PROJECT NAME: daBlaqChat

PROJECT OBJECTIVE

The main objective was to test what I had learned, so I had to build a network focused and aware Instant Messaging Application that will run on Linux CLI and be capable of communicating across multiple access networks.

The Instant Message app would have the following mandatory features:

- Messaging between two clients
- Auto-discovery on a closed network
- Protocol selection based on access network (fixed - UDP or wireless - TCP)
- Signaling reduction techniques
- No failed deliveries
- Offline and Online notifications on chat initiation
- Must either use a client-server or peer to peer architecture

BUDGET AND PROJECT TIMETABLE

I used a Gantt chart for my TimeTable, there was no budget, and the resources I used required no expense. The project time table is attached below figure 1.

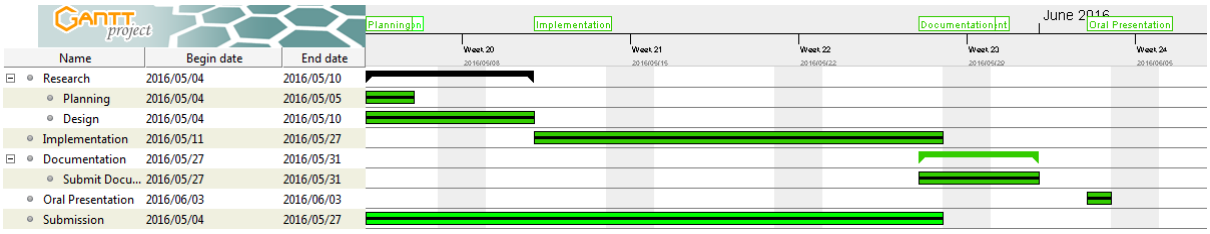

Figure 1: Gantt chart Project Time Table

PROJECT OUTCOME

Introduction

daBlaqChat is an open source chat application coded using Python language in socket programming. It is a Client-Server based model whereby clients communicate through the server. I programmed two separate applications, (1) the Server application and (2) the Client application. Please refer to figure 3 for the attached photograph of daBlaqChat.

Architecture

daBlaqChat uses the following architecture figure 2 to communicate and send data from user A to user B using the TCP stream socket. When both users connect to the server and once logged in, they can send messages to each other.

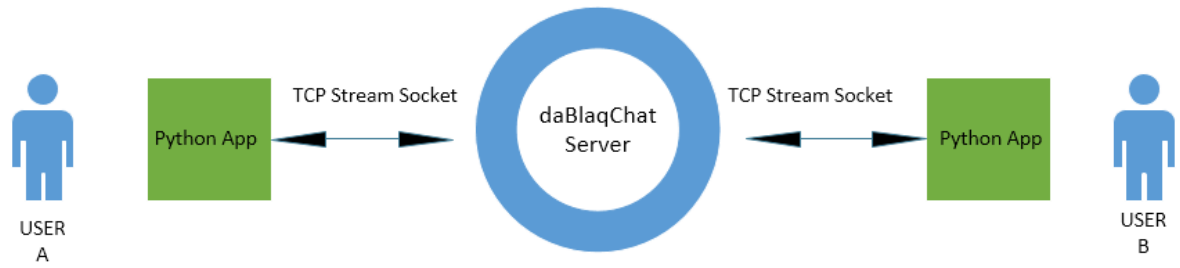

Figure 2: daBlaqChat architecture

daBlaqChat Server

The server was programmed using one class and 3 functions to achieve its purpose to transmit data to clients connected to it. It has the Listening function, the accept_new_connection function and the broadcast function.

First it will create a socket and then bind itself to the port number and its own IP address, after that is done, it will run forever. While it's running, it will listen on the port number it was bind to, for incoming connections, possible clients who want to connect. Now it needs to listen and accept clients at the same time, so it uses a select function that allows the server to take multiple clients and handle multiple connections while using the 'accept_new_connection' function to accept clients who wish to connect.
When the clients are accepted into daBlaqChat, the server stores their connections in an array so it knows is connected and if data comes in, it can send that data tom other connected clients. Now because the server can handle multiple clients, it listens for incoming data from the connected clients, when it receives the data it uses the 'broadcast' function to send the data to other connected clients, and they can all send and receive messages in the server.

The server uses TCP stream socket, a transport protocol that assures delivery of packets / data sent, when the data is not sent, it returns error to let you know some packets were lost or were not successfully sent. So the server has no failed deliveries when sending data between clients, and it tells clients that their messages have been delivered – a delivery notification.

Whenever a client goes online i.e. connecting to the server, the current connected clients receive online notifications, and when a client leaves i.e. closes the connection to the server, the current connected clients receive offline notifications.

After the successful completion of the project, daBlaqChat can:

- Message between two clients and more through the server
- Create a socket and listen on it forever on a closed network
- Accept new connections requesting to connect on the server's socket
- Use TCP stream socket to transmit data
- Broadcast messages to other clients except the sender – prevent wasteful resources
- No failed deliveries
- Online and Offline notifications on chat initiation

Strengths and Weaknesses

After I completed the project, I did a critical analysis of my project and I listed them below.

Strengths:
- Simple code structure
- Easy to implement
- Uses select function to handle multiple clients
- Uses TCP, end-point of a connection is guaranteed
- Messages are sent successfully, no failed deliveries
- Message resources are not wasted

Weaknesses:
- Only one class was used
- Client application is a script, but allows restructuring
- The code has poor extensibility
- Has a few bugs, but can be fixed

Trade-Offs:
- Used simple code instead of classes and functions (Client)

Conclusion

A simple basic chat server that allows users to communicate and exchange information over the internet. Programmed using Python language, both a Server application and a Client application interacting as one application unit to form a Chat Application running on a Linux CLI.

Figure 3: A screenshot of client 1 (bottom left) communicating via the server (top middle) to client 2 (bottom right) and vice versa.