*Chat Protocol*

*Manikishore Medam*

*Class: winter 2020*

*Department of Computer Science*

*Drexel University*

Table of contents:

|  |  |  |
| --- | --- | --- |
| S.no | Topic | Page no |
| 1 | Service Description | 2 |
| 2 | Message Definition (PDUs) | 3 |
| 3 | DFA | 4 |
| 4 | Extensibility | 5 |
| 5 | Security implications | 6 |

**1.Service Description:**

A close up of text on a white background

Description automatically generated

Chat protocol is used to establish communication between client systems. It works in the application layer of the OSI model. It runs on the TCP protocol of the transport layer. This protocol is used to transmit text messages between clients. The basic system will consist of server and clients that want to communicate. This chat process works on client and server model of networking. To establish chat first we need to run the server, once the server is on it will look for the clients that are trying to establish connection. Clients will connect to the server using host IP(127.0.0.1) and port number(3300), once they give these details they will be asked to authenticate where they will be providing their user id and password. Once the user is verified the user will establishes connection with the server and it will be added to the chat session. Similarly any number of clients can establish connection with the server and once connection is successful clients can communicate with each other. Whenever client sends a message it will be received by the server which will be forwarded by the server to other client. By this way messages will flow between clients in the chat session.

**2. Protocol Data Unit:**

A screenshot of a cell phone

Description automatically generated

Abbreviations for the header and trailer is given below,

|  |  |  |
| --- | --- | --- |
| **PDU type** | **Abbreviation** | **Data Type** |
| vr | Version of Protocol | Integer(unsigned) |
| Time stamp | Time stamp for the message received | string |
| len | Length of message | Integer(unsigned) |
| Error checking | Error checking | String(hex code) |

The PDU for chat protocol consist of a header and trailer. The header consists of version, time stamp, reserved bits, length of message and Error checking.

**Version :**

The version in the header is of 1 byte length and it is used to indicate the version of protocol. The version will be appended at the receiver end which will be used by the client at the receiving end to receive message in correct format.

**Time Stamp:**

The timestamp in the header is of 8 byte size it is used to store the time at which that particular message is sent by client. It will be stored as a string which will be converted into a stream of bits for transmission over the network.

**Len:**

Length in the header is of the type unsigned integer and it is of 2 bytes. It indicates length of the message that is sent by clients. The maximum number that can be represented by 2 bytes is 65535, so it is the maximum length of the message that can be transmitted in the chat protocol.

**Reserved Bits:**

There are also reserved bits of size 1 byte which can be used to implement any future enhancements.

**Message:**

Message is the data part for the chat, these are the messages that are sent by a client to other client that are part of chat. All messages will be converted into a stream of 8 bit ASCII codes and will be transmitted over the network.

**Error checking:**

The error checking part in the trailer is of the size 16 bytes, md5 has been used for this error checking. Md5 creates a 128 bit hash code(HEX code) for the message that is transmitted and this will be verified by the receiver at the receiving end by generating a md5 hash code again for the message that is received. This will enable us to know if message is corrupted and will help us to ensure that protocol is working correctly.

**Delimiter**: The delimiter for the protocol will be ‘¦¦’. Based on this delimiter all the parts in the header will be divided.

**Note:** All the data in the protocol will be represented in Big-endian format.

**3. DFA:**

**A close up of a map

Description automatically generated**

**Messages:**

Messages in the DFA are listed below,

1. User
2. Password
3. Invalid user/password
4. Join
5. Left
6. Success
7. Failed
8. Send/Receive Message
9. Resend
10. Error

State diagram can be explained as follows,

* Initially the protocol will be in idle state.
* On receiving the user ID, server will check if it is valid or not. If it is valid user ID then the protocol will move to Auth state. If it is not valid then the protocol will move back to idle state again.
* When user provides a valid pass word it will move to ‘Pass Auth’ state, if password is invalid then protocol will get back again to idle state.
* Once the authorization is successful the protocol will wait for the others to join and it will move to ‘Time-wait’ state where it will wait for other clients to join.
* If the client terminates in this state, then the protocol will get back to idle state again.
* If any other clients join chat session then, then the protocol will move to ‘chat mode’, if clients left then protocol will again go back to ‘Time-wait’ state.
* If all clients terminate from chat then the protocol will move again to idle state.

**4. Extensibility:**

**Versioning:**

The protocol has a version control header which captures the version of the chat protocol. The version part in the header is of 1 byte, so this protocol can be extended up to 256 versions. Whenever changes are made in the protocol the version number should be updated sequentially. Based on the header version the client will know the header format in the message.

**Reserve bits:**

In the header we have some reserved bits that can be used for future purposes. If in future we want to make any enhancements to the existing protocol this can be implemented using reserved bits. Using these reserved bits the protocol can be extended to implement handshake, option negotiation and many other enhancements.

Currently the protocol doesn’t have any encryption mechanisms for data transmission. The protocol can be extended to have encryption as well.

**5. Security of Chat Protocol:**

**1. Authentication:**

Access to the chat room will be provided for the client only after authentication. The authentication process is as follows,

* The client will share his user name and password with the server.
* Server will verify these credentials before establishing connection.
* If credentials are valid the client will be allowed in the chat session.
* If the credentials are not valid chat session will not be established with the client.

This will ensure right people accessing the chat server. However, there are no inbuilt techniques to encrypt data, as a result data will be transmitted in the form of plain text. So hackers can easily steal these passwords. This will be avoided by encrypting the data using using TLS/SSL.

**2. Encryption:**

The text messages that are transmitted between client and server are not encrypted. So the people can easily sneak into the channel and see the discussion happening between clients. This will be improved by incorporating some encryption mechanisms like TLS/SSL in future.

**Differences in Second Version:**

* DFA is updated based on your feedback
* Message list for DFA has been updated
* Updated Protocol header and added a delimiter in the header
* Data types and Endianness of data in the protocol header were updated based on feedback.

**Performance Implications:**

When comes to performance implications, I haven’t used any compression technique to compress the data, So in real world scenario it uses high band width and more time for transmission when compared with compressed data that is transmitted over network. Also there is no encryption as of now for the data that is getting transmitted. These can be fixed by enabling compression and encryption in future.