A

Major Project

Submitted

In partial fulfillment

For the award of the Degree of

Bachelor in Technology

In Department of Computer Science &Engineering

Supervisor:

Mr. Sandeep Bhargava

Asst. Professor

SGVU, Jaipur

Submitted By:

Parikshit Agarwal

SGVU101014968

B Tech CS (6th Sem)

Department of *Computer Science &*Engineering

Suresh GyanVihar University

Mahal, Jagatpura, Jaipur

April 2013

ABSTRACT

‘Wavr’ project is started in the aim of developing a cross platform instant chat messaging application for the local LAN network based upon the idea that a user can able to chat irrespective of the underlying architecture both in terms of hardware and software in his/her network.

The main objectives of this project is to develop a fully chat application that works as an expert system. This system provides an easy-to-use, faster, efficient and robust platform for the users. The Other objective of this project is to provide users a cheap, reliable, and convenient way to communicate with each other using computer systems.

Candidate's Declaration

I hereby declare that the work, which is being presented in the Major Project, entitled "Wavr" a LAN Chatting application in partial fulfillment for the award of the degree of "Bachelor of Technology" in Computer Sciene and submitted to the Department of **Computer Science and Engineering, Suresh Gyan Vihar University** is a record of my own investigations carried under the Guidance of Mr. Sandeep Bhargava, Department of Computer Science and Engineeering.

I have not submitted the matter presented in this Project any where for the award of any other Degree.

(Name and Signature of Candidate)

Parikshit Agarwal

Enrolment No: SGVU101014968

Counter Singed by

Name of Supervisor

Mr. Sandeep Bhargava

Assistant Professor, CSE Dept

Suresh Gyan Vihar University

Table of contents

1. Abstract……1

2. Candidates Declaration……2
3. Introduction……..5
4. Definitions……6
    a. Overlay Networks……..6
    b. P2P Networking…….....6
5. Description……...7
6. Manual……..9
    a. Build……...9
        i. Linux
        ii. Mac OS X
        iii. Microsoft Windows
    b. Installation……10
        i. Linux
        ii. Mac OS X
        iii. Microsoft Windows
7. Methodology and planning of work……11
    a. GUI (Graphical User Interface) ……13
        i. Main Window
        ii. Chat Window
        iii. Broadcast Window
        iv. Chat Room Window
        v. Transfer Window
    b. Application Core……14
    c. Distributed Message Processing Layer……..14
        i. Message Processor
        ii. Network
    d. Network Layer……14
        i. TCP Network
        ii. UDP Network
    e. Channels……14
        i. Message Channel
        ii. File Channel
8. Interface……15
9. Program Structure……..15
    a. Classes……16

i. Network……16
ii. TCP Network……..16
iii. UDP Network……17
iv. Shared……17
v. XML message……18
vi. Channel Messaging……18
vii. Datagram……18
10. Screenshots……19
a. Main Window……..19
b. Chat Window……20
c. Public Chat Window……21
d. History Window……..22
e. Settings Window……23
f. BroadCast Window……24
11. Testing……..25
a. Unit Testing
b. Integration Testing
c. High Order Testing
d. Beta Testing
12. Conclusion……27
13. Bibliography……28

Introduction

‘Wavr’ project is started in the aim of developing a cross platform instant chat messaging application for the local LAN network based upon the idea that a user can able to chat irrespective of the underlying architecture both in terms of hardware and software in his/her network.

The main objectives of this project is to develop a fully chat application that works as an expert system. This system provides an easy-to-use, faster, efficient and robust platform for the users. The Other objective of this project is to provide users a cheap, reliable, and convenient way to communicate with each other using computer systems.

This project is divided into 2 parts:

First being the library which handles all the background processes of the chat engine that is sending the messages to the peer and notify the user in case of new message arrives.

Second is the GUI (Graphical User Interface) part which is the front part of the application containing all the windows from which chat can be possible.

Definitions

Overlay Networks:

- Network
  - defines addressing, routing, and service model for communication between hosts
- Overlay network
  - A network built on top of one or more existing networks
  - adds an additional layer of indirection/virtualization
  - changes properties in one or more areas of underlying network

P2P Networking

- Each participant (user of the chat application in this case) is a "peer" and plays an identical role in the system
- Users co-operatively build the service without relying on a third party to run a server
  - Instead of the "control" being centralized at a server, it is distributed amongst all the users
  - The users must trust each other at some level to be able to do this
- Examples: Gnutella, FastTrack, Chord, CAN

Description

The goal that we wanted to achieve in the making of this program was to develop and application that can run on all systems irrespective of the underlying architecture and operating system.

Wavr project currently can operate on

- Windows
- Linux
- Mac

Android and Apple version of this project is under development.

The features of this project are:

● Five different (intuitively just touch) of difficulty.

● Very eficient.

● Threading, so that the UI during the short time remains responsive

● Past History: User can check its past messages.

● File and folder transfer is possible with drag and drop feature.

● Ability to customize our own avatar.

● Save and read a past messages.

● Group messaging is possible

● Messages can be broadcasted to all users.

● Ability to handle multiple users at once.

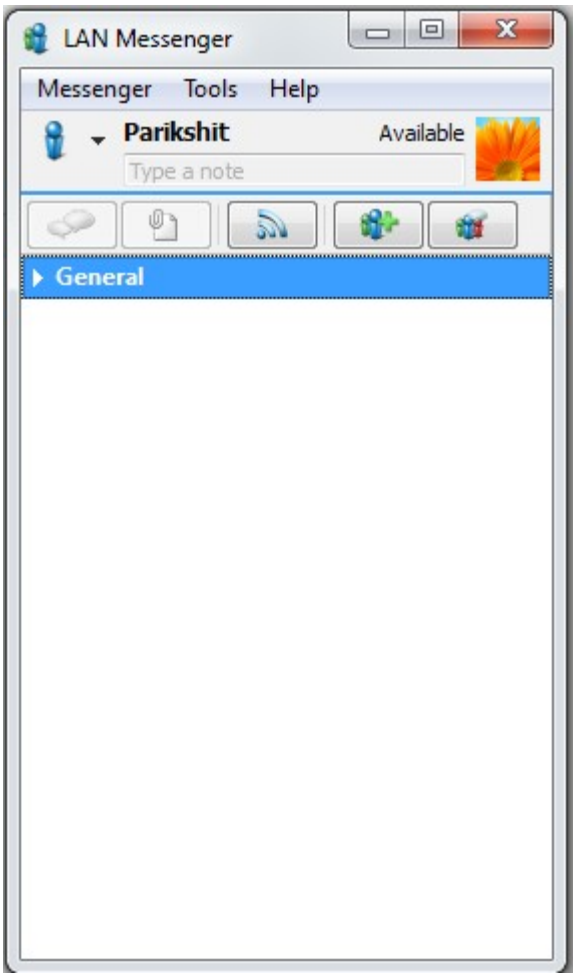
● Run the application on startup.

● Attach the files to the peer and send them to multiple users at once.

● Refreshing the contacts list of the peers.

● Notifications on status bar of active presence.

●Doxygen documentation in all classes.


LAN Messenger
Messenger
Tools
Help
Parikshit
Available
Type a note
General

Manual

Build

Linux

There is a build script included for Linux: build_linux.sh. if
you run this script, is Lan messgenger in release mode sifted, and a
tar.gz archive. The only requirement is that the Qt development libraries
are installed.

Mac OS X

Also for Mac OS X is a build script included: build_mac.sh.
When you run this script, Lan messenger sifted in release mode as a
Universal Binary1 (ie for both the x86 and PPC CPU architectures),
theQt frameworks (as dylibs, dynamic libraries) in the app package
copied, and finally a dmg disk image is created (
used format for application distribution on Mac OS X).

Microsoft Windows

Finally, it is also a build script for Windows: build_windows.bat.
This script (basically a batch?) compiles in release mode (there is
using the command-line version of Visual Studio) and then
all necessary dll's are copied to the same folder. There is no automatic
an archive or installer created because it is not available by default
on Windows. The problem with this version is that the Qt process after
of the program afsuiten remains running (this is a bug in Qt 4.3.2). as
we have also written a script for the MinGWQt 4.4

compiler (build_windows_mingw.bat), in which this problem is solved.

2.2 Installation

2.2.1 Linux

The unusual dynamic libraries in Linux is to join forces with the program (unlike Mac OS X and Windows), but is frequently use of package managers. The package manager used be to install the Qt libraries. On Ubuntu, the APT 2 package manager there, you must run this command: sudo apt-get install libqt4-core. This is only needed if Qt is not installed. at all Linux distributions that use KDE 3 desktop, Sudoku out-of-the-box work.

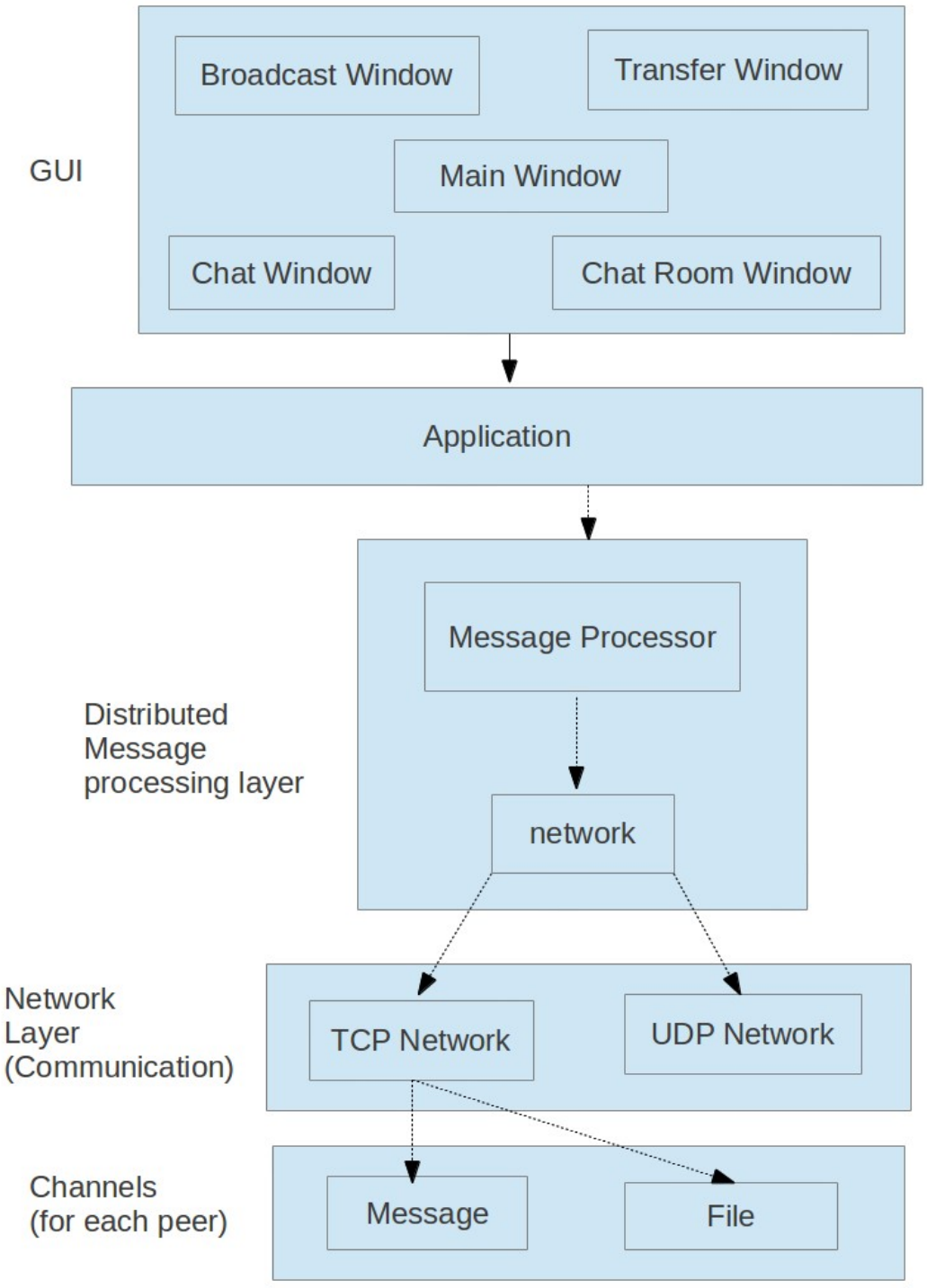
2.2.2 Mac OS X

The installation on Mac OS X works just like any other Mac application: drag the program to the / Applications directory and you're done.

2.2.3 Microsoft Windows

Thanks to the build script all dll's are all included. You can Sudoku.exe So anywhere, as long as you have the dlls but meekopieert.

Methodology and planning of work



GUI (Graphical user interface)

This layer deals with the GUI part of the application. It contains the

- Main Window
- Chat Window
- Broadcast Window
- ChatRoom Window
- Transfer Window

Main Window is the main screen of the application containing all the peers name available for chatting and we can set our presence, avatar and name here

Chat Window is specific to the peer to which we are chatting, and contains all the messages send and received from that user. The incoming messages from the peer will appear on this screen.

Broadcast Window is the screen from which we can send broadcast messages to all the active peers and the incoming broadcast messages can be seen on this screen.

ChatRoom Window is the screen containing multiple peers of which we can send and receive message from simultaneously.Only the users present in the chatroom can be able to receive and send the messages.

Transfer Window is used to transfer the files and folders to the peers to which we want to send the objects.

Application Core

This layer contains the application core contaning the functionality used to attach the GUI part of the application to the libraries of the application.

Distributed message processing layer

This layer deals with the message processing things, for eg the message the user wants to send is first converted to plain text, ie all the smileys given by the user is converted to its equivalent code. Then this message is wrapped around the threadid which is the unique code given to that message, after that the color used by the user is specified around the wrapped message, after that the font used.

Then the message is wrapped around the message type ie groupmessage or file etc, then around message id (unique id given to the message), then peerid (id of the peer to which message is to be sent, then comes the local id (id of the sender)

For security, The message generated so far is cryted by the openssl algorithms.

This cryted message is then sent to the destinee specifying the datagram type.

Network Layer

This layer deals with the type of communication we want to establish with the peer, ie either TCP or UDP

All the communication to the peers ie messaging is done via TCP protocol except the handshaking in connection establishment and broadcast messages which is done via UDP protocol.

Channels

This layer contains the channel classes which specify the type of channel on which communication is done. Ie if we want to send a simple text message then Text Channel is used and if we want to send an object ie file or folder then file channel is used.

Interface

Wavr application is build with aim of having a simple UI (user interface). All the GUI functionality can be customized by the user easily.

After launching the application, the first main screen will contain the peer names.

Now if a user wants to chat with a specific peer, all he/she needs to do is just double click on that peer, a chat window will appear with his/her name on it. Now he/she can type the message in the text box and then click on the send button.

The same follows for the broadcast messages and file/folder transfer.

Program Structure

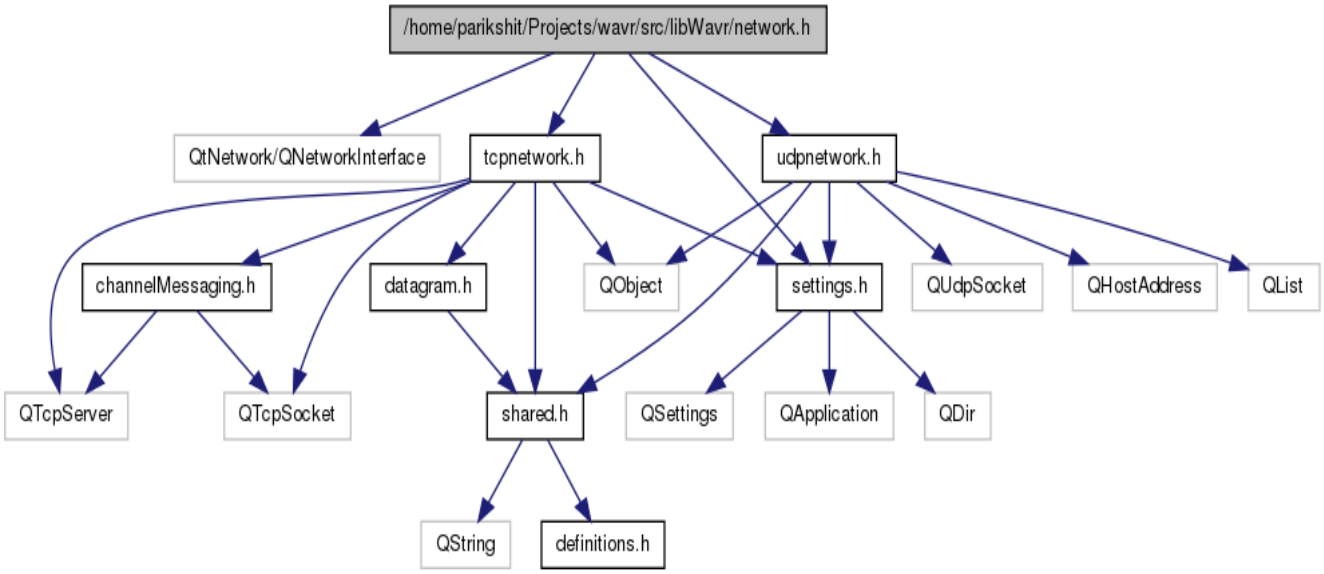
In this section we briefly discuss the data structures and algorithms.

However, it is convenient to keep the Doxygen documentation at hand

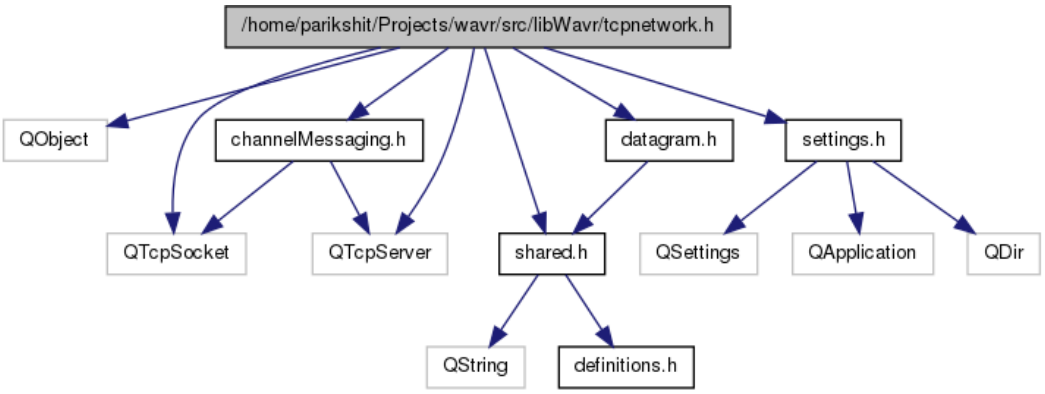
since all of our classes Doxygen documentation.

Classes

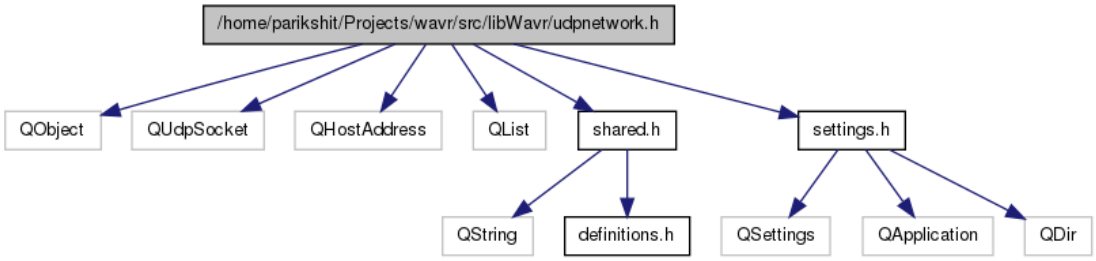
Network



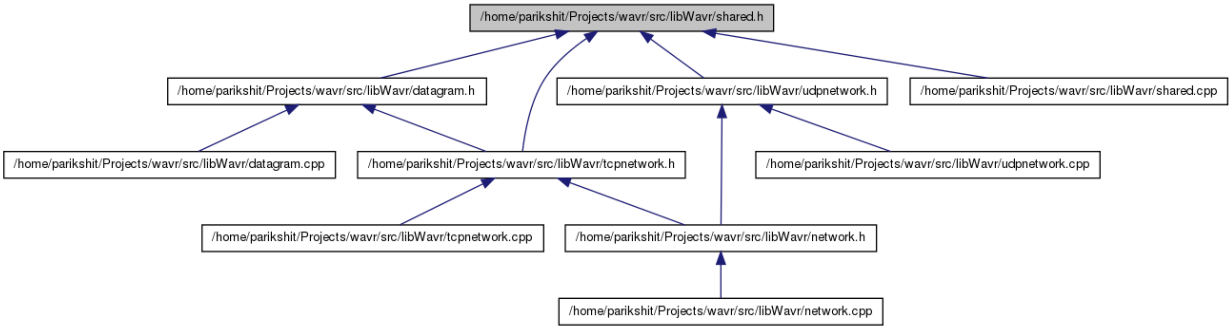
TCP Network



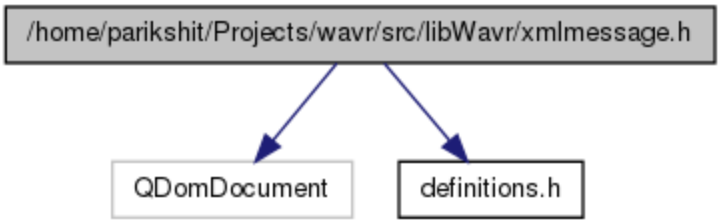
UDP Network



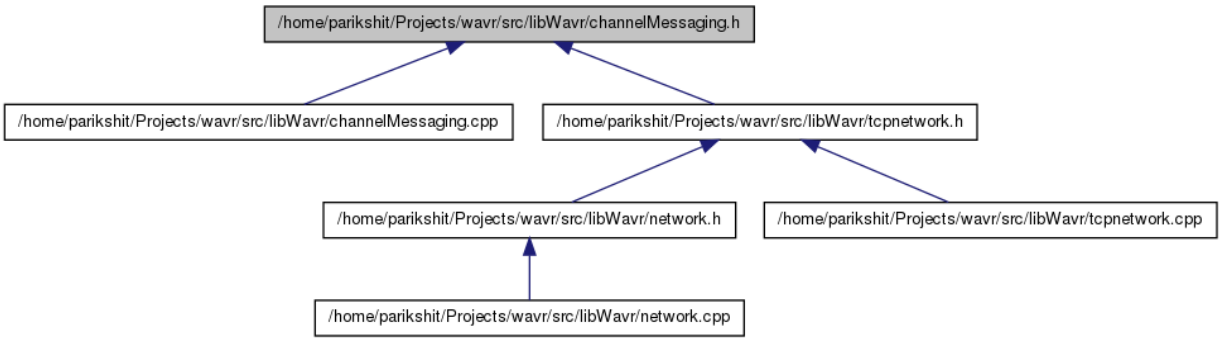
Shared



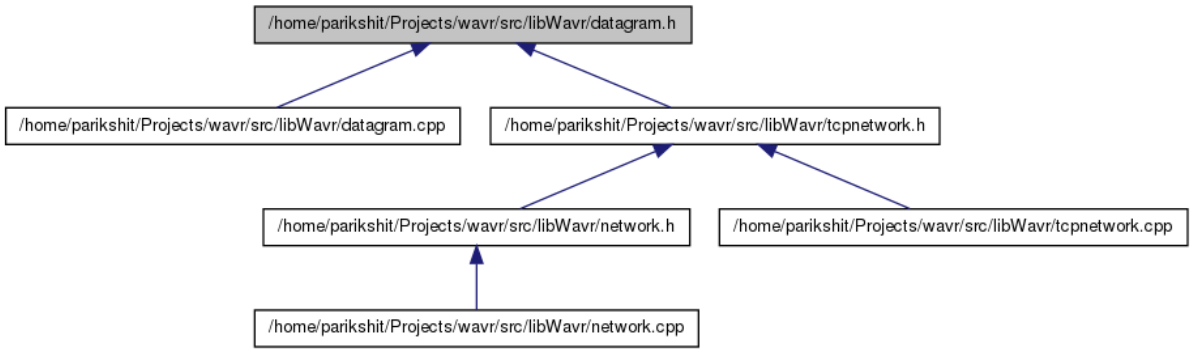
XML Message



Channel Messaging



Datagram

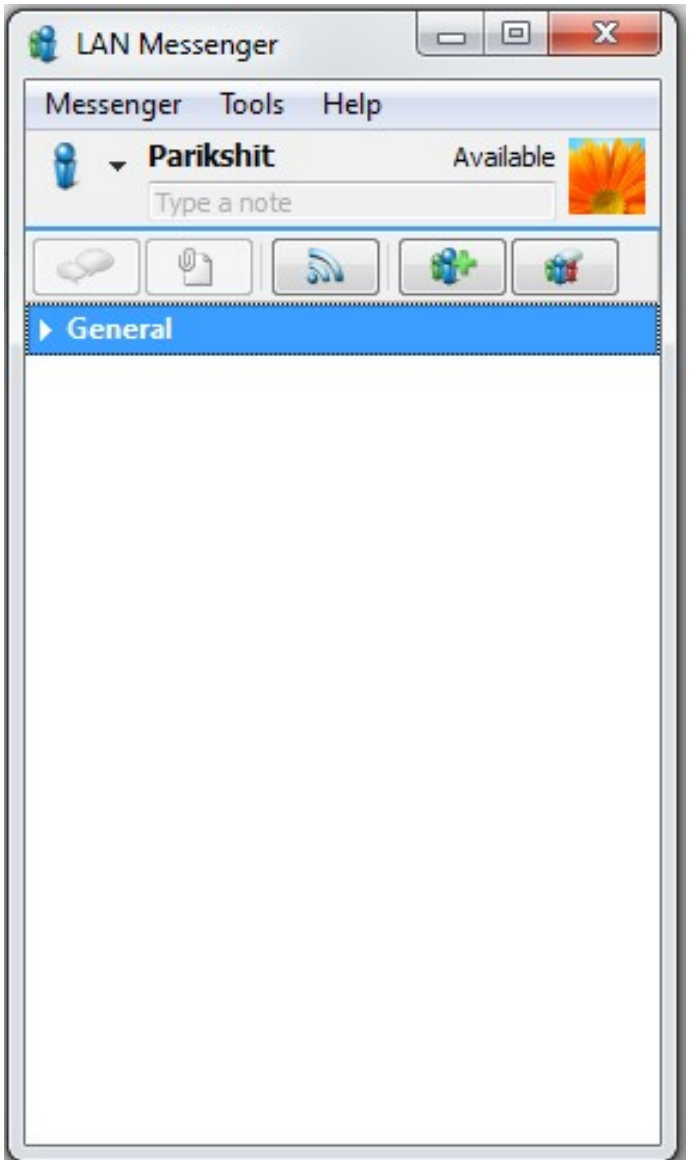


Screenshots

Main Window

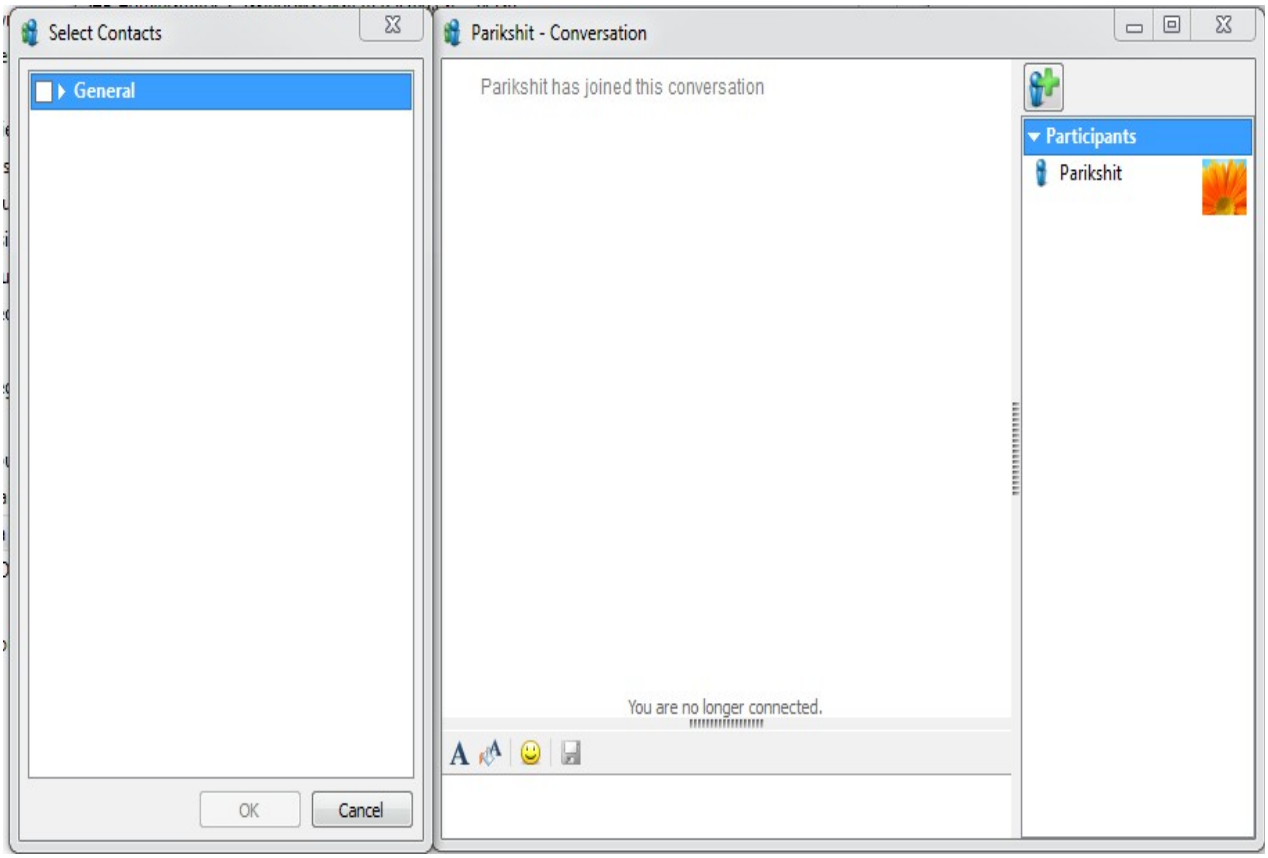
This class directs the operation of the GUI: it contains the main window (such as the name implies), start a New Chat application.

Main Window is the main screen of the application containing all the peers name available for chatting and we can set our presence, avatar and name here



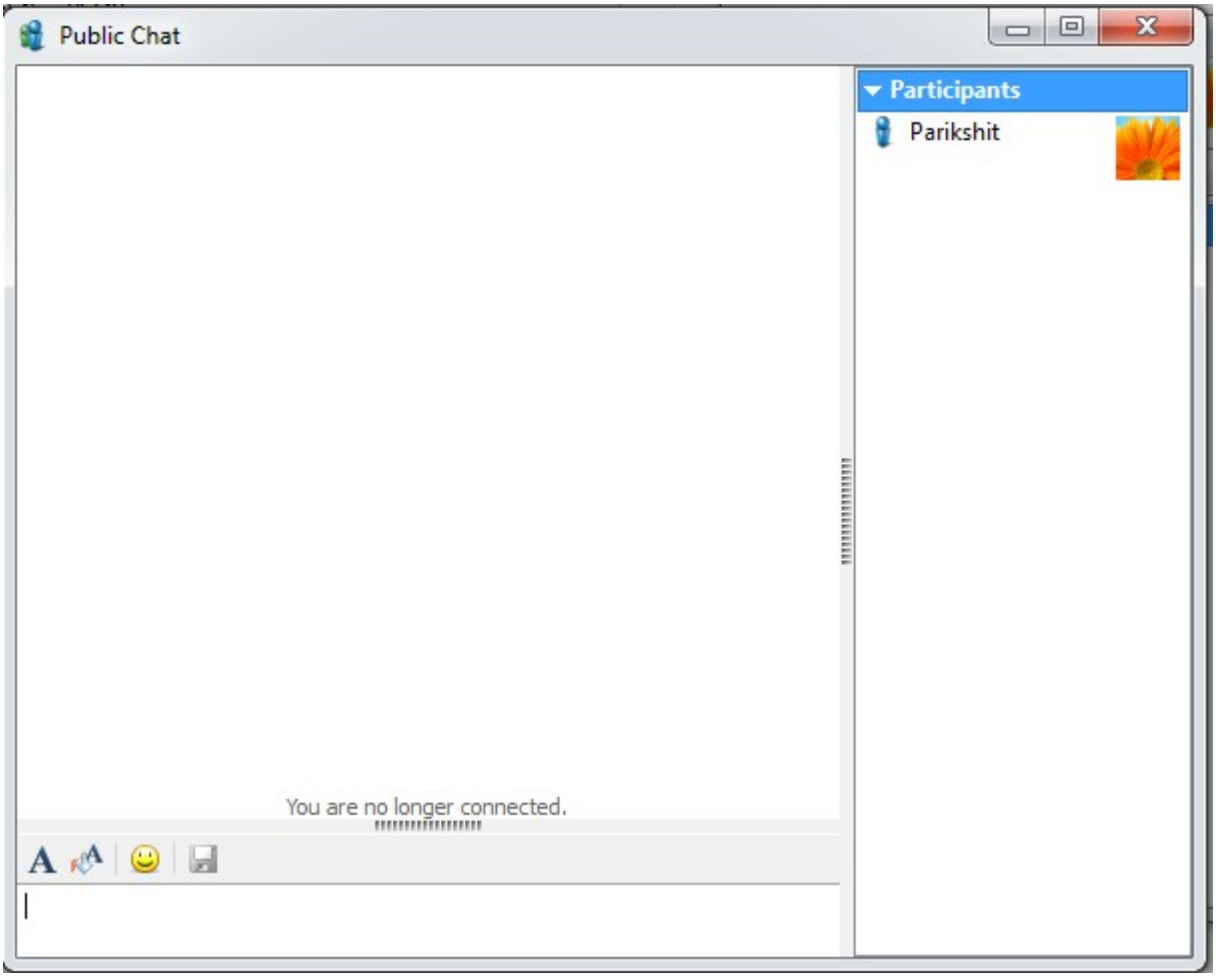
Chat Window

Chat Window is specific to the peer to which we are chatting, and contains all the messages send and received from that user. The incoming messages from the peer will appear on this screen.



Public Chat Window

In this screen we can chat the specifice peers which we want to chat. Via this we can able to chat with the multiple users simultaneously.

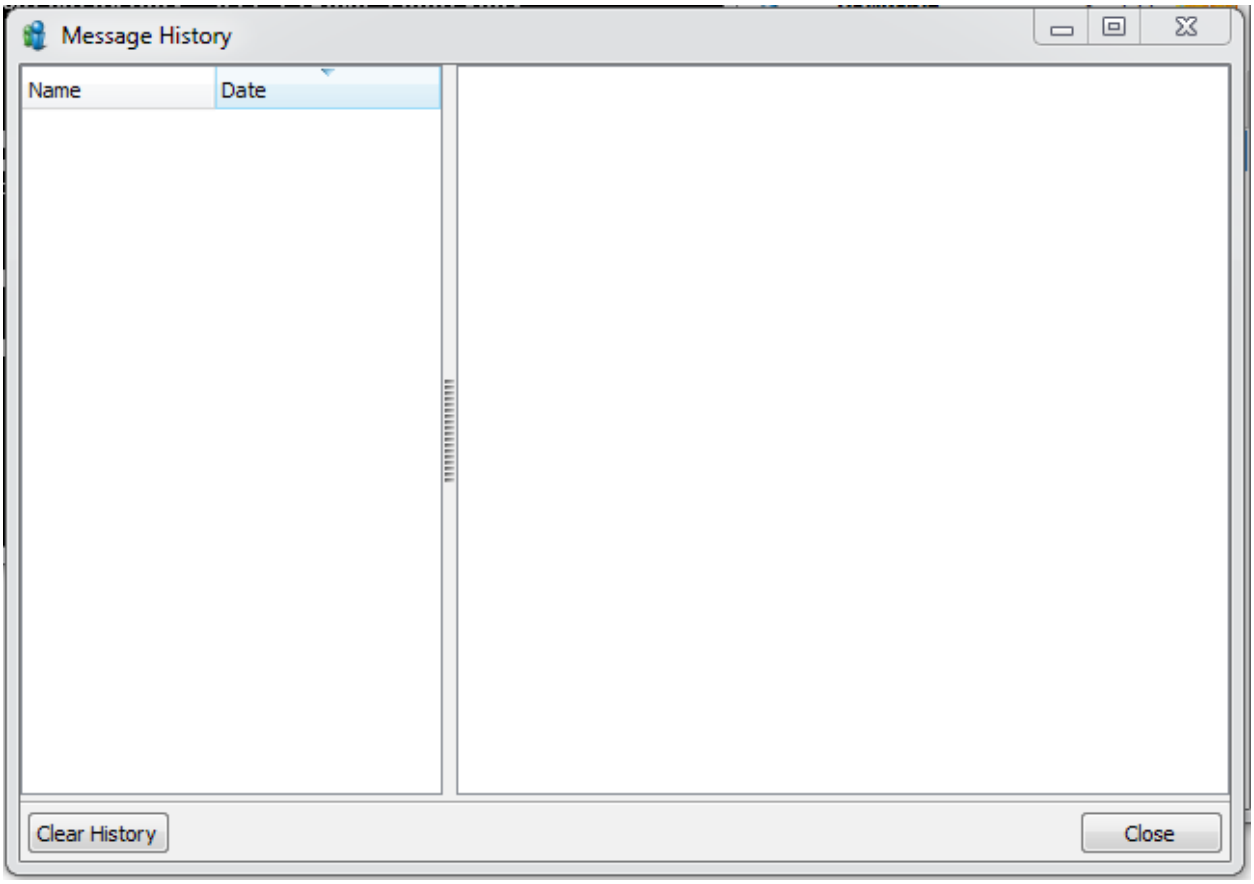


History Window

Holds the list of past messages or the history of messages which we have send or received over the network. This window can also used to save the history logs for future use.

Options included in history are clear history which is used to clear the history of the messages

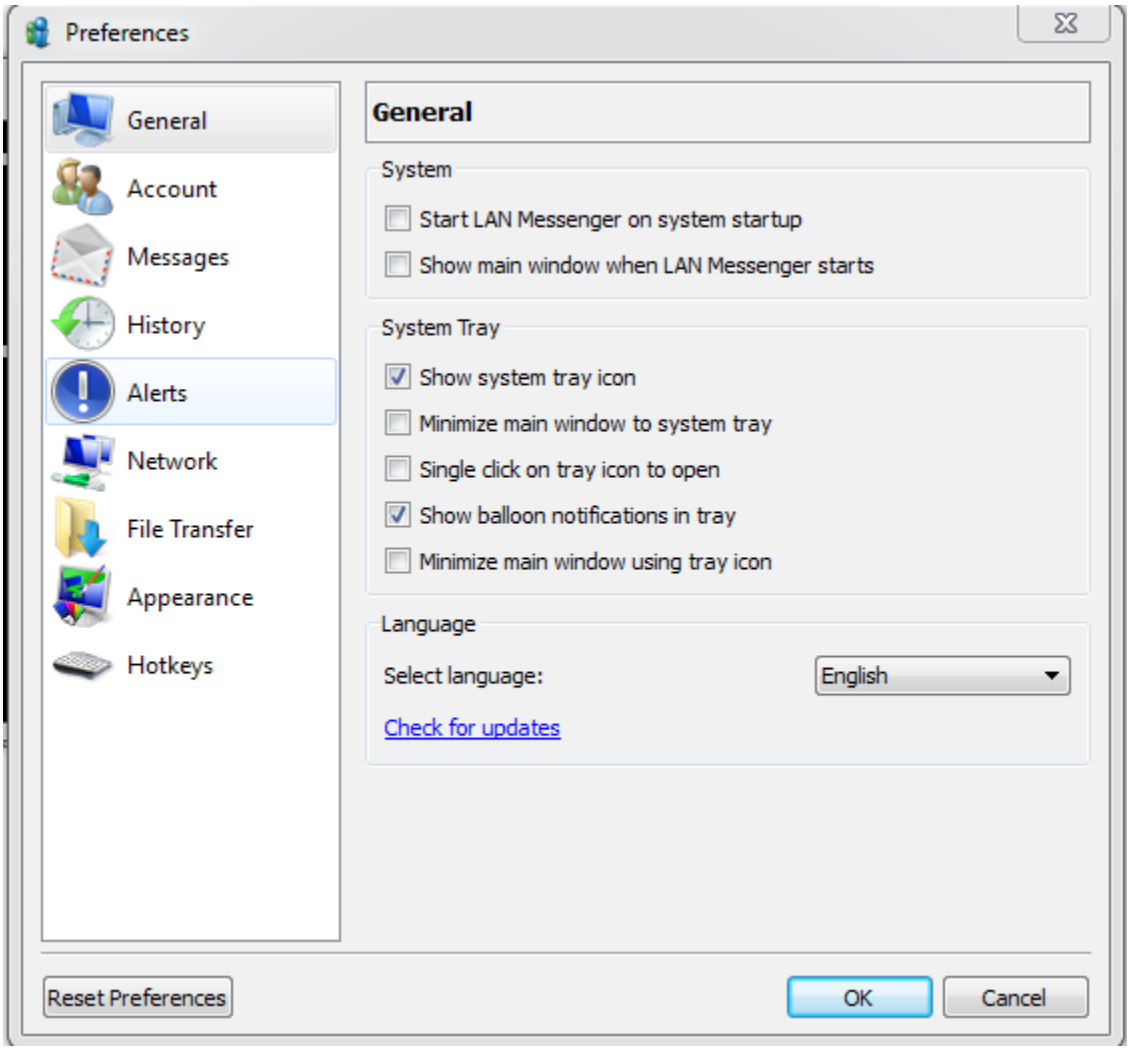
This screen shows the name of the peer along with the date of the conversation which takes place.



Settings Window

Settings window is used to specify the settings of the application.

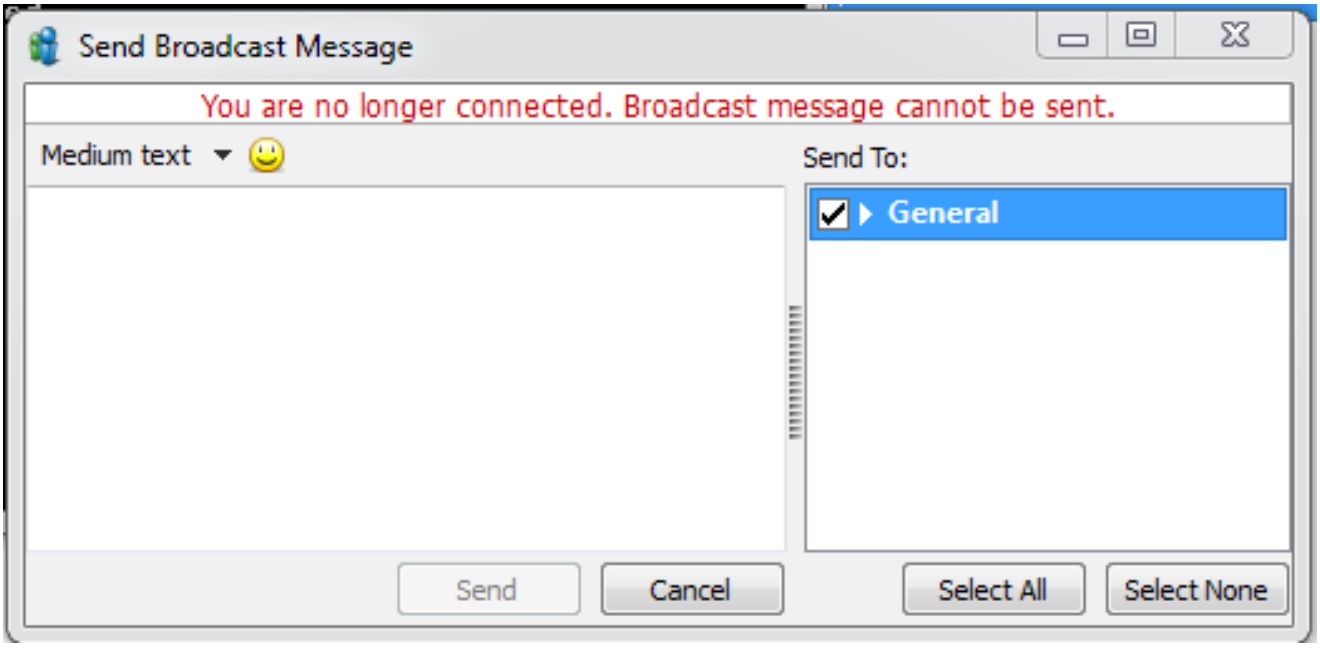
Such as starting lan messenger on system startup and account, messages, history, alerts, network and appearance related settings.



Broadcast Window

Broadcast window is used to broadcast messages to all the peers showing in our list.

It also contains the peer list to which we have to broadcast the messages.



Testing

The software support modules will need to be tested and debugged to certify proper operation . A standardized software testing has been created to facilitate the testing procedure. The software form includes areas to verify that data functions/modules function properly.

This will consist of creating test data, where the results are known, and comparing them to the output of the program. In addition, this form will test the interaction between the software module and ensure proper operation.

Lastly the integration test will confirm that the complete software system operate as expected.

Testing procedure will include:

Unit Testing

Individual module will be tested separately. For each of the Components, test will be conducted until the expected results are obtained.

Integration Testing

The component will be tested as a whole to test the orchestration of the modules. Special test data will be created to control the system as a whole. The database module will be tested to ensure proper communication between the interface and the database.

High Order Testing

This test will be performed on the complete, integrated system. Alpha testing will take place. The system will simulate the system with fake inputs. A comparision between the program output and desired values will be discussed.

Beta Testing

The tester will be asked to help with the final testing phase. Each beta tester will be given a copy of the software, and the preliminary help files. Beta testers will be expected to submit bug reports and any opinions they have concerning the software (especially the interface layout)

Conclusion

During the desing process, we search new technologies, alternative desing approaches, thought about what we can add to distinguish our system with others. Finally by the layered approach the system is gained more flexible and easily maintainable structure. With the help of it, we separate the task into module easily. This helps in making our coding process easier, less time consuming and more efficient.

Unified modeling language show the power of engineering rather than programming. Use of cascading style sheets helps in making the better GUI for applications. Qt provides us with better GUI and helps us with better and efficient libraries that help us in making more interactive and efficient software.

Bibliography

- Qt-project.org
- En.wikipedia.org
- Stackoverflow.com
- Youtube.com
- W3schools.com
- Coderanch.com
- Qt.net