Design Document FOR Chat Protocol

Computer Networks - CS-544

Srijan Pandey

March 2, 2021

1 Service Description: Chat Protocol

This protocol allows communication between two users through a common chat server. This protocol is an application layer protocol and is in the same hierarchy as HTTP, FTP etc. By the use of servers and clients, this protocol helps users send messages amongst each other.The server is a single contact for multiple clients to interact with and relay messages to other clients. Clients are user facing agents that can generate messages for servers and other clients to consume. Each time a client connects to another client or group of clients, a channel is created through which messages can be transferred. The channel is automatically closed if there are not sufficient users for communication. The protocol allows real time communication between two client computers over the internet.

For the initial implementation, this protocol facilitates communication between one server and multiple clients. The messages are sent over as commands and parameters, and these commands from clients elicit a response from the server. The message transferred between the clients are in UTF-8 format.

As full duplex communication is required for effective chatting, transfer mode between clients is asynchronous mode, which means that clients will not wait for response from another client. There is a need to identify end of packets of data, which can be done through a special character (delimiters) which does not appear in text conversations in sequence.

The steps to communicate between clients is: First, the client establishes connection through authentication and mode negotiation. Once connection is established through this mode, data can be transferred between clients. Finally a special command indicates the end of connection and the channel is closed.

This protocol relies on Transmission Control Protocol(TCP) in the transport layer and through its service provides reliable connection oriented service to the user. Any error and re-transmission is handled through the transport layer and the application layer deals with delivery of message to end user. The TCP provides error control mechanisms (such as sequence, acknowledgement numbers and checksums) to ensure reliability of service. This ensures that the messages are correctly ordered and re-transmission of failed messages is done. The hierarchies in the TCP/IP stack can also be seen in Figure 1

The initial implementation contains two mode of communication:

1. One to One Communication: In this mode of communication, a client is connected to another client through a server. There is a unique pair of client, who will communicate with each other. This is the 'U'(short for user) mode in the MODE command seen in Table 2.1.

2. Many to Many Communication: The client has an option of communicating with multiple clients over the network. This is a full duplex multi-cast mode communication. Multiple clients can connect to the server and talk to one another. This is the 'G' (short for group mode) in the MODE command seen in Table 2.1

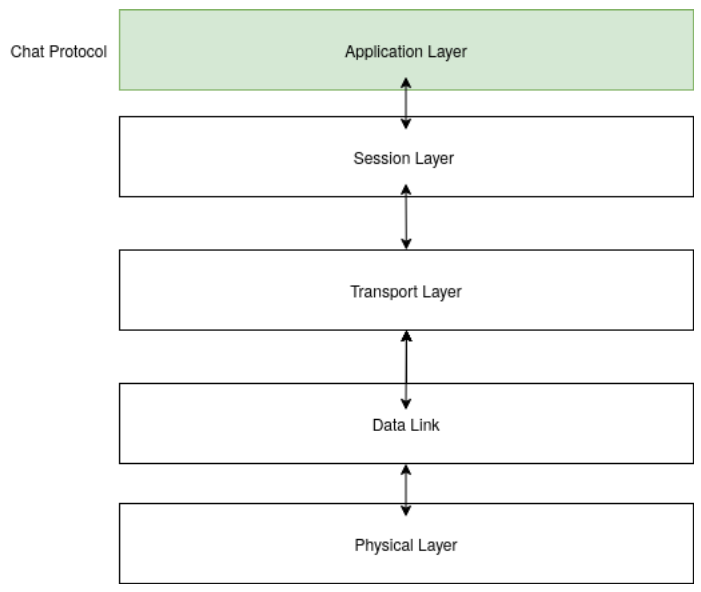

Figure 1: OSI Hierarcy Chat Protocol

2 Protocol Data Unit

2.1 Request Messages

The request and response structures of standard PDU messages are given as shown in Table 2.1 and Table 2.1 respectively. The messages can be separated by one another using Carriage Return(CR) and Line Feed(LF) combination.
Before the user can start communicating with other users, the username and the password must be registered with the server. Users can be registered by Admin users only. Upon admin user login, admins can register new user and password using commands REGU, REGP respectively. This acts as a layer of security to ensure that hackers cannot anonymously register themselves and start communicating or intercepting messages.
Whenever a user wants to start transferring messages through the protocol, the user will initiate authentication using the USER and PASS command. This acts as an authentication mechanism to identify the users that are registered in the system. The user connects to a static port number (5001) defined by the server for communication. The port number used by client for message transfer can be notified using the PORT command, and MODE command identifies the type of mode to utilize for communication. As mentioned in Section 1, there are two modes of communication and the mode type identifies the type of MODE the user communicates in.
The MODE methods to connect to a username in case of one to one communication and group in case of multicast communication mode. The MODE uses two parameters, the first being 'U','G', ie the type of mode, and the second being the user or group to connect to. Users can identify online users using LIST U(for users) or List G (for groups).
Once the chat is completed the user issues a BYE message. This indicates end of conversation for user. Depending on the type of mode, the user ends the connection when user left in the channel is one.

The CHAT command is used to send and receive messages from the protocol. The messages sent and received using CHAT command is sent over the port identified by the client. The server sends the message to the client, with the message and the username of the client. The server adjusts the message size, such that it doesn't exceed the maximum size of the PDU. In addition, server also has mechanism to check the maximum size and will not accept values that are over 256 characters in length.

Command	Argument	Description
SUSER	Username	User Authentication for admin user
SPASS	Password	Password Authentication for admin
REGU	Username Password	Username and Password registration by admin user
USER	Username	Provides the username of the user
PASS	Password	Password authentication for user
LIST	U or G	Available list user or group list
MODE	'U or 'G'	Selects mode 'U' or 'G' with username or group name
BYE		Logout of System
CHAT	Message	Messages are sent
VERSION		Version Information Request

Table 1: Request Messages

2.2 Response Messages

The response messages are identified by status code followed by additional description about the message. There is a generic error and password authentication failure message available in the protocol. Generic Error handles most error, and thus acts as a universal message for the system. In addition to that , the protocol also has numerous success messages based on the type of message it has transmitted. The error message starts with 500 series of status code where as success and other general message starts with 100 series of message.

Code	Description
100	User Registration Success
101	User Authentication Success
102	Pass Authentication Success
103	Mode List Success
104	Chat Received
105	Mode Selection Success
106	Version
107	Bye
500	Generic Error
502	Authentication Error

Table 2: Response Messages

2.3 Message Format and Delimiters

/textcolorblack The request message contains upto three main parts. The first one is the message command , parameters, postfix. Each of these sections are separated from each other using | and start and end of message is represented by « and » . Both the request and response messages are of static size of 256 characters. While the message internals can varry in size, the total size of the message remains constant.

The response message uses string code followed by a delimiter followed by the description. Although initially a big-endian numeric representation was proposed, it made sense to make the whole message into character stream as it made the implementation easier.

2.4 Other Concerns

Flow control mechanism must be implemented in the algorithm to ensure that clients don't flood the system with their messages and deny service to other users. For this, clients can only transmit 1 message in .3 second. The error control mechanism is left to the TCP to handle. Sequence number and acknowledgement number of the TCP is used to re-transmit any missed or error prone messages. The messages that have error can be verified using the checksum in TCP protocol.

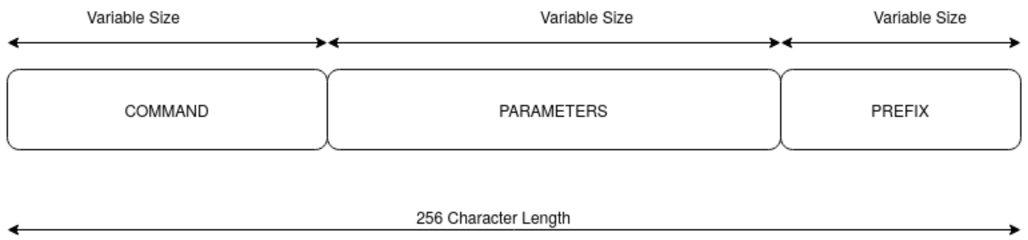

Figure 2: Request Response Message Format

3 DFA

A high level representation of the states looks as in Figure 3. The first phase for communication is authentication and mode selection. Once these options have been negotiated by the user, they can start communicating with one another. When communication is established the server and client are in wait state. The server waits for incoming messages from client and forwards it to the secondary client. Likewise, clients wait for messages or write and send their own messages. This is done by Wait for Message/Input state in DFA.

In this state, the protocol can send new messages or receive messages sent by another client. When a new message is received, the protocol goes to Display/Relay Message state, where the received message is displayed in case of a client or relayed forward incase of a server.

When the chat is complete the user sends out a BYE message to indicate that it wants to opt out of chatting. Once this message has been received, the connection is terminated with the client.

4 Security

For the purpose of security, there is an authentication mechanism available. Only when username and password are successfully validated can a user communicate through this protocol. The mes-

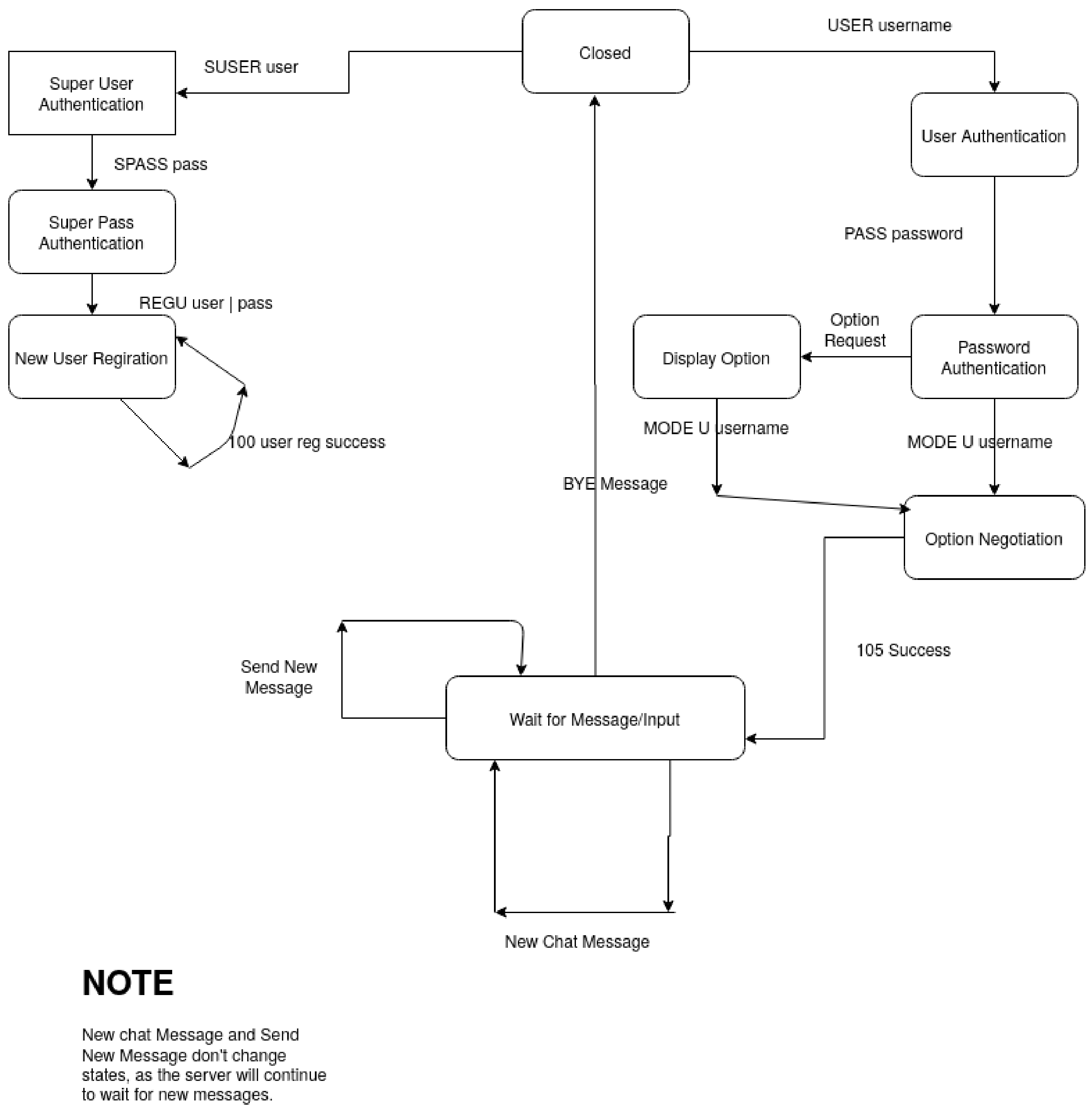

Figure 3: Discrete Finite Automata

sages are not inherently encrypted in this protocol and thus this protocol uses Secure Socket layer to add additional security.

Before chatting through this protocol, users must be registered to this protocol. However, not any user can register new users to the protocol. A special privilege user must login and add new users to the protocol. To do this, SUSER and SPASS is used to login as admin users. Once logged in as admin, REGU are used to add new username and password to the protocol.

5 Extensibility

In order to extend the protocol. There is a version command that identifies the current version of the server. This command can be used to extend the protocol and have clients or additional servers (future scope) running different versions of protocol to connect and communicate with one another.

The size of the message also allows for extensibility of the protocol. With a 256 character PDU, multiple number of additional commands can be added into the existing protocol. In addition to this, to accommodate additional sub commands for each command, the postfix feature is available (see Section 2.3). Furthermore, protocol also allows upto 10 parameters which gives sufficient extensibility.

References

[1] RFC 1889, RTP: A Transport Protocol for Real-Time Applications
`https://tools.ietf.org/html/rfc1889`

[2] RFC-1459 IRC: Internet Relay Chat Protocol
`https://tools.ietf.org/html/rfc1459`

[3] RFC-3550, RTP: A Transport Protocol for Real-Time Applications
`https://www.rfc-editor.org/rfc/pdfrfc/rfc3550.txt.pdf`