CHAPTER 1

INTRODUCTION

1.1 About project

Intralan text and voice communication application is a chat messenger implemented in networks like in an office or a college lab. It provides centralized server access and records no history of previous chats eliminating the threat of privacy leaks.

The application has server centric architecture where a central server is responsible for the whole routing of data connecting nodes in a network. The server can be hosted in any of the computers connected to the network where the LanLink network will be hosted. Each client will be initialized by the user by setting up the server address which makes the server portable within the network. LanLink uses TCP for text communication and UDP for voice communication. It also offers a broadcast option to send emergency messages or news updates to all the computers connected to the network.

Purpose is to make it simple and user friendly. When the software is opened user will get to see a list of devices connected to LAN through our software enabling the user to communicate with them through voice or text.

1.2 EXISTING AND PROPOSED SYSTEM

Currently there are other existing applications providing LAN communication like Tonic, Squiggle or Slack. Tonic and Squiggle both have a clunky unintuitive user interface and run on peer-to-peer mode. The developed application improves it by using a server centric architecture and very user intuitive simple user interface. Server centric architecture makes it easier to scale as the need arises and also for future enhancements. Server centric architecture also makes load handling easier. Slack is a freeware and our project will be made open source by uploading the code into GitHub helping to make it better with time by adding new features or customizing it as per the needs of the ones implementing it with better community support.

CHAPTER 2

LITERATURE SURVEY

2.1 An Introduction to Network Programming: Python Way

IEEE DISTRIBUTED SYSTEMS ONLINE 1541-4922 © 2005 Published by the IEEE

Minor Gordon, Technische Universität Berlin

With its batteries-included suite of client-server protocol implementations, C integration, and third-party tools and libraries, Python is ideal for prototyping and deploying network applications. Python's runtime interpretation drastically shortens the turnaround from modification to execution, a significant advantage when most changes are incremental and most problems appear at runtime. Python's clean syntax, dynamic typing, integrated exception handling, and object facilities have made it a widely used interpreted language. With the emergence of large-scale-network tools and applications, such as the Twisted framework, BitTorrent and Zope content management system, Python has proven to be a legitimate platform for network-oriented projects in a space that C has traditionally dominated.

In Python, sockets become objects with the familiar connect(), send(), and recv() as members, and error returns are thrown as exceptions. Python's native types (such as tuples, lists and strings, and dicts) and multiple return values help eliminate much of the Unix detritus that plagues the standard Posix interfaces.

2.2 Comparing Python to other languages

https://www.python.org/doc/essays/comparisons/

JAVA

Python programs are generally expected to run slower than Java programs, but they also take much less time to develop. Python programs are typically 3-5 times shorter than equivalent Java programs. This difference can be attributed to Python's built-in high-level data types and its dynamic typing. For example, a Python programmer wastes no time declaring the types of arguments or variables, and Python's powerful polymorphic list and dictionary types, for which rich syntactic support is built straight into the language, find a use in almost every Python program. Because of the run-time typing, Python's run time must work harder than Java's.

JAVASCRIPT

Python's "object-based" subset is roughly equivalent to JavaScript. Like JavaScript (and unlike Java), Python supports a programming style that uses simple functions and variables without engaging in class definitions. However, for JavaScript, that's all there is. Python, on the other hand, supports writing much larger programs and better code reuse through a true object-oriented programming style, where classes and inheritance play an important role.

PERL

Python and Perl come from a similar background (Unix scripting, which both have long outgrown), and sport many similar features, but have a different philosophy. Perl emphasizes support for common application-oriented tasks, e.g. by having built-in regular expressions, file scanning and report generating features. Python emphasizes support for common programming methodologies such as data structure design and object-oriented programming, and encourages programmers to write readable (and thus maintainable) code by providing an elegant but not overly cryptic notation. As a consequence, Python comes close to Perl but rarely beats it in its original application domain; however Python has applicability well beyond Perl's niche

C++

Almost everything said for Java also applies for C++, just more so: where Python code is typically 3-5 times shorter than equivalent Java code, it is often 5-10 times shorter than equivalent C++ code! Anecdotal evidence suggests that one Python programmer can finish in two months what two C++ programmers can't complete in a year. Python shines as a glue language, used to combine components written in C++.

2.3 The technology behind Tornado, FriendFeed's web server

http://backchannel.org/blog/tornado

- Bret Taylor, September 10 2009

This talks about the Tornado web server which is behind the FriendFeed, a social aggregator, currently owned by Facebook. It has been made open source. It is a python framework. The blog talks about why it was created and it's features explaining the requirements to make a high scalable server.

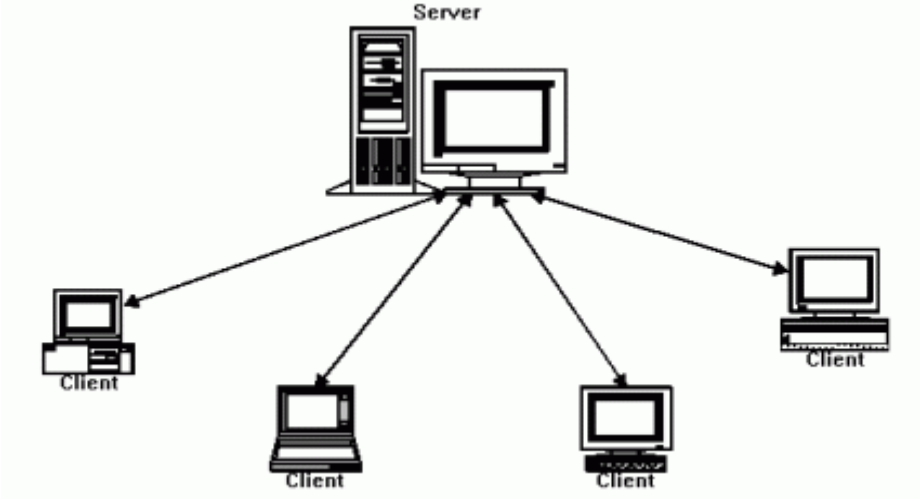
Tornado's major feature involves a large number of standing connections afforded by the non-blocking I/O programming style and epoll. Some other major features of Tornado are the that it includes site building features like cookies,third party authentication,etc and also higher performance. The blog also talks in detail the third party authentication and asynchronous requests. Tornado comes with built-in support for authenticating with Facebook Connect, Twitter, Google, and FriendFeed in addition to OAuth and OpenID. All of the authentication methods support a relatively uniform interface so that one doesn't need to understand all of the intricacies of the different authentication/authorization protocols to leverage them on his site. Tornado assumes that any connection is not asynchronous to make it easy to setup connections and handlers. This makes it the responsible of Tornado to manage closing of connections thus reducing load on developers. But it also lets them be made asynchronous by the use of concept of decorators as seen in python which lets developers' custom manage the operation. Thus it justifies the use of Tornado in FriendFeed and promotes its adoption.

2.4 Client-server model

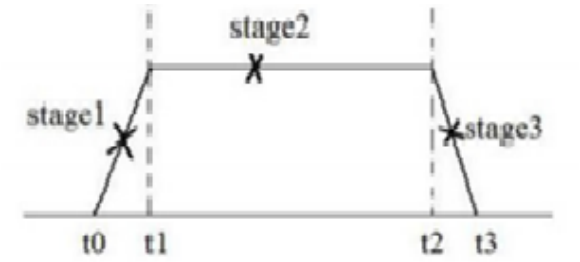
UDP Based Chat Application by Akshit Malhotra, Vaibhav Sharma, Prateek Gandhi, Dr. Neetesh Purohit, 978-1-4244-6349-7/10/c 2010 IEEE.

Server: We maintain a thread at server side which receives various incoming clients request. It stores the information of the clients i.e. IP address and name in a list. This list is then broadcasted to all the users. Contrary to this, whenever a client logs out, it deletes that particular client entry from its list and updates it accordingly. It also keeps track of various chat rooms, providing a group IP to each multicast group.

Client: It first registers itself by sending its username to the server and starts a thread which receives list of all online users from server. Each user has an option of establishing a peer to peer communication with any of the online user, sending a file and creates a room where users selected by him interact with each other.



Failure Model



During our implementation, we incurred various kinds of failures for which we proposed the solution in the following manner. As depicted in the Fig 7, we have discussed both system and network failure. In system failure, either a server, sender or receiver may get failed due to some unforeseen Figure 7. Failure Model reasons. We have discussed following stages in our failure model-

• **Stage 1:** It belongs from time t0 to t1 which corresponds to buffering time of the packet at sender side. During this stage, if a sender gets failed, then packet would not be sent and gets stored in the queue. This packet will be send only when sender side recovers back. Next, if receiver gets failed, then packet will not be received

• **Stage 2:** It belongs from time t1 to t2 which corresponds to the time, packet is in the network i.e. the packet is sent from the buffer of sender but not yet received. If at this stage, sender gets failed, then no matter what message will be received. Next, if receiver gets failed, then packet will not be received. But this information about receiver failure is not known to sender. For this problem, we have implemented acknowledgment of each packet we send from sender side. If we don't receive the acknowledgement at the sender side, then it means receiver has failed. By this, we do know about receiver status at the sender side.

• **Stage 3:** It belongs from time t2 to t3 in which packet has now arrived in the buffer of the receiver. At this stage, failure at sender won't have any impact on the packet because it has already arrived in the receiver buffer. Next, if receiver gets failed, then this packet would be recovered when the receiver comes up again.

During all these stages, if server gets failed then a message informing about server failure would be shown on each client in the network.

CHAPTER 3

SYSTEM REQUIREMENTS SPECIFICATION

A System Requirements Specification is a complete description of the behavior of the system to be developed. A requirement is a function that a system must perform and a desired characteristic of a system. The outcome of the SRS phase is the system requirement specification documents, which describes the complete external behavior of the proposed system. It includes the functional and non-functional requirement for the software to be developed. Requirements must be measurable, testable related to identified needs or opportunities, and defined to a level of detail sufficient for system design.

The process of establishing the services the system should provide and the constraints under which it must operate is called requirements engineering. Specifications of requirement engineering process are requirements definition, requirements specification and software specification. In system engineering and requirements engineering, a non-functional requirement is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviors. This should be contrasted with functional requirements that define specific behavior or functions. The plan for implementing functional requirements is detailed in the system design. The plan for implementing non-functional requirements is detailed in the system architecture.

3.1 FUNCTIONAL REQUIREMENTS

Functional requirements capture the intended behavior of the system. This behavior may be expressed as services, tasks or functions the system is required to perform. Functional requirements are very important system requirements such as technical specifications, system design parameters and guidelines, data manipulation, data processing and calculation modules etc. The following are the functional requirements for the proposed system:

- Create a socket to connect client and server to establish connection.
- Function to locate all the users active in the network.

- Send the details to the client.
- Request the server to obtain details of other client.
- Server generates the required details in the client.
- Server informs the other client about the need for connection.
- Both the clients create socket for client to client communication.

3.2 Non Functional Requirements

Non Functional Requirements are the constraints on the services or functions offered by the system. They include timing constraints on the development process, standards, etc. Qualities, that are non-functional requirements, can be divided into two main categories:

1. Execution qualities, such as security and usability, which are observable at run time.

2. Evolution qualities, such as testability, maintainability, extensibility and scalability, which are embodied in the static structure of the software system.

3.3 SYSTEM REQUIREMENTS

The hardware and software components of a computer system are required to install and use software efficiently. System requirements for operating systems will be hardware components, while other application software will list both hardware and operating system requirements. System requirements are most commonly seen listed as *minimum* and *recommended* requirements. The minimum system requirements need to be met for the software to run at all on your system.

3.3.1 HARDWARE REQUIREMENTS

- Intel Pentium 2 onwards 1.96 GHz
- RAM 1GB
- Mike Voice Input
- Speaker/Headphone Voice Output
- Ethernet ports and cables

3.3.2 SOFTWARE REQUIREMENTS

- PyCharm IDE.
- Windows XP/7/8
- MySQL
- Programming language-Python

CHAPTER 4

SYSTEM ANALYSIS

The task of system analysis is to identify limitations of the existing system and to establish in detail what the proposed system will do. The analysis of an activity, procedure, method, technique or business to determine what must be accomplished and how the necessary operations may best be accomplished. The principal objective of the system analysis phase is the specification of what the system needs to do to meet the requirements of end users. In the system design phase, such specifications are converted to a hierarchy of charts that define the data required and the processes to be carried out on the data, so that they can be expressed as instructions of a computer program. Many information systems are implemented with generic software, rather than with such custom-built programs.

Characteristics of systems analysis include:

- Establishing the objectives of the new system, conducting an analysis of its costs and the benefits to be derived from it, and outlining the process of systems implementation.
- Detailed systems analysis must also establish who the system users are, what information they should get and in what form, and how this information will be obtained from the incoming data and from the databases.

System analysis can be categorized into four parts:

- Information gathering
- Applying tools for the structured analysis
- Feasibility study
- Cost analysis

4.1 INFORMATION GATHERING

Techniques for gathering information during systems analysis can be grouped into four categories. A combination of these approaches is usually employed. They include, asking the users, deriving from an existing system, deriving from the analysis of the business area, experimenting with system under development.

4.1.1 Asking the Users

This usually includes interviewing and questionnaires. Interviewing the users requires considerable skill and preparation. The interviewing process must be planned, since managers at several levels may have to be questioned. During the interview, the analyst must convey a clear understanding of the purpose of the interview, ask specific questions in terms understandable to the interviewee listen rather than anticipate answers, control the interview but be open to a suddenly discovered rich source of information, and create a record of what is learned. The analyst should analyze the results immediately following an interview session. Interviews are a very rich but costly and time-consuming communication channel. Questionnaires are an efficient way of asking many users at once, particularly users who are dispersed in the field. Questionnaires are distributed on diskettes or intranets.

4.1.2 Deriving From an Existing System

The requirements for a proposed system may be derived from an existing information system. The possibilities are:

- The system that will be replaced by the proposed system.
- A system similar to the proposed installed elsewhere and is accessible to the analyst.
- A proprietary package whose functionality may be analyzed.

The requirements for the proposed system are derived from the data contained in the outputs of the existing system and inputs to it. The data can also be obtained from the existing programs and systems documentation such as procedures, manuals, organization charts, file descriptions, operations manuals, and so forth.

Existing data replication schemes aims at either reducing the query delay or improving the data availability, but not both. Hence, in this project new data replication scheme is used to balance the trade-offs between query delay and data availability.

4.2 Applying Tools for the Structured Analysis

The purpose of systems analysis is to devise a logical model of the proposed system. Using the methodology known as structured systems analysis, we graphically describe the system as interacting processes that transform input data into output data. These processes may then become code modules as the system is further designed and programmed.

4.3 Feasibility Study

The main objective of the feasibility study, the introductory phase of development, is to establish whether the proposed system is feasible or, to be more accurate, desirable, before resources are committed to the full-scale project. In a feasibility study, systems analysts perform a preliminary investigation of the business problem or opportunity represented by the proposed system development project. Specifically, they undertake the following tasks:

- Define the problem or the opportunity which the system will address.
- Establish the overall objectives of the new system.
- Identify the users of the system.
- Establish the scope of the system.
- Propose general hardware/systems software options for the new system.

- Perform a make-or-buy analysis for the application.
- Perform a value assessment, such as the cost-benefit analysis, based in part on the estimate of the development project size.
- Assess the project risk.
- Recommend whether to proceed with the project, or whether to abandon the project.

This project is feasible, as there will not be very much difficulty in getting the required resources for the development and maintaining the system. All the resources needed for the development of the software as well as the maintenance of the same is available in the organization, in this project resources are utilized which are already available.

4.4 Requirements Analysis

The principal objective of requirements analysis is to produce the requirements specifications for the system, which set out in detail what the system will do. Requirements (also known as functional) specifications establish an understanding between the system developers, its future users, the management and other stakeholders.

Requirements analysis needs to establish:

- What outputs the system will produce, what inputs will be needed, what processing steps will be necessary to perform inputs into outputs, and what data stores will have to be maintained by the system.
- What volumes of data will be handled, what numbers of users in various categories will be supported and with what levels of service, what file and database capacities the system will need to maintain, and other quantitative estimates of this type.
- What interface will be provided for the users to interact with the system, based on the skills and computer proficiency of the intended users.
- The control measures that will be undertaken in the system.

All required analysis for the LanLink software development has been done as per the above procedure.

CHAPTER 5

SYSTEM DESIGN AND IMPLEMENTATION

5.1 USE CASE DIAGRAM

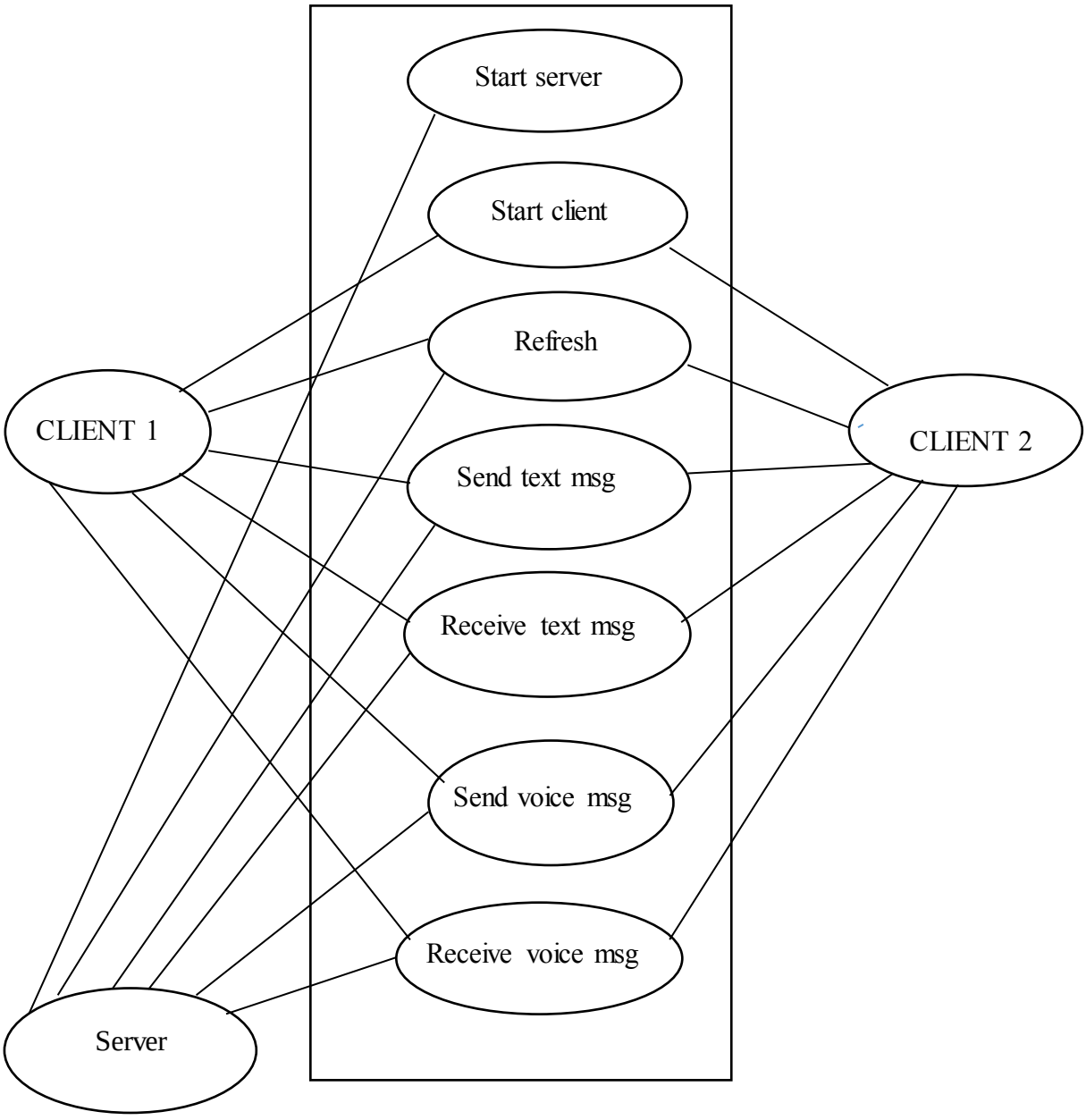


Figure 5.1: Use case diagram

Use case diagram gives a pictorial brief of the design and functions of the software. Here client and server are the primary actors developed using the server centric architecture. Client is capable of initializing six actions. One of the clients starts the software and configures the server information and the message of either text, voice is sent to the server. Server which maintains the client's information and it reroutes the messages to the specific destination.

USER SCENARIO

Use case: Sharing messages either in text, voice format

Primary actor: Client, Server

Goal in context: Sending and receiving messages between connected clients via server.

Precondition:

1.System must be fully configured with basic processing power and network configuration hardware.

2.Client software must be running the user desktop.

Trigger:when a message has to be sent to the counterpart,a family,a friend etc. then trigger the client software.

Use Case Scenario:

1. client runs the software .
2.client may go to the configuration menu in main window to set up the server configuration.
3.client may press about menu to know about the software.
4.client sees available friends in the main window and pressing on any of the button takes him to the next frame i.e chat window.
5.client types message in the chat box and press either send,send voice depending on the data format.
6.client can open multiple chat windows belongd to many other clients.
7.client may press Emergency button in the chat window to alert the server about the problem.

Exceptions:

1.user machine has outdated network configuration or non compatible operating platform.

2.Invalid configuration of the server IP.

3.Filling the chat box with no letters though pressing the send button.

Priority: Operating network configuration is of high priority ,implemeted in association with the basic functions.Server is also a high priority design capable of handling client events time effectively.

Channel to actor: Via pc having NIC's with tcp socket streams.

Secondary actors: There are no secondary actors.

Frequency of use:can be used infrequent having available connections atleast more than one.

Open issues:

1.How many clients can communicate in the network.

2.What measures are taken if the server fails.

3.How do you alert the cleint about server failure.

4.How do you manage the privacy of the user profile containing history of messages.

5.What measures are taken if many clients initialises communication with the same client at same time.

5.2 ACTIVITY DIAGRAM

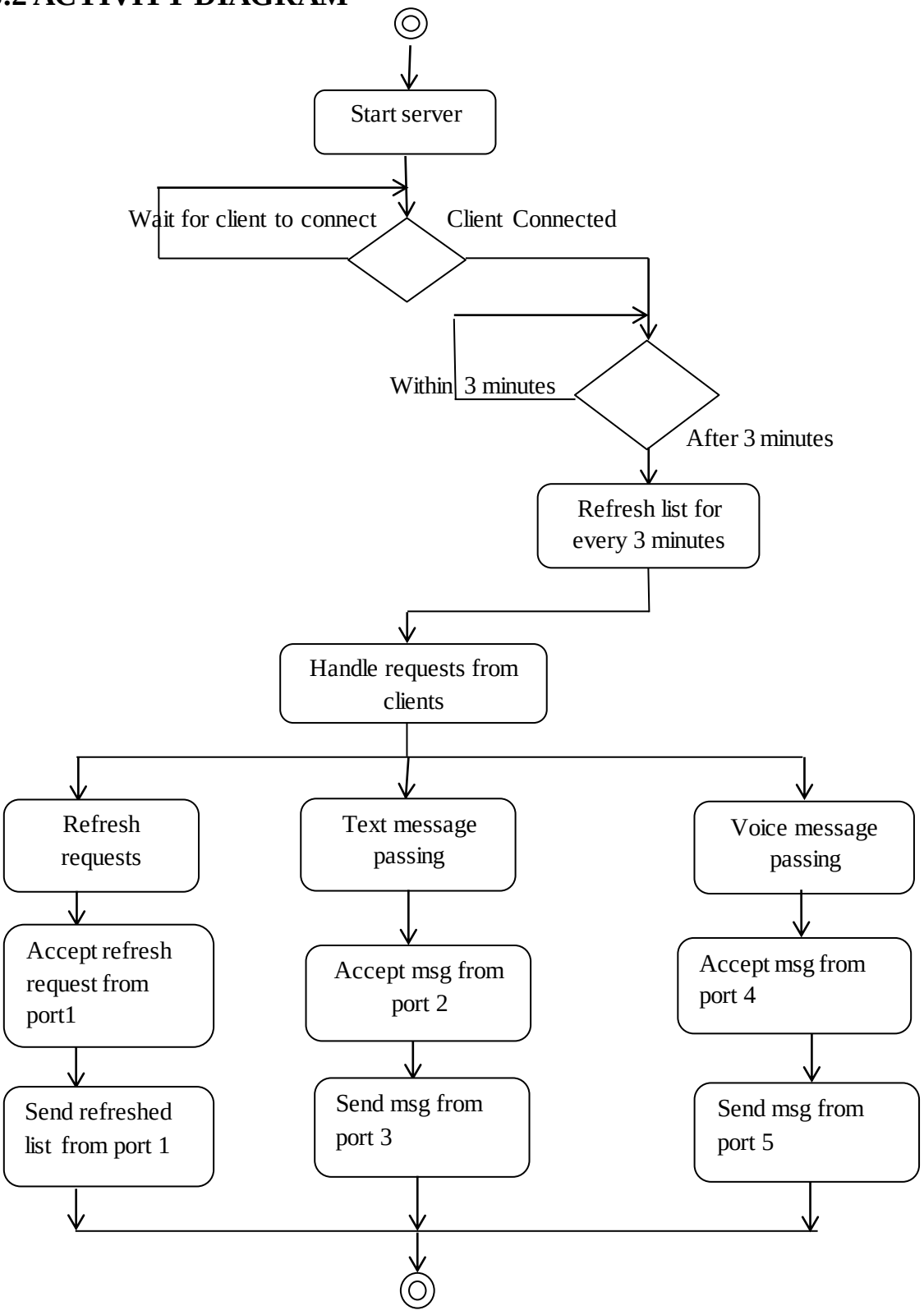


Figure 5.2.1: Activity Diagram for server

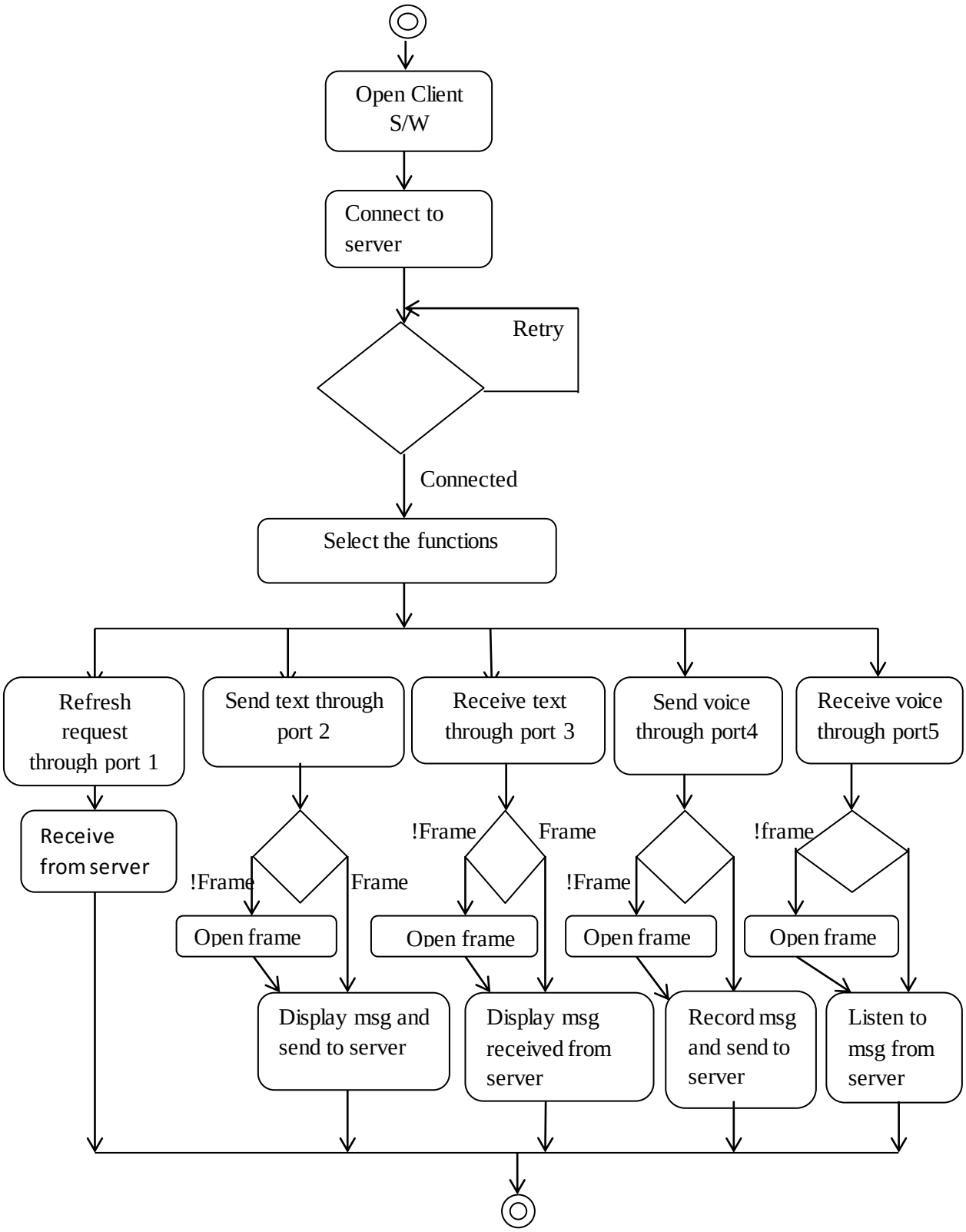


Figure 5.2.2: Activity diagram for client

5.3 ARCHITECTURE DIAGRAM

DATA CENTRED ARCHITECTURE

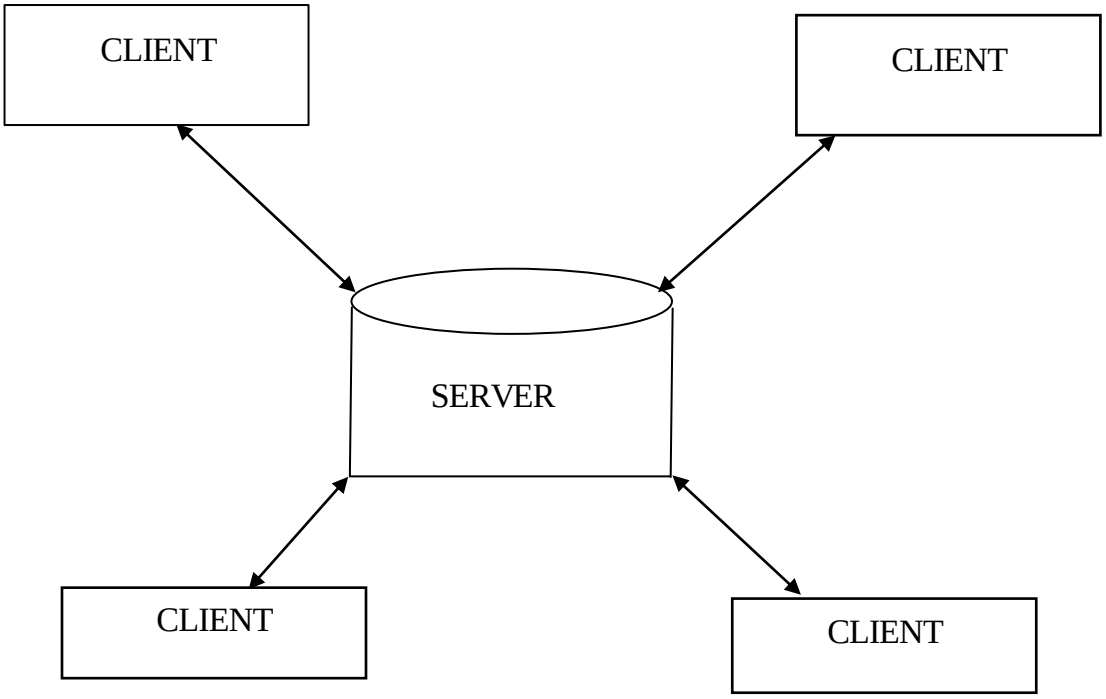


Figure 5.3: Architectural diagram

Data centered architecture is one in which a data store resides in the center which is accessed by other components in the system. Here the server acts the part of the data center and the client applications are the ones which access it. This promotes independent performance of clients as they run independently and are not dependent on other clients for existence. This architecture promotes inerrability which means that existing components can be changed and new clients can be added to the architecture without any concern about the architecture. In short, it aids in scalability without altering the architecture.

5.4 DATA FLOW DIAGRAM

5.4.1 CONTEXT-LEVEL DIAGRAM

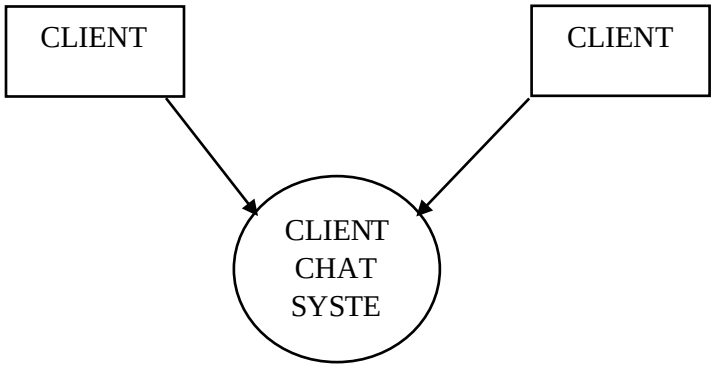


Figure 5.4.1: Data-flow diagram level 0

5.4.2 DFD LEVEL 1

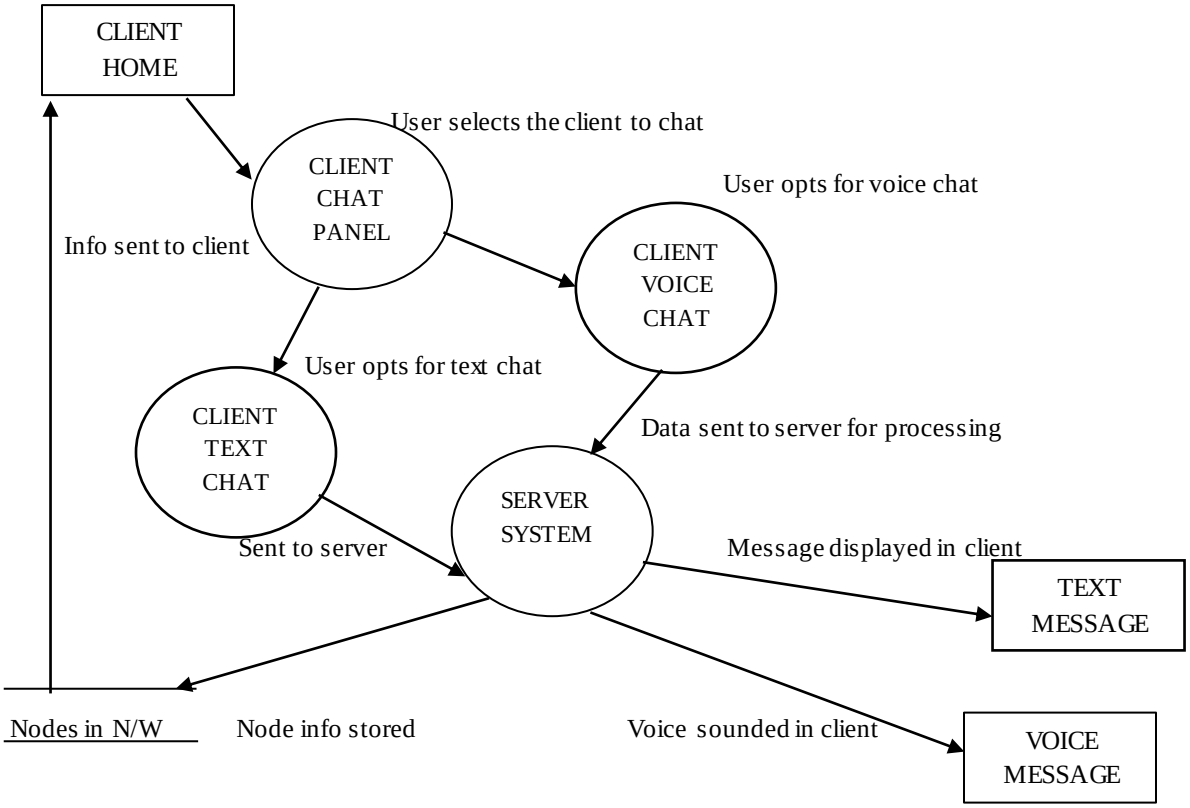


Figure 5.4.2: Data-Flow diagram level 1

CHAPTER 6

METHODOLOGY

6.1 SERVER SIDE MODULES

Port scanning

- Once the server is started, it collects the list of IP address from the "arp.txt".
- This .txt file is created by running command prompt with the command "arp -a". With this, IP addresses of systems connected from local ARP cache can be obtained.
- Server then picks up the IP s with those starts with "192"
- Result=socket.connect_ex(("ip_address",PORT_NO))
- Name=socket.gethostbyname(ip_address)

Server refresh

- This is requested by the client and during this process, the server sends the list which it built during the PORT SCANNING process to the requesting client.
- Socket.sendall(name_list_as_string)

Text and voice routing

- For text messaging, client connects to port number 20002 of the server and then sends the text data. All the communication uses the services of TCP. Server on receipt of text data spawns with the name of sender to forward it to the other client by connecting to port 20001.
- In voice messaging, Server first receives the file in .wav format through socket with port number 20008. Prior to file receiving it accepts the name of sender and receiver. Connects to receiver at port 20007.

6.2 CLIENT SIDE MODULES

Configuring server

- This is used to setup an initial connection between the client and the server. This sets up the server address in the client. In a way it also acts as sort of an authentication in the sense that only those who know the proper server address can login to that chat setup.

Dummy server

- This a server setup in the client which enables the server to identify if a client is active in the setup and ready to be involved in communications with the application.

Refresh request

- It involves requesting the server for data regarding the computers that are active in the network running application.

Text and voice routing

- Text sending is done by setting up a connection stream between the client and the server.
- Then the message written by the client in the text field is extracted and sent to the server.
- This transaction uses port number 20001.
- On receiving a socket connection from the server, the resources are allocated for the connection.
- The format of data received is one where the message is suffixed with the name of the sender. Port number used for this purpose is 20002.

Voice sending and receiving

- Voice sending involves the usage of PyAudio modules which records voice from the default mic in the client's computer and stores it in the .wav format.

- This file is then sent to the server through the port number 20007.Timeof recording is set by the developer.
- Voice receiving is done through port number 20008. This is done using an active socket in the client which is handled by a select server.
- It involves receiving the name of the sender first followed by the data of the .wav file recorded in the sender's computer.
- The data thus saved is then opened using a stream and played through the default speakers in the client's computer

6.3 PROGRAMMING CONCEPTS USED

- **Socket creation:** A socket is a protocol independent method of creating a connection between processes. Sockets are characterized by their domain, type and transport protocol.
- socket=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
- **Threading:** The advantage of threading is the ability to create applications that uses more than one thread of execution. . In Python, threading module supports starting new threads, synchronizing multiple threads and aborting them.
- thread=threading.Thread(name='name',target=functionname, args = (,))
- **Select Servers:** UNIX standard select () function. This function, defined in select module, is used on multiplexing I/O requests among multiple sockets or file descriptors.

6.4 GRAPHICAL USER INTERFACE (GUI)

- Frame: The wxPython Frame widget has minimize, maximize and close by default and contents.
- Button: Binds the action to the button press. Transfers control to other part of the program.
- Panel : A panel is a space on which controls are placed.
- TextCtrl: text area where the user types the message.
- Dialog box: Dialog box comes in different varieties, a yes/no dialog box , MessageDialog box.
- Gauge: A gauge is a horizontal or vertical bar which shows a quantity. It can be used as a progress bar .

CHAPTER 7

TESTING

7.1 INTRODUCTION

7.1.1 Design of test case

Testing is the process of evaluating a system or its component(s) with the intent to find whether it satisfies the specified requirements or not. In simple words, testing is executing a system in order to identify any gaps, errors, or missing requirements in contrary to the actual requirements. According to ANSI/IEEE 1059 standard, Testing can be defined as - A process of analyzing a software item to detect the differences between existing and required conditions and to evaluate the features of the software item.

A primary purpose of testing is to detect software failures so that defects may be discovered and corrected. Testing cannot establish that a product functions properly under all conditions but can only establish that it does not function properly under specific conditions. The scope of software testing often includes examination of code as well as execution of that code in various environments and conditions as well as examining the aspects of code: does it do what it is supposed to do and do what it needs to do. In the current culture of software development, a testing organization may be separate from the development team. There are various roles for testing team members. Information derived from software testing may be used to correct the process by which software is developed.

7.1.2 Test case

A **test case**, is a set of test inputs, execution conditions, and expected results developed for a particular objective, such as to exercise a particular program path or to verify compliance with a specific requirement. A test case could simply be a question that you ask of the program.

7.1.3 Testing methodologies

System is carried out in three stages:

- Unit testing
- Integration testing
- System testing

Unit testing

A unit is the smallest testable part of software. It usually has one or a few inputs and usually a single output. In procedural programming a unit may be an individual program, function, procedure, etc. In object-oriented programming, the smallest unit is a method, which may belong to a base/ super class, abstract class or derived/ child class.

The major units identified in this project are

- Client configuration by connecting to server using IP address
- Sending one message
- Sending broadcast message

Integration testing

Integration testing is the phase in software testing in which individual software modules are combined and tested as a group. It occurs after unit testing and before validation testing. Integration testing takes as its input modules that have been unit tested, groups them in larger aggregates, applies tests defined in an integration test plan to those aggregates, and delivers as its output the integrated system ready for system testing.

System testing

System testing of software or hardware is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. System testing falls within the scope of black-box testing, and as such, should require no knowledge of the inner design of the code or logic

7.2 Sample test cases

TC#	Test cases	Expected output	Actual output	Result status
TC01	Verification to run the server program	Program should run	Application is executing properly without any interrupts.	PASS
TC02	Verification to execute/run without any interrupts	Application should run	Application is executing properly without any interrupts	PASS
TC03	Configuring Client .	Client should connect to server	Client gets connected to server.	PASS

TC#	Test cases	Expected output	Actual output	Result status
TC04	Refreshing the connected client list.	Client should receive the updated list from the server.	List is received from the server.	PASS
TC05	Opening the chat frame.	Chat frame with all the control structure should be built.	Chat frame is loaded without any GUI errors.	PASS
TC06	Sending text message.	User text to be created and status bar to show success message.	Displays success message on status bar without errors.	PASS

TC#	Test cases	Expected output	Actual output	Result status
TC07	Sending voice message.	Voice file to be created and send it to server.	Message is successfully received at the server.	PASS
TC08	Receiving the text message.	Message to be appended to frame of particular sender and if the frame is not open, create a frame and append.	Message is appended without errors.	PASS
TC09	Receiving voice message.	Receiving voice with popup to play or not.	Specified action is performed without errors.	PASS

TC#	Test cases	Expected output	Actual output	Result status
TC10	Routing text message.	Receiving text and routing the message to the sender.	Message is received at the sender.	PASS
TC11	Routing voice message.	Receiving voice and routing the voice to the sender.	Voice is received at the sender.	PASS
TC12	Refreshing the connected client list.	List is updated and sent to the requesting client.	No error in updating list and sending to client.	PASS

CHAPTER 8

SCREENSHOTS AND RESULT DISCUSSION

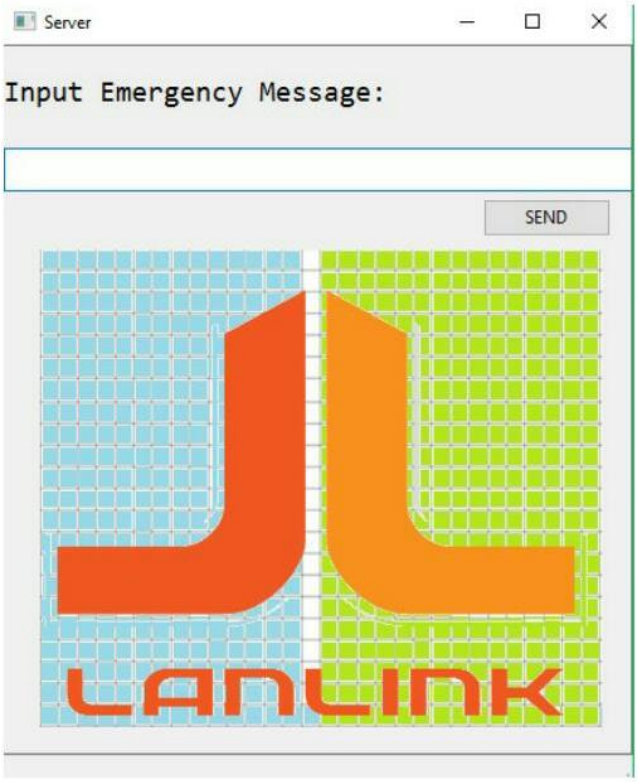


Figure 8.1: Server initial screen

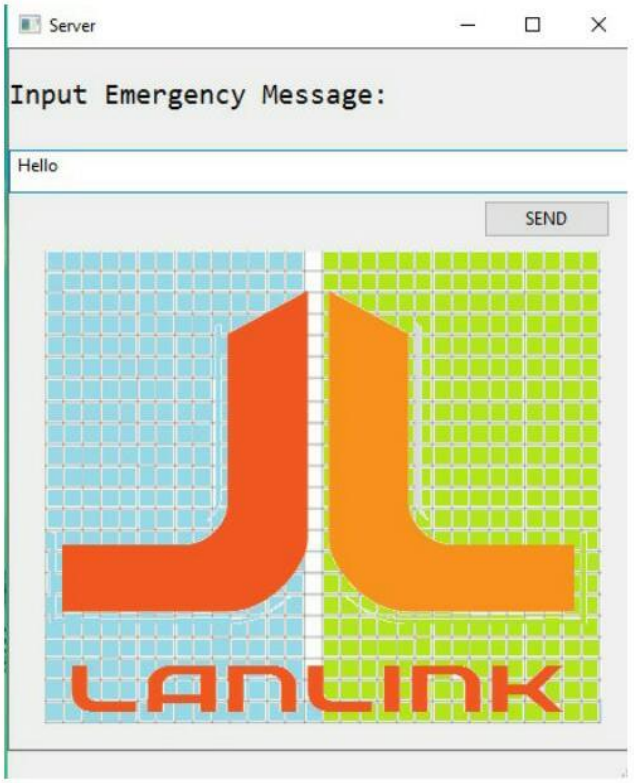


Figure 8.2: Server sending broadcast message

- When server program is executed, the logo of the application and a frame to enter the braodacast message is displayed as shown in the figure 8.1
- Figure 8.2 shows how input is given in the text box of the frame

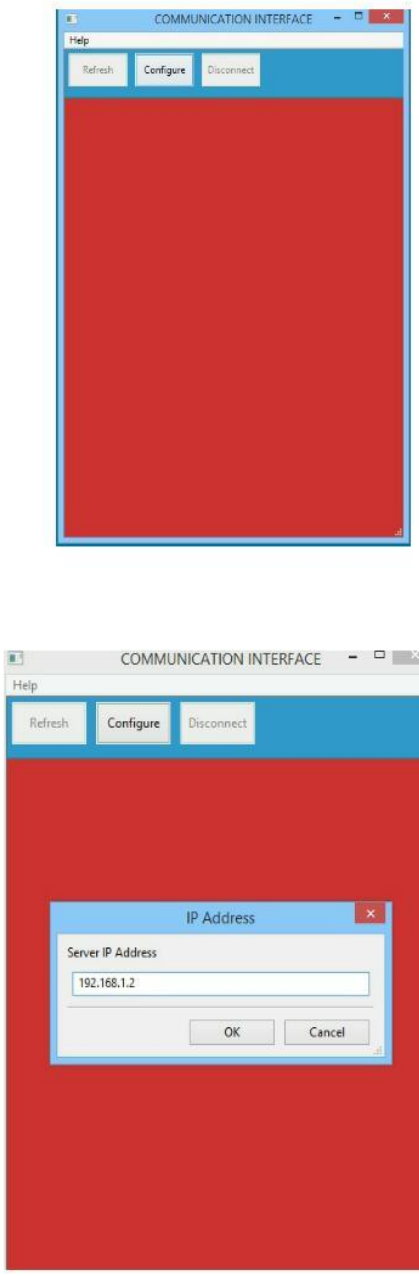


Figure 8.3: Client connecting to server

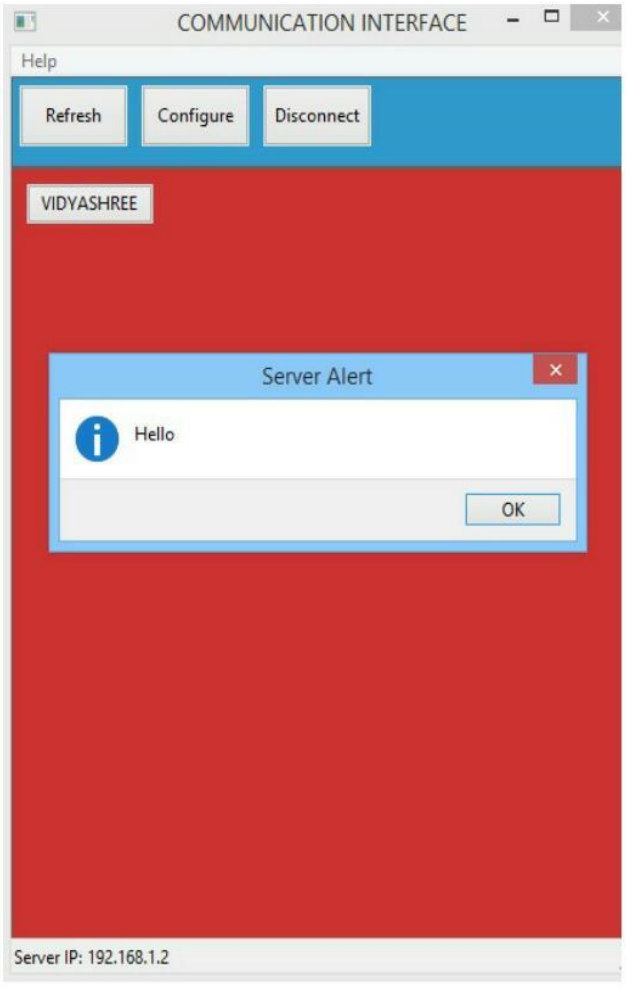


Figure 8.4: Client receiving broadcast message

- Figure 8.3 shows how the client can enter IP address of the server by clicking configure button.
- After configuring with server, client receives a broadcast message as shown in the figure 8.4

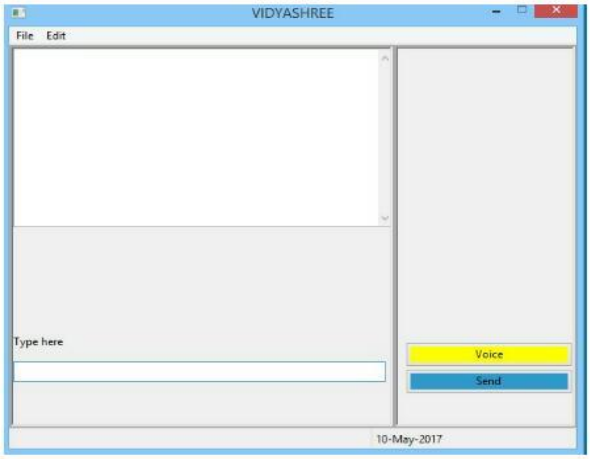


Figure 8.5: Chat window

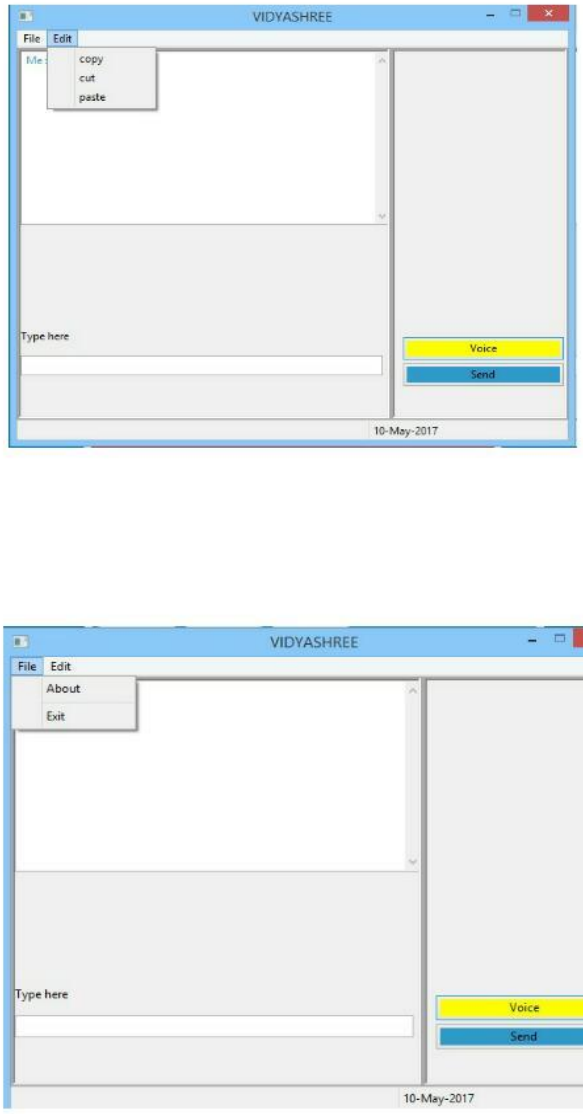


Figure 8.6: File and edit options

- When client clicks any of the other clients in the network, a chat window as shown in the figure 8.5 is displayed.
- The window has two menu options "file" and "edit" as shown in the figure 8.6

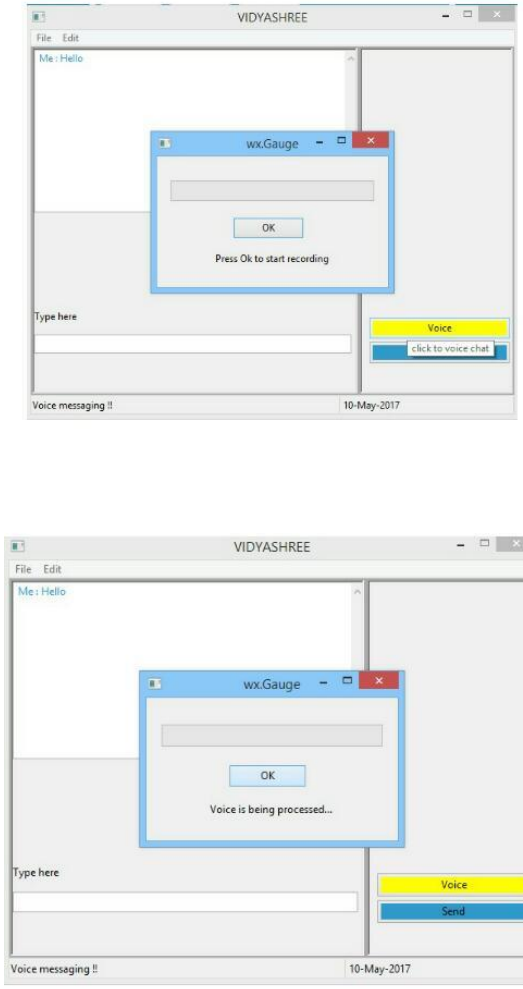


Figure 8.7: Sending voice message

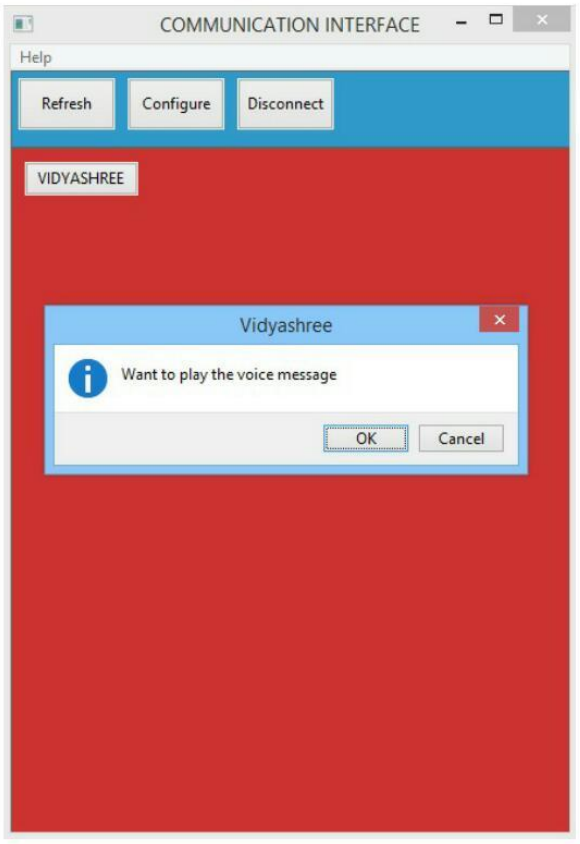


Figure 8.8: Action to be performed on receiving a voice message

- A client can send voice message to another user using "voice" option in the chat window.
- The voice will be processed by pyAudio and sent to the receiver as shown in the figure 8.7
- The receiver gets an alert message to play the voice message as shown in the figure 8.8

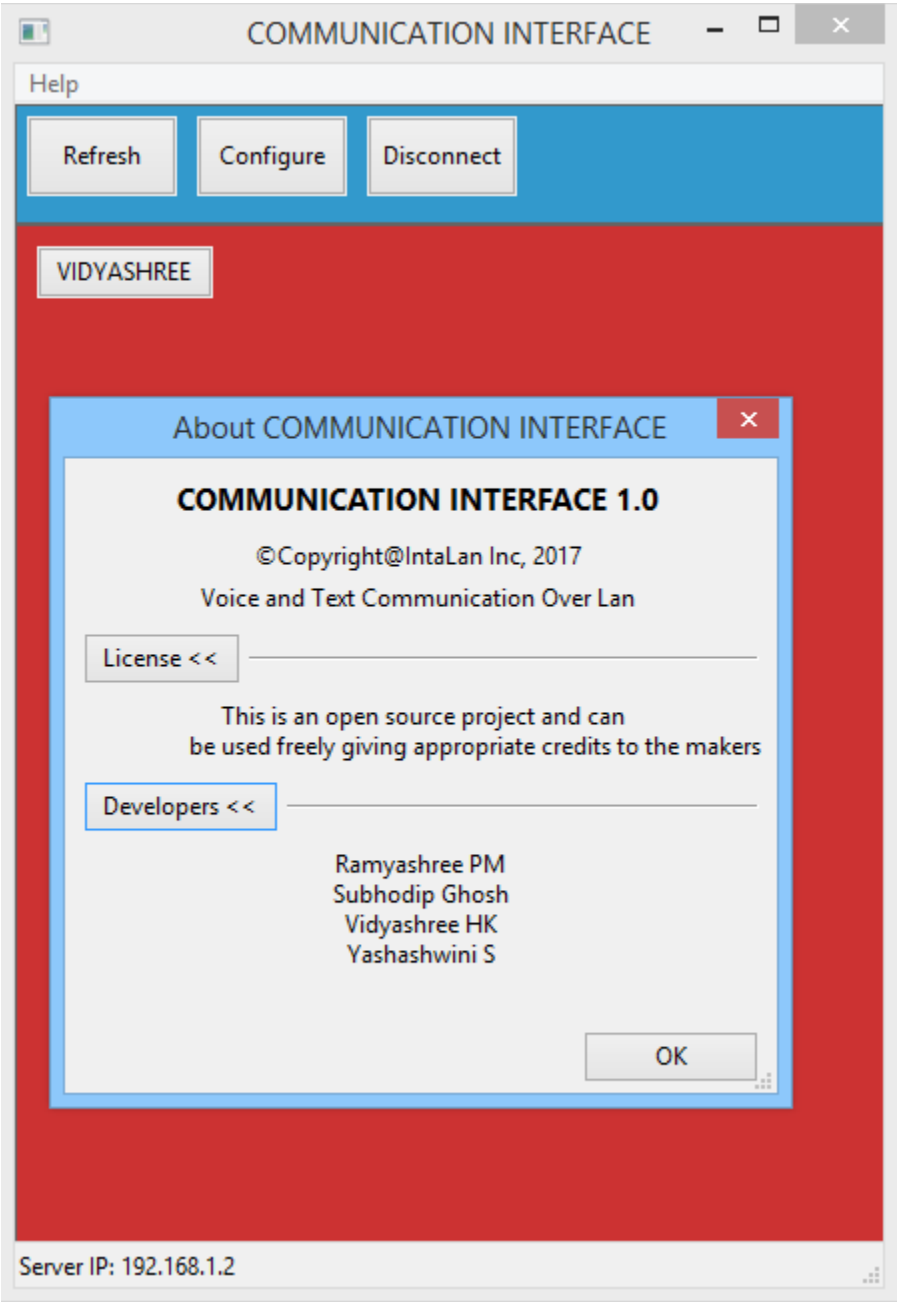


Figure 8.9 : About the application

- The information about the application, license and developers can be obtained by selecting help option in the menu as shown in the figure 8.9

CHAPTER 9

REFERENCES

Websites

- www.stackoverflow.com
- www.quora.com
- www.wiki.wxpython.com
- www.python.org
- www.ieee.explorer.org

Books

- Data Communications and Networking by Behrouz A. Forouzan
- Computer Networks: A System Approach Larry L. Peterson and Bruce S. Davie
- Data and Computer Communications by William Stallings
- Computer Networks: International Edition
- Computer Networking: A Top - Down Approach
- A Brain-Friendly Guide eBook: Al Anderson, Ryan Benedetti
- Data Communications and Networking
- Hello World! Computer Programming for Kids and Other Beginners(http://helloworldbookblog.com), 2nd ed., Warren Sande and Carter Sande, Manning
- Invent your Own Computer Games with Python(http://inventwithpython.com), 2nd edition, Al Sweigart
- Python for Software Design: How to Think Like a Computer Scientist, Allen B. Downey, Jeff Elkner and Chris Meyers, Green Tea Press
- Python Programming for the Absolute Beginner(http://www.cengage.com/search/pr...), 3rd ed., Michael Dawson, Course Technology

APPENDIX

Python

Python is a widely used high-level programming language for general-purpose programming, created by Guido van Rossum and first released in 1991. An interpreted language, Python has a design philosophy which emphasizes code readability (notably using whitespace indentation to delimit code blocks rather than curly braces or keywords), and a syntax which allows programmers to express concepts in fewer lines of code than possible in languages such as C++ or Java. The language provides constructs intended to enable writing clear programs on both a small and large scale. Python features a dynamic type system and automatic memory management and supports multiple programming paradigms, including object-oriented, imperative, functional programming, and procedural styles. It has a large and comprehensive standard library.

Python interpreters are available for many operating systems, allowing Python code to run on a wide variety of systems. CPython, the reference implementation of Python, is open source software and has a community-based development model, as do nearly all of its variant implementations. CPython is managed by the non-profit Python Software Foundation.

WxPython

wxPython is a wrapper for the cross-platform GUI API (often referred to as a "toolkit") wxWidgets (which is written in C++) for the Python programming language. It is one of the alternatives to Tkinter, which is bundled with Python. It is implemented as a Python extension module (native code). Other popular alternatives are PyGTK, its successor PyGObject and PyQt. Like wxWidgets, wxPython is free software. wxPython was created when Robin Dunn needed a GUI to be deployed on HP-UX systems and also on Windows 3.1 within a few weeks. While evaluating commercial solutions, he ran across Python bindings for the wxWidgets toolkit. Thus, he learned Python and, in a short time, together with Harri Pasanen, became one of the main developers of wxPython, which grew from those initial bindings.

The first versions of the wrapper were created by hand. However, soon the code base became very difficult to maintain and keep synchronized with wxWidgets releases. Later versions were created with SWIG, greatly decreasing the amount of work to update the wrapper. The first "modern" version was announced in 1998.

Applications developed with wxPython

- BitTorrent, a peer-to-peer BitTorrent application
- Chandler, a Personal Information Manager
- Editra, a multi-platform text editor
- Google Drive, desktop client for the Google cloud-based storage system[6]
- GRASS GIS, a free, open source geographical information system
- Métamorphose, a batch renamer
- Phatch, a Photo Batch Processor
- PlayOnLinux and PlayOnMac, Wine front-ends

PyAudio

PyAudio provides Python bindings for PortAudio, the cross-platform audio I/O library. With PyAudio, you can easily use Python to play and record audio on a variety of platforms. PyAudio is inspired by:

- pyPortAudio/fastaudio: Python bindings for PortAudio v18 API.
- tkSnack: cross-platform sound toolkit for Tcl/Tk and Python.