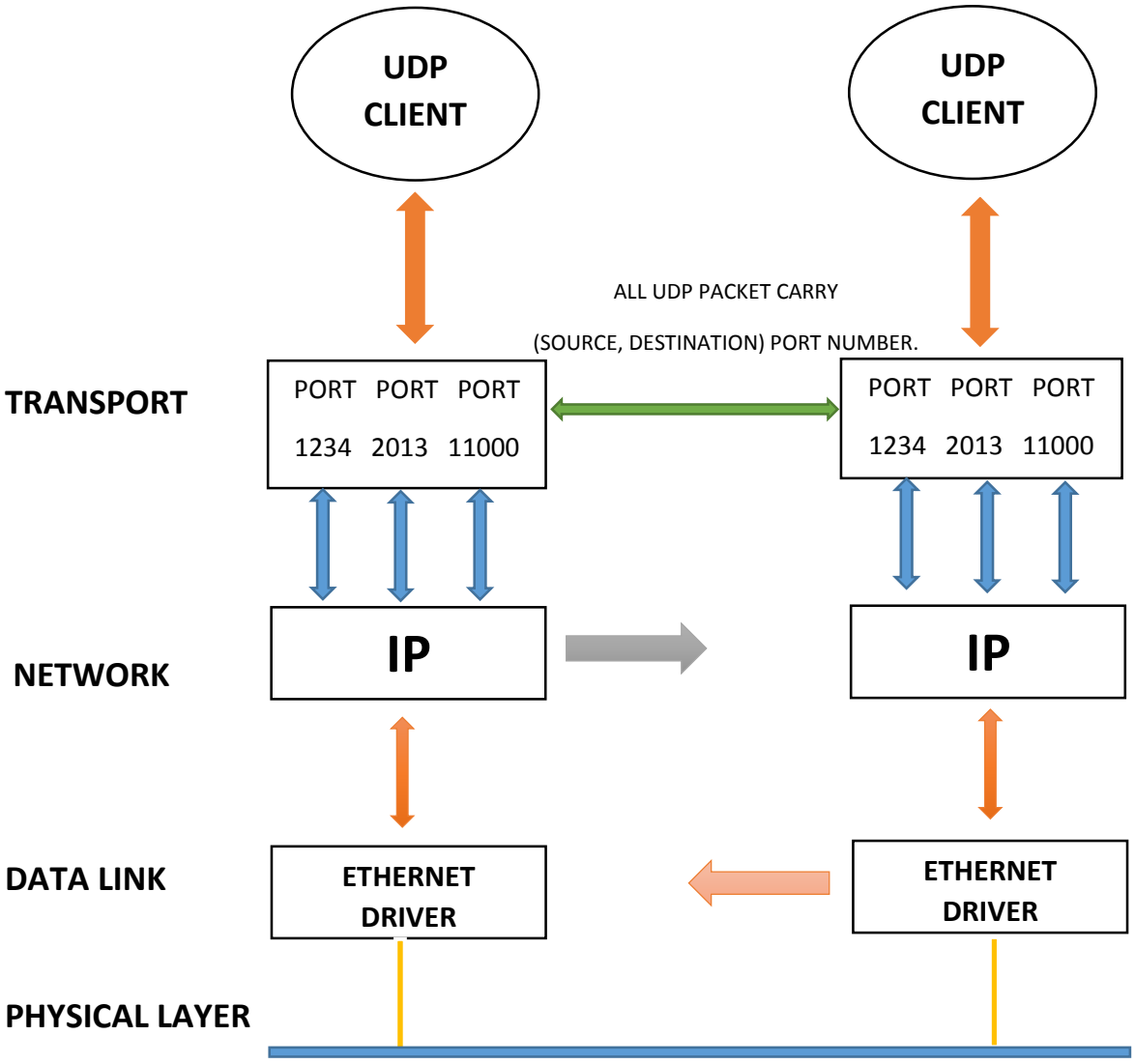
Data Flow Control	TCP does Flow Control. TCP requires three packets to set up a socket connection, before any user data can be sent. TCP handles reliability and congestion control.	UDP does not have an option for flow control
Error Checking	TCP does error checking and error recovery. Erroneous packets are retransmitted from the source to the destination.	UDP does error checking but simply discards erroneous packets. Error recovery is not attempted.
Fields	1. Sequence Number 2. AcK number 3. Data offset 4. Reserved 5. Control bit 6. Window 7. Urgent Pointer 8. Options 9. Padding 10. Check Sum 11. Source port 12. Destination port	1. Length 2. Source port 3. Destination port 4. Check Sum
Acknowledgement	Acknowledgement segments	No Acknowledgment
Handshake	SYN, SYN-ACK, ACK	No handshake (connectionless protocol)

6. SYSTEM DESIGN

6.1 FLOWCHART:

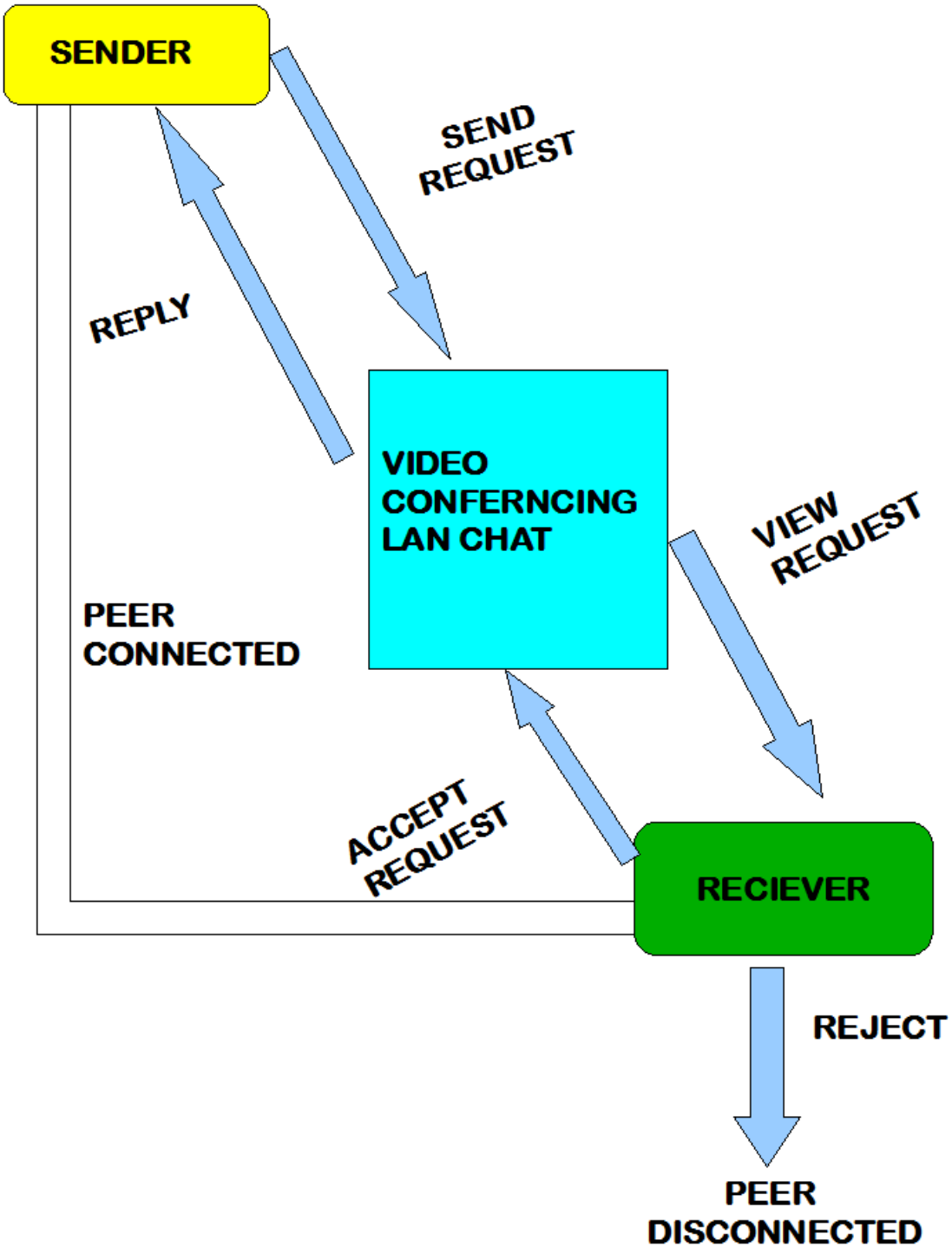


6.2 LAYERS OF UDP:



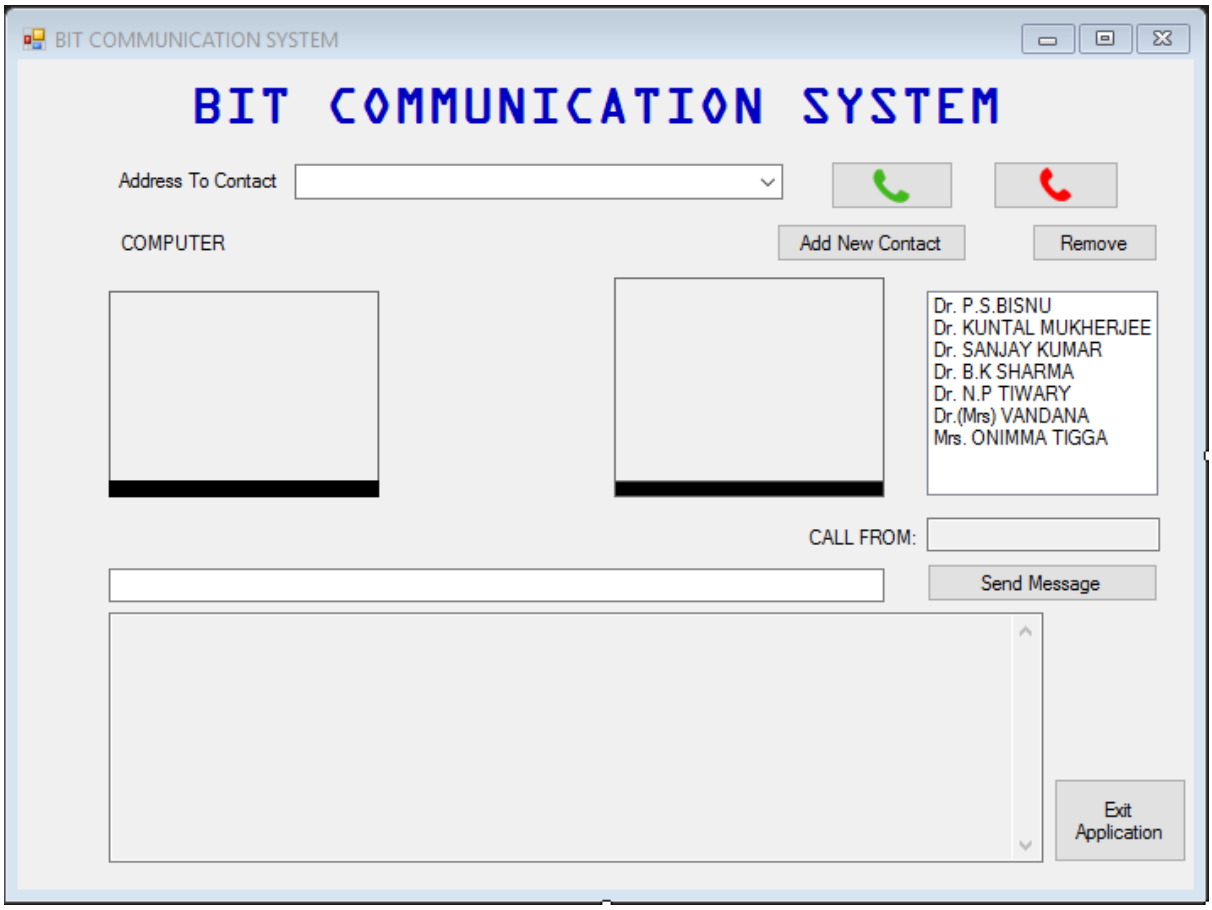
7. ACTIVITY DIAGRAM

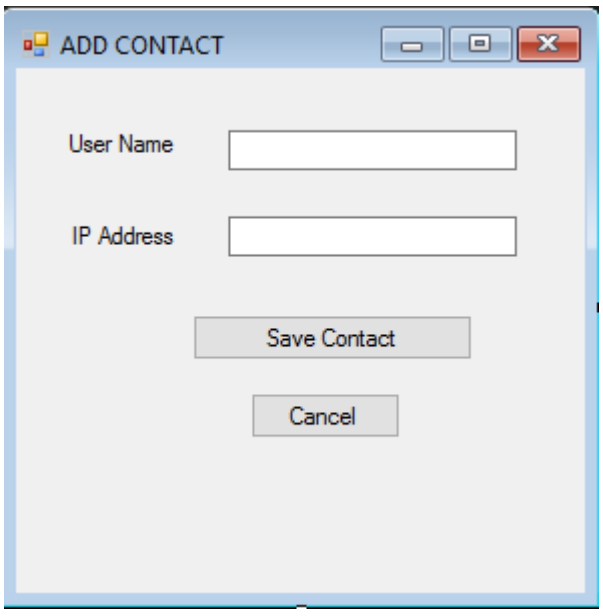
7. ACTIVITY DIAGRAM:

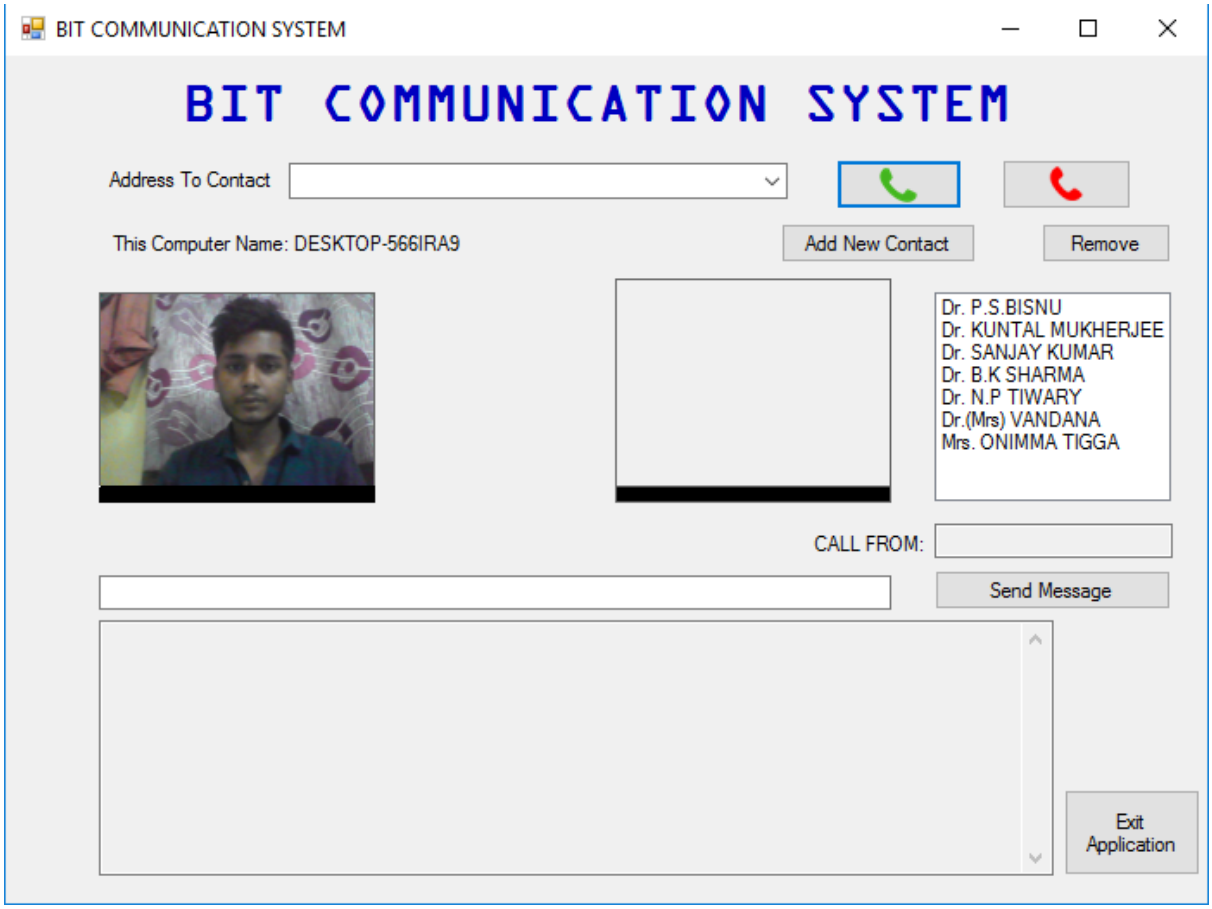


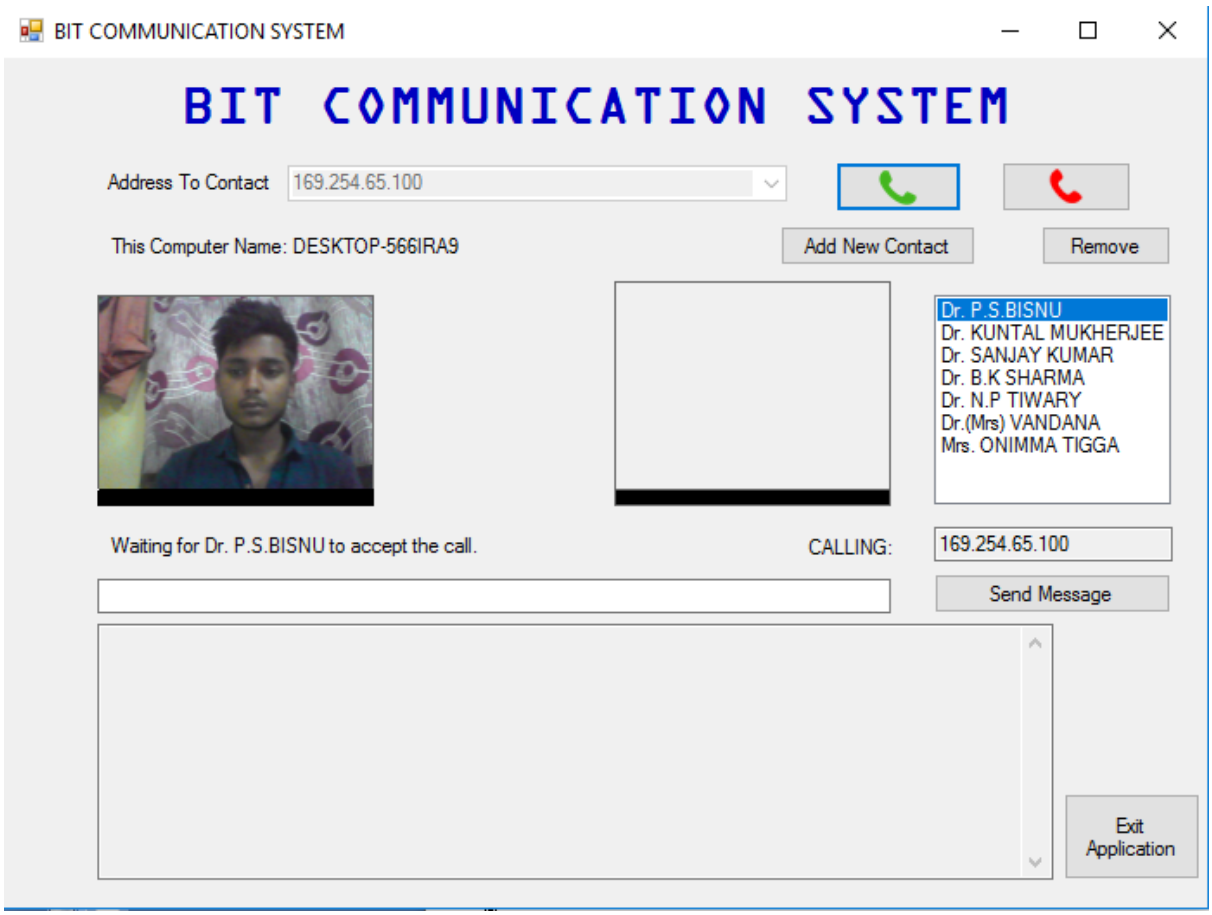
8. SNAPSHOTS

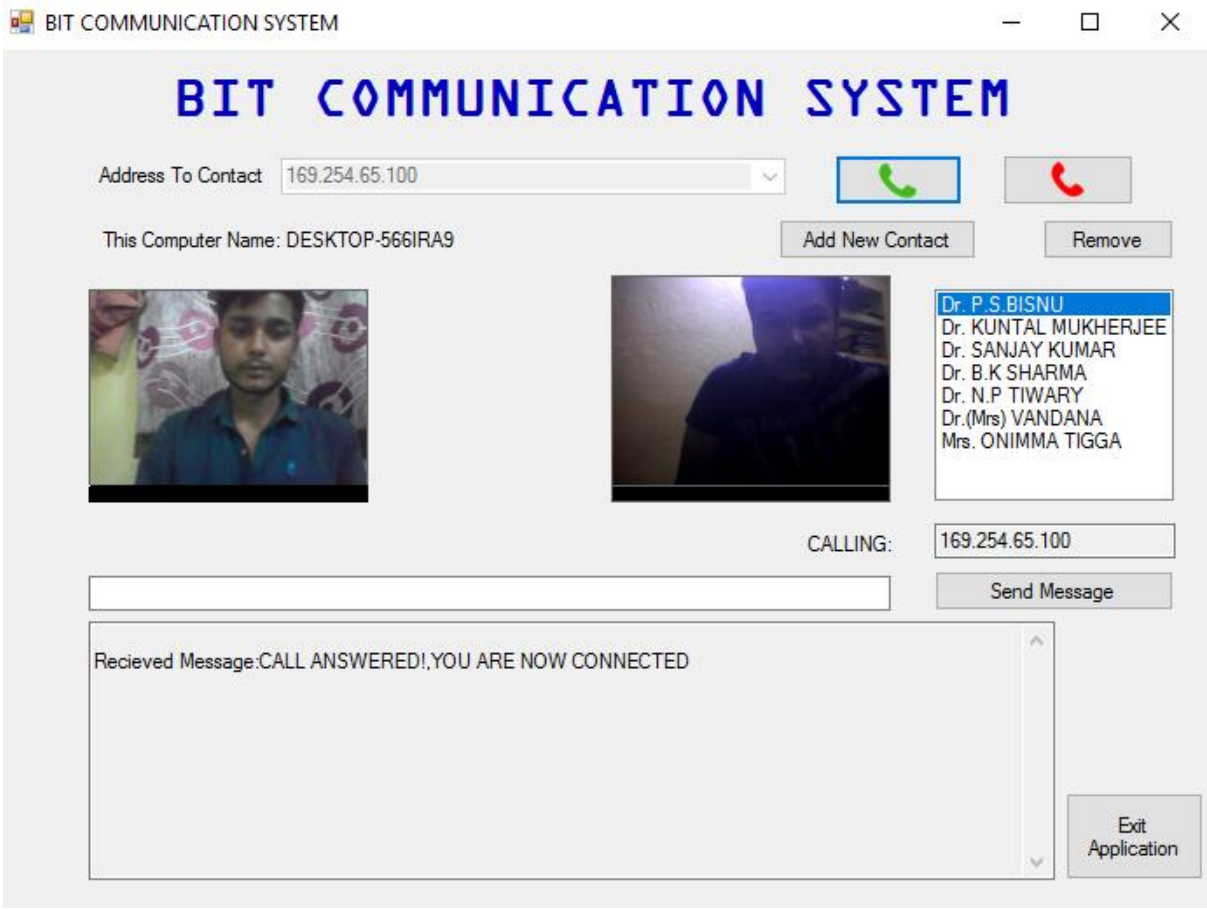
SNAPSHOTS FROM THE PROJECT:

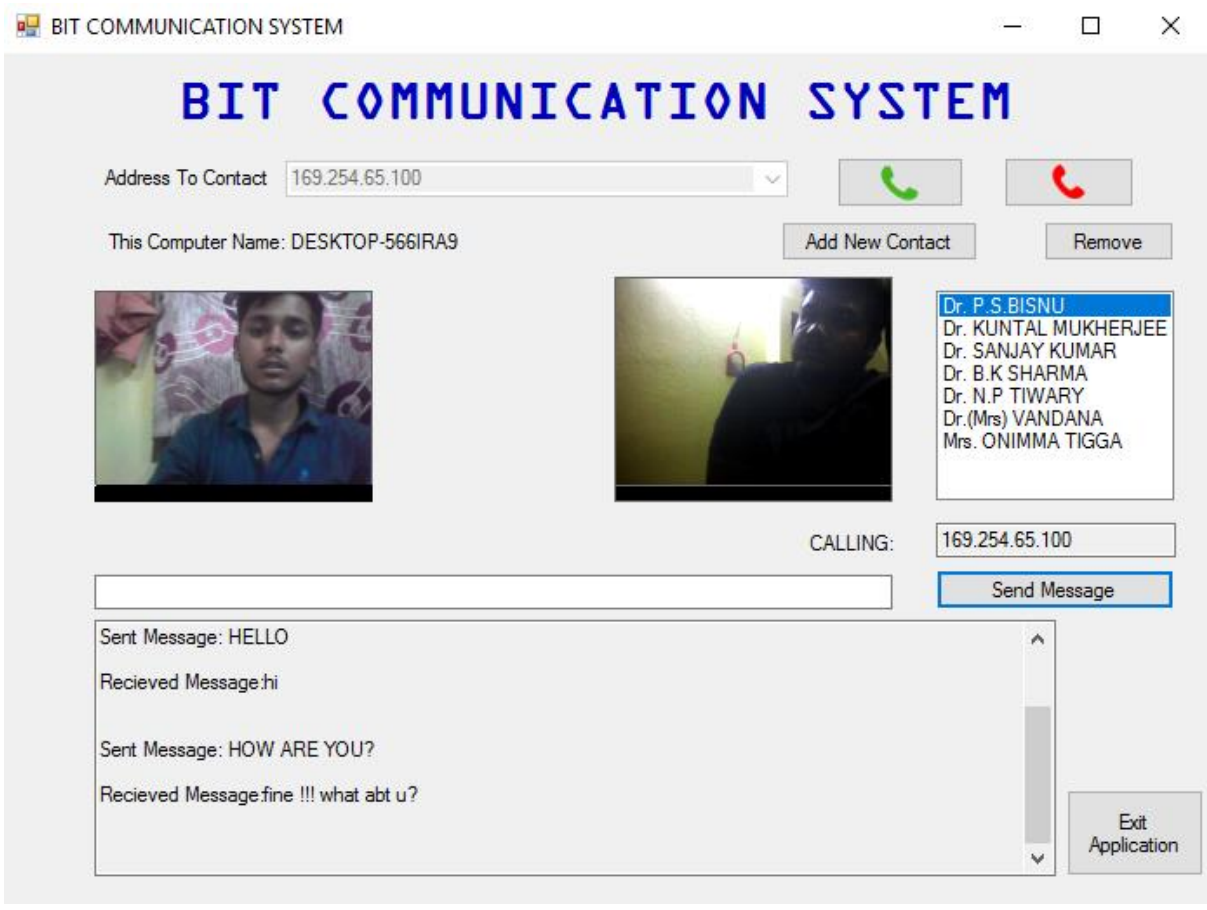





BIT COMMUNICATION SYSTEM
BIT COMMUNICATION SYSTEM
Address To Contact
This Computer Name: DESKTOP-566IRA9
Add New Contact
Remove
Dr. P.S.BISNU
Dr. KUNTAL MUKHERJEE
Dr. SANJAY KUMAR
Dr. B.K SHARMA
Dr. N.P TIWARY
Dr.(Mrs) VANDANA
Mrs. ONIMMA TIGGA
CALL FROM:
Send Message
Exit Application


BIT COMMUNICATION SYSTEM
BIT COMMUNICATION SYSTEM
Address To Contact
169.254.65.100
This Computer Name: DESKTOP-566IRA9
Add New Contact
Remove
Dr. P.S.BISNU
Dr. KUNTAL MUKHERJEE
Dr. SANJAY KUMAR
Dr. B.K SHARMA
Dr. N.P TIWARY
Dr.(Mrs) VANDANA
Mrs. ONIMMA TIGGA
Waiting for Dr. P.S.BISNU to accept the call.
CALLING:
169.254.65.100
Send Message
Exit Application


BIT COMMUNICATION SYSTEM
BIT COMMUNICATION SYSTEM
Address To Contact
169.254.65.100
This Computer Name: DESKTOP-566IRA9
Add New Contact
Remove
Dr. P.S.BISNU
Dr. KUNTAL MUKHERJEE
Dr. SANJAY KUMAR
Dr. B.K SHARMA
Dr. N.P TIWARY
Dr.(Mrs) VANDANA
Mrs. ONIMMA TIGGA
CALLING:
169.254.65.100
Send Message
Recieved Message:CALL ANSWERED!,YOU ARE NOW CONNECTED
Exit Application


BIT COMMUNICATION SYSTEM
BIT COMMUNICATION SYSTEM
Address To Contact
169.254.65.100
This Computer Name: DESKTOP-566IRA9
Add New Contact
Remove
Dr. P.S.BISNU
Dr. KUNTAL MUKHERJEE
Dr. SANJAY KUMAR
Dr. B.K SHARMA
Dr. N.P TIWARY
Dr.(Mrs) VANDANA
Mrs. ONIMMA TIGGA
CALLING:
169.254.65.100
Send Message
Sent Message: HELLO
Recieved Message:hi
Sent Message: HOW ARE YOU?
Recieved Message:fine !!! what abt u?
Exit Application

9. ADVANTAGES

9. ADVANTAGES:

- **Offline Communication**: There is no use of internet connection.
- **Two way chatting and Video Conferencing can be done with the help of the system thorough both "Guided" and "Unguided" media:**
- **Cost Effective:** Since the connection uses no internet. So we don't need to pay for the internet to any ISP (Internet Service Provider).
- **Full Duplex System:** This application works on a full duplex communication system.
- **Uses UDP which is approximately 3 times faster than TCP/IP.**
- One of the attractive features of UDP is that since it does not need to retransmit lost packets nor does it do any connection setup, sending data incurs less delay. This lower delay makes UDP an appealing choice for delay-sensitive applications like audio and video.

- Multicast applications are built on top of UDP since they have to do point to multipoint. Using TCP for multicast applications would be hard since now the sender would have to keep track of retransmissions/sending rate for multiple receivers.

- UDP provides better application level control over what data is sent, since the data is packaged in a UDP segment and immediately passed over to the network layer, hence no-frills segment delivery service is observed.

- There is no need for connection establishment hence no delay (unlike TCP, which requires handshaking before the actual data transfer).

- There is no need to maintain connection state in the end systems (i.e. no need for send and receive buffers, congestion control parameters and sequence and acknowledgement number parameters), hence more active clients could be supported.

- Small packet header overhead for UDP (only 8 bytes) whereas TCP has 20 bytes of header.

- **Relevance**: provides a place for real life examples and experience to be exchanged

- **Community**: over time can develop into a supportive, stimulating community which participants come to regard as the high point of their course
- **Limitless**: you can never predict where the discussion will go; the unexpected often results in increased incidental learning

10. LIMITATIONS

LIMITATIONS:

- **Lack of Personal Interaction:** Some meeting requires a personal touch to be successful. Video Conferencing can be less personal than meeting face to face.
- **We can send around only 65,000 bytes (i.e., (160x122) pixel x 3 bytes) at a time using UDP:** This is the major limitation for the video calling over User Datagram Protocol. The dimension of the picture box should be less than 160 x 122 pixel (approximately). And it is very smaller in size.
- **Technical Problems:** The major disadvantage are the technical difficulties associated with smooth transmission that could result from software, hardware or network failure. Remote connections are sometimes known to be hampered by environmental changes.
- **UDP doesn't provide error control mechanism:**
- **No physical cues**: without facial expressions and gestures or the ability to retract immediately there's a big risk of misunderstanding
- **Information overload**: a large volume of messages can be overwhelming and hard to follow, even stress-inducing

11. FUTURE SCOPE

Future Scope:

1. Conferencing Type Subdivision:
    - One-To-Many
    - Many-To-Many
2. Other transferring system:
    - File Transfer: This will enable the user to send files of different formats to other.
3. Mobile compatibility Application:
    - A mobile Application can also be developed which will perform the same functions, when connected using **WLAN.**
4. More user friendly Application with many other enhancement like muting the call, conversion of call from audio call to video call and vice versa.
5. The Application can also be used by different organizations for communication such as Hospitals, Business Organization, etc.

12. REFERENCES

REFERENCE

1. **VISUAL BASICS.NET 10: THE COMPLETE REFERENCE**
2. **BLACK BOOK (VISUAL BASICS), PETER HOLZNER**
3. **www.codeproject.com**
4. **www.microsoft.com**
5. **www.codeplex.com**
6. **www.github.com**
7. **www.codemonkey.com**
8. **NETWORK PROGRAMMING IN .NET (WITH C# AND VISUAL BASICS .NET), BY FIACH REID.**

13. APPENDIX

SOURCE CODE

```
Imports System.Net.Sockets
Imports System.Threading
Imports TouchlessLib
Imports System.Net
Imports System.Text
Imports System.IO
Imports Microsoft.VisualBasic
Imports System.Runtime.InteropServices
Imports System.Net.Sockets.Socket
Imports System

Public Class Form1
Inherits System.Windows.Forms.Form
Dim subscriber As New UdpClient()
Dim publisher As New UdpClient()
Dim mycomputername As String = Environment.MachineName
Dim mycomputerIP() As System.Net.IPAddress = System.Net.Dns.GetHostAddresses(mycomputername)

Dim Touchless As New TouchlessLib.TouchlessMgr
Dim Camera1 As TouchlessLib.Camera = Touchless.Cameras.ElementAt(0)
Dim picsize As Size = New Size(160, 120)

Public receivingUdpClient As UdpClient
Public RemoteIpEndPoint As New System.Net.IPEndPoint(System.Net.IPAddress.Any, 0)
Public ThreadReceive As System.Threading.Thread

Dim SocketNO As Integer
Dim GLOIP As IPAddress
Dim GLOINTPORT As Integer
Dim bytCommand As Byte() = New Byte() {}
Dim udpClient As New UdpClient
```

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
MyBase.Load

publisher.Client.Blocking = False
subscriber.Client.ReceiveTimeout = 100
subscriber.Client.Blocking = False
subscriber.ExclusiveAddressUse = False
publisher.ExclusiveAddressUse = False
TextBox1.Text = ""

Label3.Text = "This Computer Name: " & Environment.MachineName

Touchless.CurrentCamera = Camera1
Touchless.CurrentCamera.CaptureWidth = picsize.Width
Touchless.CurrentCamera.CaptureHeight = picsize.Height

AxVideoChatReceiver2.ReceiveAudioStream = True
AxVideoChatReceiver2.ReceiveVideoStream = False

AxVideoChatReceiver2.Listen(Convert.ToString(mycomputerIP), 1234)
subscriber.Client.Bind(New Net.IPEndPoint(Net.IPAddress.Any, 2013))
'if nt wrking then in btn1_click

Try
SocketNO = "11000"
receivingUdpClient = New System.Net.Sockets.UdpClient(SocketNO)
ThreadReceive = New System.Threading.Thread(AddressOf ReceiveMessages)
ThreadReceive.Start()

Catch x As Exception
Console.WriteLine(x.Message)
TextBox2.Text = TextBox2.Text & vbCrLf & x.Message
End Try
```

```
End Sub
Public Sub ReceiveMessages()

Try
Dim receiveBytes As [Byte]() = receivingUdpClient.Receive(RemoteIpEndPoint)
txtIP1.Text = RemoteIpEndPoint.Address.ToString
TextBox1.Text = RemoteIpEndPoint.Address.ToString
Dim BitDet As BitArray
BitDet = New BitArray(receiveBytes)

Dim strReturnData As String = System.Text.Encoding.Unicode.GetString(receiveBytes)

TextBox2.Text = TextBox2.Text + vbCrLf + "Recieved Message:"
If (Encoding.ASCII.GetChars(receiveBytes) = "End call") Then
MyBase.Close()
Application.Restart()
End If

TextBox2.Text = TextBox2.Text & Encoding.ASCII.GetChars(receiveBytes) + ""
TextBox2.Text = TextBox2.Text & vbCrLf

TextBox2.Text = TextBox2.Text & vbCrLf
NewInitialize()
Catch e As Exception
Console.WriteLine(e.Message)
End Try
End Sub
Public Sub NewInitialize()
Console.WriteLine("Thread *Thread Receive* reinitialized")
ThreadReceive = New System.Threading.Thread(AddressOf ReceiveMessages)
ThreadReceive.Start()
End Sub
Private Sub Timer1_Tick(ByVal sender As Object, ByVal e As EventArgs) Handles Timer1.Tick
```

```
Try
Dim bitmapz As Bitmap = New Bitmap(picsize.Width, picsize.Height)
bitmapz = Touchless.CurrentCamera.GetCurrentImage
PictureBox1.Image = bitmapz

Dim sendbytes() As Byte
bytesfromimage(sendbytes, bitmapz)

publisher.Send(sendbytes, sendbytes.Length)
Catch ex As Exception
End Try


Try
Dim ep As System.Net.IPEndPoint = New System.Net.IPEndPoint(System.Net.IPAddress.Any, 0)
Dim rcvbytes() As Byte = subscriber.Receive(ep)
Dim bitmapz As Bitmap = New Bitmap(picsize.Width, picsize.Height)
imagefrombytes(rcvbytes, bitmapz)
PictureBox2.Image = bitmapz

Catch ex As Exception
End Try

End Sub

Private Sub imagefrombytes(ByRef bytez() As Byte, ByRef piccolor As Bitmap)
Dim rect As New Rectangle(0, 0, piccolor.Width, piccolor.Height)
Dim bmpData As System.Drawing.Imaging.BitmapData = piccolor.LockBits(rect,
Drawing.Imaging.ImageLockMode.ReadWrite, Imaging.PixelFormat.Format32bppRgb)
Dim ptr As IntPtr = bmpData.Scan0
Dim bytes As Integer = bmpData.Stride * piccolor.Height
Dim rgbValues(bytes - 1) As Byte
System.Runtime.InteropServices.Marshal.Copy(ptr, rgbValues, 0, bytes)
```

```
Dim secondcounter As Integer
Dim tempred As Integer
Dim tempblue As Integer
Dim tempgreen As Integer
Dim tempalpha As Integer
secondcounter = 0
While secondcounter < rgbValues.Length
tempblue = rgbValues(secondcounter)
tempgreen = rgbValues(secondcounter + 1)
tempred = rgbValues(secondcounter + 2)
tempalpha = rgbValues(secondcounter + 3)
tempalpha = 255

tempred = bytez(((secondcounter * 0.25) * 3) + 0)
tempgreen = bytez(((secondcounter * 0.25) * 3) + 1)
tempblue = bytez(((secondcounter * 0.25) * 3) + 2)

rgbValues(secondcounter) = tempblue
rgbValues(secondcounter + 1) = tempgreen
rgbValues(secondcounter + 2) = tempred
rgbValues(secondcounter + 3) = tempalpha

secondcounter = secondcounter + 4
End While
System.Runtime.InteropServices.Marshal.Copy(rgbValues, 0, ptr, bytes)
piccolor.UnlockBits(bmpData)
Label5.Text = ""
End Sub
Private Sub bytesfromimage(ByRef bytez() As Byte, ByRef piccolor As Bitmap)
Dim rect As New Rectangle(0, 0, piccolor.Width, piccolor.Height)
Dim bmpData As System.Drawing.Imaging.BitmapData = piccolor.LockBits(rect,
Drawing.Imaging.ImageLockMode.ReadWrite, Imaging.PixelFormat.Format32bppRgb)
Dim ptr As IntPtr = bmpData.Scan0
Dim bytes As Integer = bmpData.Stride * piccolor.Height
```

```
Dim rgbValues(bytes - 1) As Byte
System.Runtime.InteropServices.Marshal.Copy(ptr, rgbValues, 0, bytes)

Dim secondcounter As Integer
Dim tempred As Integer
Dim tempblue As Integer
Dim tempgreen As Integer
Dim tempalpha As Integer
secondcounter = 0
Dim bytelist As List(Of Byte) = New List(Of Byte)

While secondcounter < rgbValues.Length
tempblue = rgbValues(secondcounter)
tempgreen = rgbValues(secondcounter + 1)
tempred = rgbValues(secondcounter + 2)
tempalpha = rgbValues(secondcounter + 3)
tempalpha = 255

bytelist.Add(tempred)
bytelist.Add(tempgreen)
bytelist.Add(tempblue)

rgbValues(secondcounter) = tempblue
rgbValues(secondcounter + 1) = tempgreen
rgbValues(secondcounter + 2) = tempred
rgbValues(secondcounter + 3) = tempalpha

secondcounter = secondcounter + 4
End While

System.Runtime.InteropServices.Marshal.Copy(rgbValues, 0, ptr, bytes)

piccolor.UnlockBits(bmpData)
```

```
Dim bytearray(bytelist.Count - 1) As Byte
For i = 0 To bytelist.Count - 1
bytearray(i) = bytelist(i)
Next
bytez = bytearray

End Sub

Private Sub Button1_Click(ByVal sender As Object, ByVal e As EventArgs) Handles Button1.Click
publisher.Connect(TextBox1.Text, 2013)

AxVideoChatSender1.VideoDevice = 0
AxVideoChatSender1.AudioDevice = 0
AxVideoChatSender1.VideoFormat = 0
AxVideoChatSender1.FrameRate = 15
AxVideoChatSender1.VideoBitrate = 128000
AxVideoChatSender1.AudioComplexity = 0
AxVideoChatSender1.AudioQuality = 8
AxVideoChatSender1.SendAudioStream = True
AxVideoChatSender1.SendVideoStream = False

AxVideoChatSender1.Connect(TextBox1.Text, 1234)

AxVideoChatReceiver1.ReceiveAudioStream = True
AxVideoChatReceiver1.ReceiveVideoStream = False

AxVideoChatReceiver2.ReceiveAudioStream = True
AxVideoChatReceiver2.ReceiveVideoStream = False

AxVideoChatReceiver2.Listen(Convert.ToString(mycomputerIP), 1234)
If (ListBox1.SelectedIndex >= 0) Then
Label5.Text = "Waiting for " + ListBox1.SelectedItem + " to accept the call."
Else
Label5.Text = "Waiting for " + TextBox1.Text + " to accept the call."
```

```
End If
TextBox1.Enabled = False
Dim pRet As Integer

Try
GLOIP = IPAddress.Parse(TextBox1.Text)
GLOINTPORT = "11000"
udpClient.Connect(GLOIP, GLOINTPORT)
If (TextBox2.Text = "") Then
bytCommand = Encoding.ASCII.GetBytes("ANSWER THE CALL..")
Label6.Text = "CALLING:"
txtIP1.Text = TextBox1.Text

Else
bytCommand = Encoding.ASCII.GetBytes("CALL ANSWERED!,YOU ARE NOW CONNECTED")
Label6.Text = "Connected:"
TextBox2.Text = TextBox2.Text + "YOU ANSWERED THE CALL"
End If
pRet = udpClient.Send(bytCommand, bytCommand.Length)

Console.WriteLine(Encoding.ASCII.GetString(bytCommand))

Catch ex As Exception
Console.WriteLine(ex.Message)
TextBox2.Text = TextBox2.Text & vbCrLf & ex.Message
End Try
End Sub

Private Sub Label4_Click(sender As Object, e As EventArgs)
End Sub

Private Sub Button2_Click(sender As Object, e As EventArgs) Handles Button2.Click
Form2.Show()
```

```
End Sub

Private Sub SaveFileDialog2_FileOk(sender As Object, e As System.ComponentModel.CancelEventArgs)
Handles SaveFileDialog2.FileOk

End Sub

Private Sub TextBox1_SelectedIndexChanged(sender As Object, e As EventArgs) Handles
TextBox1.TextChanged

End Sub

Private Sub btnremove_Click(sender As Object, e As EventArgs) Handles btnremove.Click

ListBox1.Items.Remove(ListBox1.SelectedItem)

TextBox1.Text = " "

End Sub

Private Sub ListBox1_SelectedIndexChanged(sender As Object, e As EventArgs) Handles
ListBox1.SelectedIndexChanged

If (ListBox1.SelectedIndex = 0) Then

TextBox1.Text = "169.254.65.100"

End If

If (ListBox1.SelectedIndex = 1) Then

TextBox1.Text = "169.254.222.209"

End If

If (ListBox1.SelectedIndex = 2) Then

TextBox1.Text = "169.254.193.89"

End If

If (ListBox1.SelectedIndex = 3) Then

TextBox1.Text = "192.168.1.1"

End If

If (ListBox1.SelectedIndex = 4) Then

TextBox1.Text = "192.0.0.1"

End If

If (ListBox1.SelectedIndex = 5) Then

TextBox1.Text = "169.0.0.2"

End If
```

```
If (ListBox1.SelectedIndex = 6) Then
TextBox1.Text = "This Index is Empty"
End If
End Sub

Private Sub Button3_Click(sender As Object, e As EventArgs) Handles Button3.Click
Dim pRet As Integer

Try
GLOIP = IPAddress.Parse(TextBox1.Text)
GLOINTPORT = "11000"
udpClient.Connect(GLOIP, GLOINTPORT)
bytCommand = Encoding.ASCII.GetBytes("End call")
pRet = udpClient.Send(bytCommand, bytCommand.Length)

Console.WriteLine(Encoding.ASCII.GetString(bytCommand))
TextBox2.Text = TextBox2.Text + vbCrLf + "Sent Message: "
TextBox2.Text = TextBox2.Text & Encoding.ASCII.GetChars(bytCommand) & ""

TextBox2.Text = TextBox2.Text + vbCrLf
Catch ex As Exception
Console.WriteLine(ex.Message)
TextBox2.Text = TextBox2.Text & vbCrLf & ex.Message
End Try
Try
ThreadReceive.Abort()
receivingUdpClient.Close()
Catch ex As Exception
Console.WriteLine(ex.Message)
End Try
AxVideoChatSender1.Disconnect()
AxVideoChatReceiver1.Disconnect()
AxVideoChatReceiver2.Disconnect()
```

```
publisher.Client.Close()
subscriber.Client.Close()

udpClient.Client.Close()

Application.Restart()
End Sub

Private Sub Label2_Click(sender As Object, e As EventArgs) Handles Label2.Click

End Sub

Private Sub Button4_Click(sender As Object, e As EventArgs) Handles Button4.Click
Dim pRet As Integer

Try
GLOIP = IPAddress.Parse(TextBox1.Text)
GLOINTPORT = "11000"
udpClient.Connect(GLOIP, GLOINTPORT)
bytCommand = Encoding.ASCII.GetBytes(txtMessage.Text)
pRet = udpClient.Send(bytCommand, bytCommand.Length)

Console.WriteLine(Encoding.ASCII.GetString(bytCommand))
TextBox2.Text = TextBox2.Text + vbCrLf + "Sent Message: "
TextBox2.Text = TextBox2.Text & Encoding.ASCII.GetChars(bytCommand) & ""

TextBox2.Text = TextBox2.Text + vbCrLf
Catch ex As Exception
Console.WriteLine(ex.Message)
TextBox2.Text = TextBox2.Text & vbCrLf & ex.Message
End Try
txtMessage.Text = ""
End Sub
```

```
Private Sub txtMessage_TextChanged(sender As Object, e As EventArgs) Handles txtMessage.TextChanged
End Sub

Private Sub TextBox2_TextChanged(sender As Object, e As EventArgs) Handles TextBox2.TextChanged
TextBox2.SelectionStart = TextBox2.TextLength
TextBox2.ScrollToCaret()

End Sub

Private Sub TextBox1_SelectedIndexChanged_1(sender As Object, e As EventArgs) Handles
TextBox1.SelectedIndexChanged

End Sub
Private Sub Form1_Closing(ByVal sender As Object, ByVal e As
System.ComponentModel.CancelEventArgs) Handles MyBase.Closing

Try
ThreadReceive.Abort()
receivingUdpClient.Close()
Catch ex As Exception
Console.WriteLine(ex.Message)
End Try
AxVideoChatSender1.Disconnect()
AxVideoChatReceiver1.Disconnect()
AxVideoChatReceiver2.Disconnect()

publisher.Client.Close()
subscriber.Client.Close()

udpClient.Client.Close()
Application.Restart()
End Sub
```

```
Private Sub Button5_Click(sender As Object, e As EventArgs) Handles Button5.Click

Try
ThreadReceive.Abort()
receivingUdpClient.Close()
Catch ex As Exception
Console.WriteLine(ex.Message)
End Try
AxVideoChatSender1.Disconnect()
AxVideoChatReceiver1.Disconnect()
AxVideoChatReceiver2.Disconnect()

publisher.Client.Close()
subscriber.Client.Close()

udpClient.Client.Close()
Application.Exit()

End Sub
End Class
```

14. CONCLUSION

CONCLUSION

Video conferencing has its application in almost every field including **Education, Business, Entertainment, etc.** Our main goal towards this project "BIT COMMUNICATION SYSTEM" is: In any college or institute, the education, study or planning for some events can be improved by increasing interaction between students with teachers or teacher with another teacher. The students may have different queries which they want to discuss with the teachers but the teacher may not available or free at that time. For some discussion or planning something the teacher may need to discuss with one another and can serve well to the students or the organization.

This problem can be solved by providing a software solution to them which will help them to communicate with each other without leaving the computer or cabins, more specifically a video conferencing or video calling system and messaging.

We have learnt a lot in the period of making the project. Since the advancement of technology is taking place day by day, we have to be updated with it. We had face many difficulties in making this project like; in feasibility study selection of the right alternatives, integration of the audio stream, video stream and messaging stream into a single module and the notification control of the program, activation and closing of the application, end of the call, connection of the call, etc. But finally we came up with a well working project which is capable of doing video calling and messaging. And we also have different future extensions:

6. Conferencing Type Subdivision:
    - One-To-Many
    - Many-To-Many
7. Other transferring system:
    - File Transfer: This will enable the user to send files of different formats to other.
8. Mobile compatibility Application:
    - A mobile Application can also be developed which will perform the same functions, when connected using **WLAN.**

9. More user friendly Application with many other enhancement like muting the call, conversion of call from audio call to video call and vice versa.
10. The Application can also be used by different organizations for communication such as Hospitals, Business Organization, etc.

We have used VB.Net for programming because it creates a stand-alone application, robust application, etc. This application can run with very less hardware and software configuration.

We had used User Datagram Protocol (UDP) for the transmission of data from one end user to another end user. Since UDP is about 3 times faster than TCP that's why we have selected this.

We had also researched for WebRTC (Web Real Time Communication) for this project but we didn't find it feasible because it requires more software and hardware configuration than required in UDP.

So we chose UDP to go further with this project, and here we came with the conclusion, a developed application in which we can do video calling and text messaging with a good user interpretation.

15. BIBLIOGRAPHY

BIRLA INSTITUTE OF TECHNOLOGY

Extension Center Lalpur, Ranchi

BIBLIOGRAPHY

BOOKS:

[1] Fiach Reid*, Network Programing in .Net*, Digital Press, 2004.

[2] Peter Holzner, *Blackbook Visual Basic,* 2005.

[2] Vineet Jaiswal, *ConnectUs Conferencing System,* IT Department, TERNA Engineering College MUMBAI UNIVERSITY.

[3] S Prachat, D. Hardman, "*Voice Quality in Converging Telephony and IP Networks*," Agilant Technologies Whitepaper, August 2002, pp. 11.

WEBSITES:

[1] www.howstuffworks.com/start in 1998 at a college proffesor's kitchen table.

[2] www.microsoft.com/Bill Gates, Paul Allen founded in April 4, 1975

[3] www.laptrinhvb.net/

[4] https://www.youtube.com/watch?v=eMZiaJ8HAJc

[5] www.codeplex.com/Launched May 2006