Drexel University

Winter 2020

CS544

Professor: Mike Kain

PROTOCOL DESIGN
EM Chat Protocol (EMCP)

March 13, 2020

Elizabeth McLaughlin

eis29@drexel.edu

ID 11764593

1 Changes to Original Design

The implementation follows very closely to the original design. Minor changes to the actual chat behavior were made, and the application is a single shared chat. Other changes listed below:

1. Minor updates to DFA. See section 4.
2. Some reply codes not used, and others added. See section 3.2.
3. Methods sent by client differ from design. See section 3.1.

2 Definition of Service

EMCP is designed for use in a client server paradigm, where messages are passed from client to server, and forwarded onto the designated recipient(s). The objective of this protocol is to facilitate a chat session between two or more participants. It promotes reliability and efficiency of message passing, while maintaining data integrity across differing machines and operating systems.

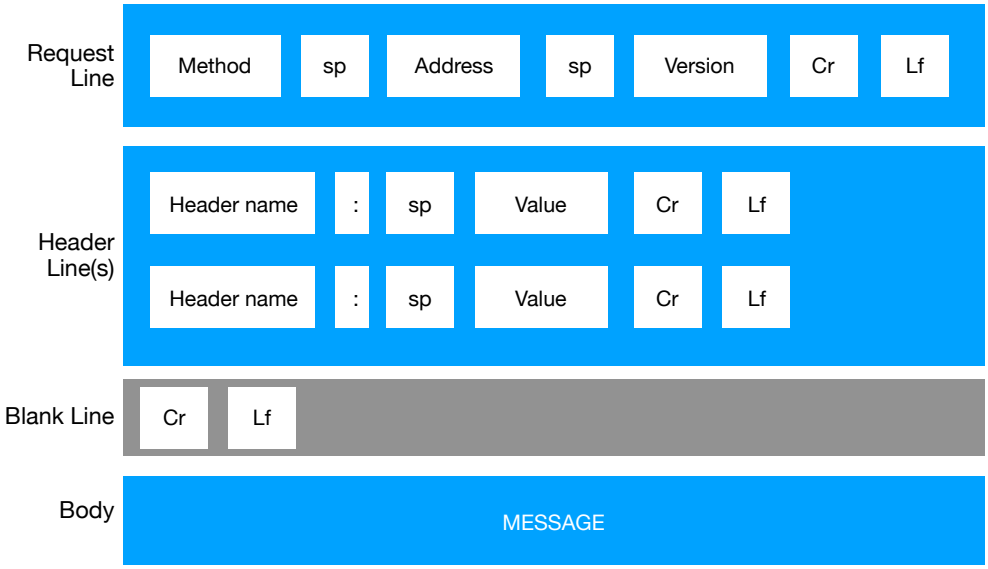
The protocol transfers message in bit format to connected users. All applications using this protocol must have the ability to decode from type ASCII. A string is defined as a series of ASCII bits ended with the <CRLF> sequence. A client's logical byte size is padded as necessary and converted to 8-byte ASCII representation before transfer to and from the server. The transfer byte is 8 bits. Messages are transferred as a continuous sequence of data bytes (8 bits). The protocol is designed to be run over TCP, which handles reliability/error control of the data transfer.

The protocol runs on port number 33318.

3 Message Definition (PDUs)

The structure of my PDU diagrams are borrowed from the text, as found for HTTP (Forouzan and Mosharraf, 2012, 51).

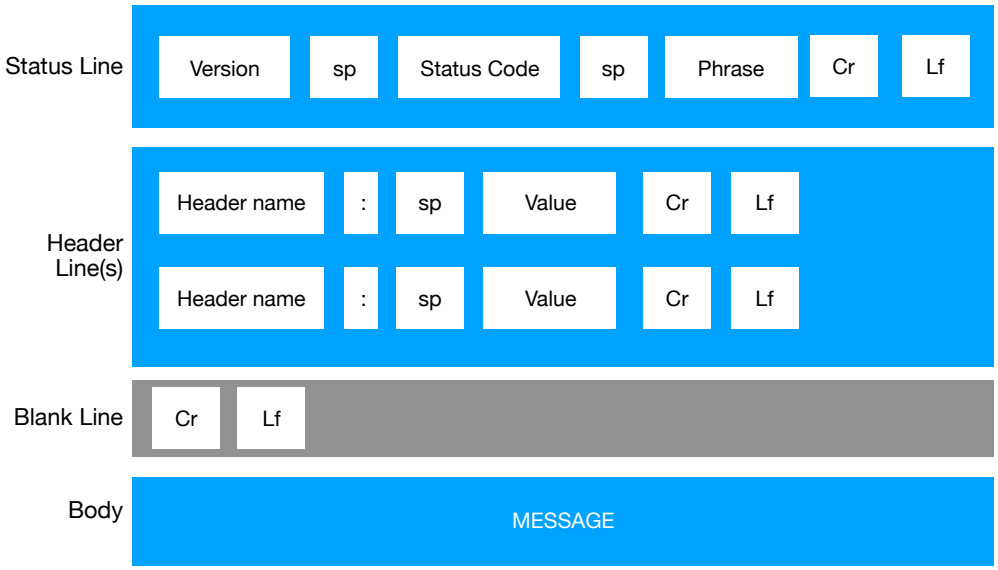
3.1 Client (request) Messages



Messages are send as a continuous stream of bits with the below structure. A string is defined as a series of ASCII bits encoded with a newline. The request line (top line) contains the method (32 bits), the client's socket address (32 bits), and the version of the protocol (4 bits) the client is using. Elements of the request line are separated by a space and end in a carrige return, line feed. The 32 bit socket address allows for all current possible addresses with sufficient cushion for growth, and the 4 bit version numbers provide 2^4 different version possibilities. As this is the first instance of the protocol, "version" at the time of this design will be 1.0 for all messages.

There are four methods a client may choose from when communicating with the server:

1. 0001-LOGIN REQUEST: checks client credentials to enter chat
2. 0010-BROADCAST: broadcasts message to currently connected clients
3. 0011-BROADCAST NEW: broadcasts notification of new client connection to currently connected clients
4. 0100-QUIT: request from client to close connection



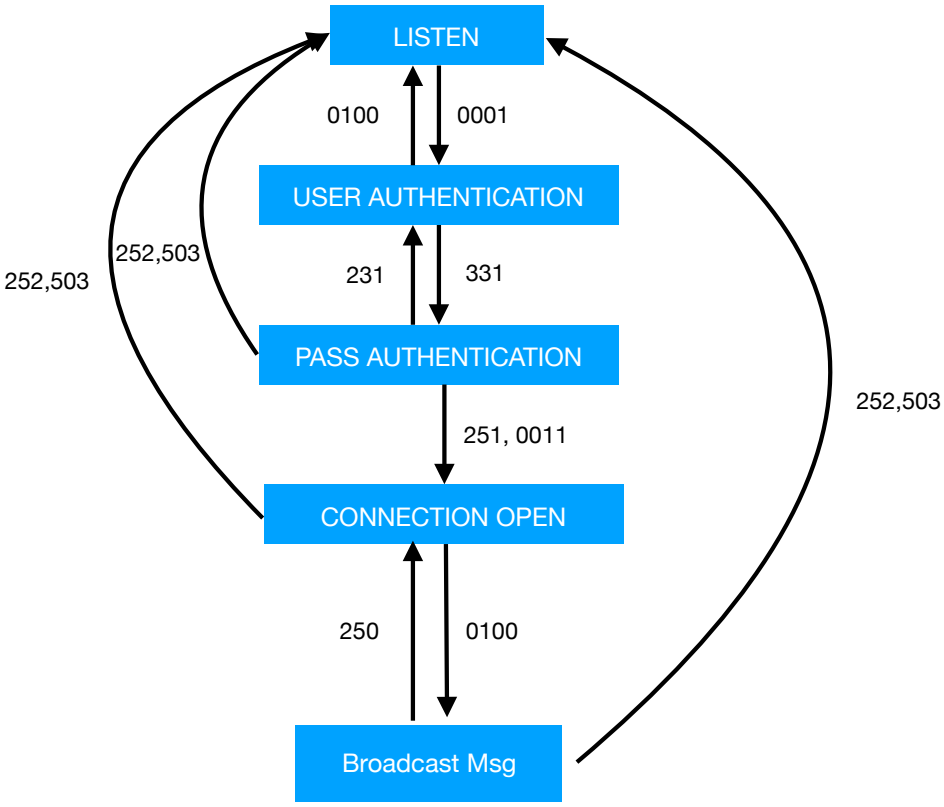
3.2 Server (response) Messages

For response messages, status line contains the version (4 bits), the status of the request (4 bits), and status in plain English (32 bits). Possible status codes with descriptions are listed below:

Reply Codes	
331	Login Success
231	Incorrect username or password
250	Message action complete / message sent
251	New Connection msg action completed
250	Msg action completed
252	Client connection closed
503	Error

Optional header lines are returned when the server wants to include the username of the client wishing to broadcast a message. This enables each client program to concatenate the username to each message as it appends the message to each chat window.
Header names:

4 DFA - EMCP



5 Extensibility

Inclusion of the version number in the request and status lines enables the protocol to be updated in the future. If future design changes alter the design significantly, the client would specify which version it's using on the request line when initiating a connection. The server then knows which version the client can accept. The structure of my PDU also allows easy addition of commands as needed.

Requiring any application using the protocol to encode/decode from only ASCII encoding minimizes complexity when updating. Accepting other types would add unnecessary complexity with unintended side effects when changes are made in the future. In addition, EMCP ensures extensibility by allotting sufficient space for header field names. The version field is 4 bits, which provides an extended quantity of version numbers possible should we

need them.

The socket address field is set to 32 bits. Choosing 32, instead of 16, ensures we can cover present and future addressing needs as number of network users, applications and protocols increase.

One consideration when planning for extensions is greater security risk. The only security our protocol handles is the user authentication process when a client initiates a connection with the server. Since we layer over other protocols for any further security needs, we limit the amount this needs to be mitigated when extending in the future. More complexity with built in security would have side effects and open vulnerabilities with later updates to the protocol if not handles with care. Another argument in favor of layering our protocols!

6 Security

Security is provided by an authentication process at the initiation of the connection request from client to server. The server validates a client's identity, via login name and password, before opening the connection and allowing message transmission. Client login names and passwords are stored on the server for later access.

The DFA provides security as described in lecture. By checking which state we are in, a server can cross check that any messages received are appropriate for the given state, and reject if not.

Security is not of great importance for a simple chat protocol, as the cost of any breach is trivial. Should the protocol be extended for other, more sensitive services, security measures can easily be added with layering. For example, further security methods such as encryption can be obtained by layering over additional, security specific protocols.

References

Forouzan, Behrouz A. and Firouz Mosharraf. 2012. *Computer Networks A Top-Down Approach*. McGraw-Hill.