README

ECEN 602 Network Programming Assignment # 02
TCP SIMPLE BROADCASE CHAT SERVER AND CLIENT

Team No: 11
Team Members:
1) Srividhya Balaji : UIN: 827007169
2) Sanjana Srinivasan : UIN :927008860

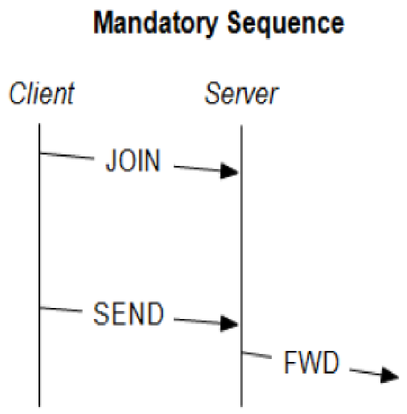
Individual Contributions:
1) Srividhya did the client code along with the Pack and Unpack functions for the server and client.
2) Sanjana did the remaining server code along with testing the usecases

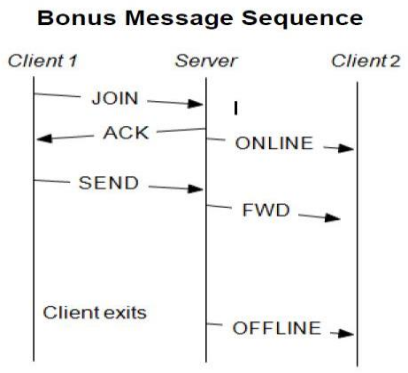
The package contains 5 files:
1. chatc.c
2. chats.c
3. Makefile
4. README.txt
5. Test Cases.pdf

EXECUTION INSTRUCTIONS:

To compile the code, run the Makefile: *make*
Run the server code using the command line: *./chats server_ip server_port maxclients*
Run the client code using the command line: *./chatc username server_ip server_port*
To clean the executables, run: *make clean*

IMPLEMENTATION OVERVIEW : ARCHITECTURE





A server maintains a chatroom in which multiple clients can try to connect and communicate or chat among each other. The client sends a JOIN message, requesting to be included into the chat-room. The client then continues by sending the SEND message which contains the actual data text. The server uses the FWD message to broadcast it to other clients. This sequence is represented by the first figure.

Beyond this basic connection sequence it is necessary to accommodate for various other NAK, ONLINE, OFFLINE etc scenarios in the chatroom implementation. This has been taken care in the bonus features, including sending NAK when same username is used, sending NAK when maximum number of allowable clients is reached, sending ACK when client has joined successfully, broadcasting online and offline status of clients.

These messages are sent in a format called SBCP format. This frame wraps up together the version number, message type, message etc in one single frame, thus making it easier to recognize and process the message.
The individual functionalities, messages used by both Server and Client are listed below ,

1. SERVER

The server was done using C. A Master fd is created to hold the connected fds, which keeps track of all the other fds created. Select() is used in order to manage both sending and receiving of data by single host.

Select() : This is used to know if the fd is ready to receive data. Used in order to handle several inputs at the same time in both the client and server.

The server performs the following functionalities,

1. Checks if any client user wants to connect. Regular accept and socket connection.

2. If an already connected user tries to communicate;
   2.1 JOIN request process based on max number of clients and send ACK or NAK
   2.2 Once joined, send ONLINE message to other clients
   2.3 When number of bytes received is equal to 0 send OFFLINE message
   2.4 Send IDLE when client remains inactive for a long time

The following message types are used to identify various kinds of ACK, NAK messages.

SBCP protocol message types :

```
sbcp_msg.type == '5' - NAK OR REJECT MESSAGE
sbcp_msg.type == '7' - ACK MESSAGE
sbcp_msg.type == '8' - ONLINE MESSAGE
sbcp_msg.type == '6' - OFFLINE MESSAGE
sbcp_msg.type == '9' - IDLE MESSAGE
sbcp_msg.type == '3' - FWD MESSAGE
sbcp_msg.type == '4' - SEND MESSAGE
```

1. **NAK** – In case of max capacity of clients ONLINE is reached or the username already exists, NAK is SENT

2. **ACK** – Server sends ACK to client to confirm the JOIN request

3. **ONLINE** – In the presence of a new client in the chat room the status is set to ONLINE which is then broad casted to the remaining clients in the chat room

4. **OFFLINE** – When a client leaves the chartroom, the status is set to OFFLINE which is then broad casted to the remaining clients in the chat room

5. **IDLE** – A client that has been inactive for more than seconds after sending a message, sends an IDLE message to the server indicating its state. The server then broadcasts to other clients the use name of the IDLE client

6. **FWD** – When a Server receives the SEND message from a client it forwards FWD messages to other clients in the chat room

FUNCTIONS AND DATA STRUCTURES USED:

The following structures were used, in order to represent the message header.

1. struct SBCP_attribute
Is a structure containing the SBCP protocol’s attribute fields.
It contains the following fields ;
   1. Type - Size of 2 bytes which indicates the SBCP Attribute type
   2. Length – Indicates the length of the SBCP attribute
   3. Payload – Contains the attribute payload

2. struct SBCP_message
Is a structure for the SBCP message field. The complete header is of this form.
It contains the following fields;

1. Vsrn – Protocol version 3
2. Type – Indicates the SBCP message type
3. SBCP_Attribute

Below are the functions used in the server code,

1. initialize_sbcp_attr():
Function to initialize the SBCP attribute structure's object
2.initialize_sbcp_msg()
Function to initialize the SBCP message structure's object
3. packi16() and pack()
Function to pack the input message in SBCP protocol format
4. unpack() and unpacki16()
Function to unpack the message in SBCP format back to the input message format
5. sigchld_handler()
Signal Handler in order to effectively handle a zombie process
6. **Main function()**
This has all the select, JOIN, other feature implementations.

2. CLIENT

The client was primarily done in C format. The client performs the below functionalities.

1. Connects to a server.
2. Performs select() in order to choose between the sent and received messages.

- Sends JOIN request to server packed in header
- Select chooses to receive over send and processes messages in format to readable form
- Once ONLINE, packs and sends its username to all through the server. Further, sends messages also in the same way
- Also, when it receives a message it unpacks and processes based on the message type.

Data structures and functions used in the client are similar to those used in the server.

BONUS FEATURES IMPLEMENTED

The following Bonus Features were implemented

1) New message types : ACK, NAK, ONLINE and OFFLINE were implemented and also keeping track of these statuses from each client and broadcasting to other clients.
2) IDLE feature was also implemented. If the client remains idle for more than 10 seconds it sends an idle message to server which then broadcasts to other clients a message which provides the username of the idle client.

REFERENCES USED :

- UNP Vol-1, 3rd Edition Stevens et al.
- Beej's Guide to Network Programming
- https://docs.python.org/2/library/struct.html :For Pack and Unpack
- https://www.stat.berkeley.edu/~spector/extension/perl/notes/node87.html : For Pack and Unpack