Independent Submission
Request for Comments
Category: Experimental

J. Qi
George Washington University

December 16, 2018

Chat Room Protocol for Implementing Online Chat Room

Abstract

This document describes a communication protocol for online chat room service, which allows users to chat with other users, and each session can only provide a two-person chat room. This protocol resides at the application layer and relies on Transmission Control Protocol (TCP) and Internet Protocol (IP) to transfer data between the client and the server. The communication messages are based on ASCII.

Status of this Memo

This document is not an Internet Standards Track specification; it is published for examination, experimental implementation, and evaluation.

This document defines an Experimental Protocol for the Internet Community.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained on the CSCI 6431 Blackboard discussion forum.

Table of Contents

1. Introduction

Chat Room Protocol (CRP) is based on Transmission Control Protocol (TCP), so that the information in the packets can be passed intact. The client and server model is used to implement this chat room.

1.1. Overview

Each user has their own userID. A user should know the userID for the user they want to reach out to before they can successfully connect to each other. After establishing connection, each user sends texts to the server via a client application, the server, then, forwards the texts to the other user. In each session, only two users can chat with each other, while a user can chat with different users in different sessions, which means the server can handle multi-threaded sessions simultaneously.

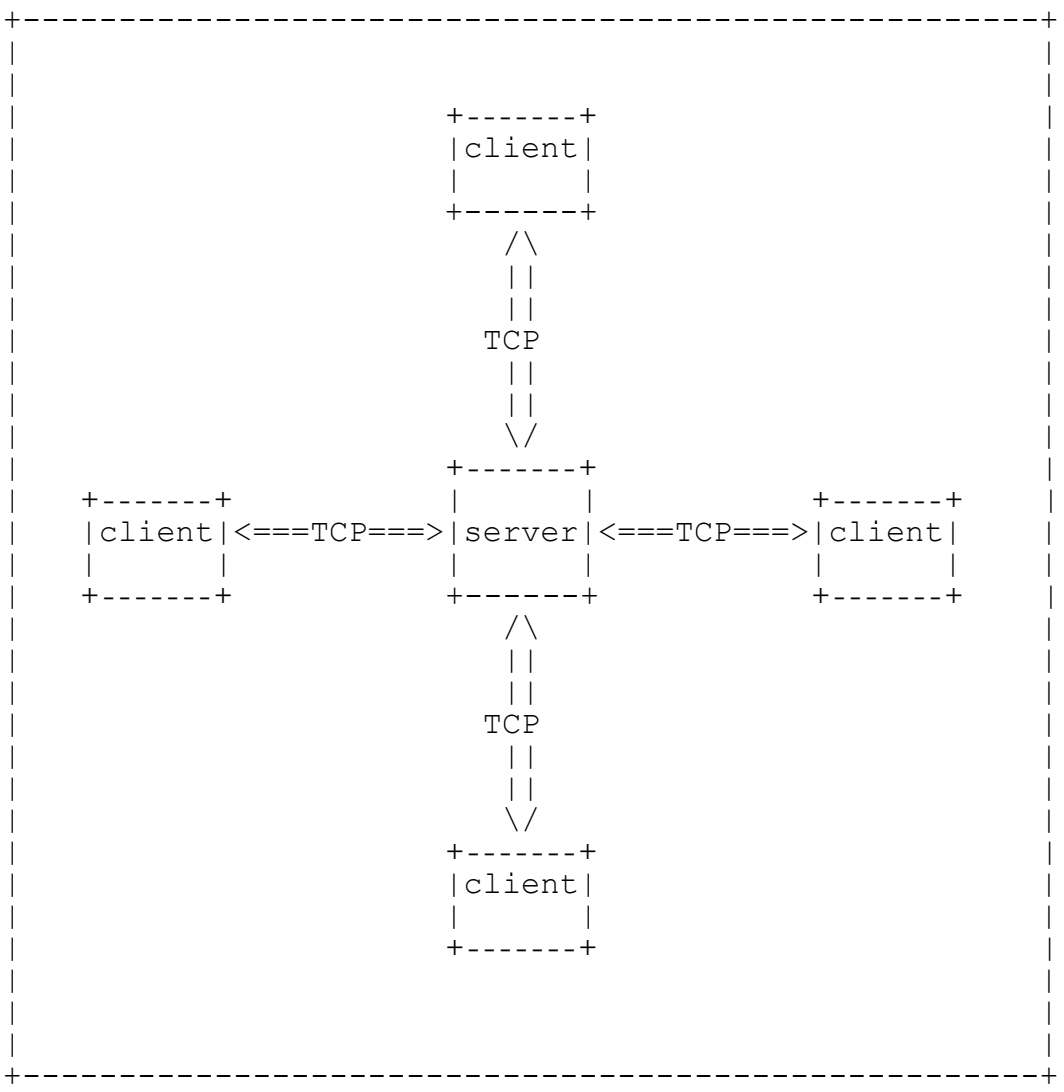

Figure 1 Overview

1.2. Terminology

- User: The people who use the chat room service to communicate with others

- Server: The chat room service provider, who transfers texts between users

- Client: The application running on a host that helps users send and receive texts to/from another client via the server

- Session: The abstract of the two-person chat room, which allows two and only two users to chat with each other

- SessionID: The unique identifier for a session, which should start with an upper-case “S” followed by 5 digits from “0” to “9”

- Account: The identifier for a user, which allows the user being recognized while they use different clients to access the server (Each account has its own userID and password)

- UserID: The unique identifier for each account

- Text: The characters or strings or whatever based on ASCII that the user sends to another user

- Message: ASCII-based information for transferring data between the client and the server, which could be the carrier of texts if applicable

2. Components

2.1. Client

The client is an application running on a host that helps users send and receive texts to/from another user via the server rather than communicate directly with other clients.

2.1.1. Registration

If it is the first time a user uses the service, the client will ask the user to establish an account. The client will gather the userID and the password from the user and send to the server. Userid should starts with a letter, followed by letters or numbers. The length of userID should between 5 and 10. Userid is case-sensitive. Password consists of letters and numbers, whose length should between 6 and 13. Password is case-sensitive. The server will check whether this registration is legal. The server will return a status information depending on the status of the registration.

2.1.2. Signing in

If the user already has an account, they can either sign in with this account or sign up for another one.

2.1.3. Request for Establishing Connections

After the user successfully signs in with their account, the user could request connections to other users. The user should know in advance the userID of whom they want to chat with. Users can exchange their userIDs privately, such as talking with others face to face about their userIDs, sending their userIDs via SMS, and so on. The chat room system is not responsible for showing the mapping between user and their userIDs. If the one who is invited accepts the request, they will connect to each other and start to chat. If not, the client will get the rejection from the server.

2.1.4. Response to invitation from other clients

When a client receives an invitation from other clients forwarded by the server. The users could either accept the invitation or refuse it. If they accept it, then a connection will be built.

2.1.5. Send texts

After the connection is established, the client, then, can send texts to another client via the server, while from users' perspective, they can send texts to other users.

2.1.6. Close a session

Whichever user of a session closes the session, the connection will be killed.

2.2. Server

The server is the core of the online chat room services. It is responsible for maintaining the accounts for users. The server also handles the requests from different clients, and forwards texts between clients as well.

2.2.1. Registration

When the server receives a request for registration, it checks whether the userID is legal and if the userID has already been registered. Then, it responds to the client with the status of the registration.

2.2.2. Signing in

The server handles the requests for signing in from different clients. If the password is correct for the userID, the server will

keep the connection with the client. If not, the server will return an information of failure to the client and close the connection from the client.

2.2.3. Forwarding Requests

When the server receives requests from clients to other clients, it checks if the other client is online. If so, it will forward the invitation to the other client. Then, gets the response from the other side of client. Finally, the server will send a response to both clients letting them know they are connected and know the sessionID.

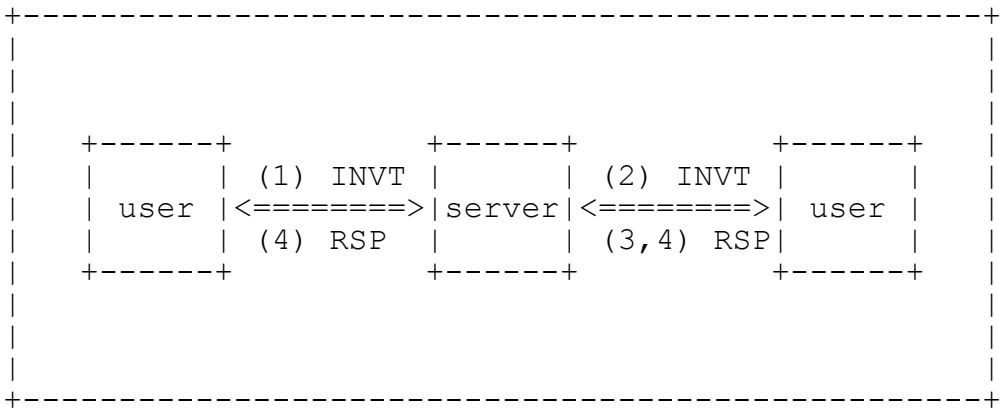

Figure 2 Forwarding Requests

2.2.4. Establishment of Connection between clients

According to the case of 2.2.3, if the other client accepts the invitation, then, the server builds the connection between these two clients, and lets both know they are connected.

3. Protocol

CRP consists of messages from client to server and server to client.

All types of message consist of a start-line, zero or more header fields (aka “headers”), an empty line (i.e., a line with nothing preceding the CRLF) indicating the end of the header fields, and possibly a message-body. The end of message is a line with nothing preceding the ASCII character, “\0”. The total length of a message should not exceed 50 lines.

The message sent by client will be received by server, vice versa. In this document, a XXX message means a message beginning with a start-line of XXX send by client or server.

The start-line can be REG, SIGN, INVT, RSP, TXT, CLOSE and QUIT. The header field should be <Key>=<Value> pair. The Keys can be Status, UserID, Password, SourceUser, TargetUser and SessionID.

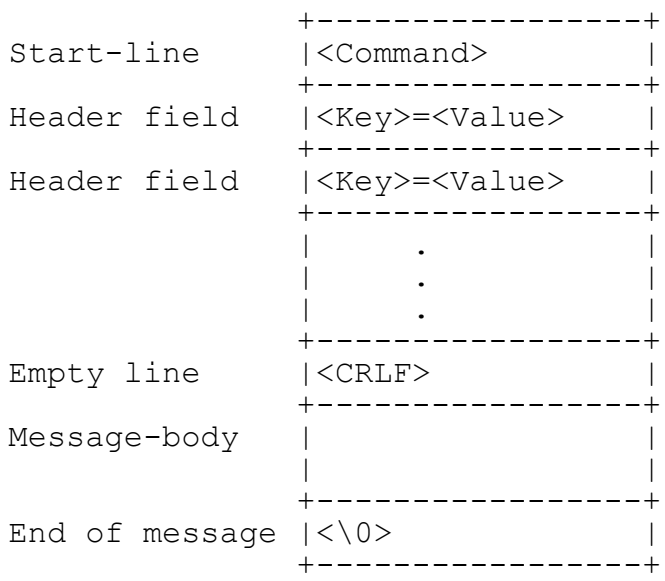

Figure 3 Overview of messages

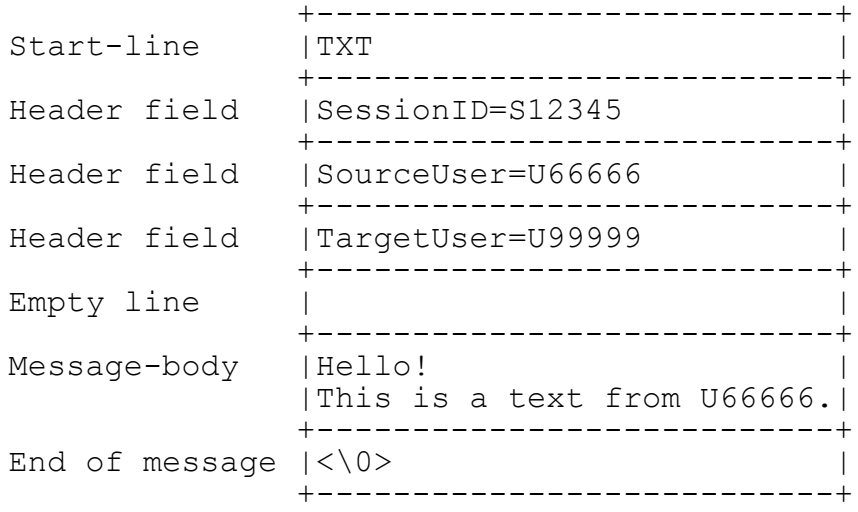

Figure 4 Example of messages

3.1. Messages sent by client

There are 7 kinds of start-line sent by client: REG, SIGN, INVT, RSP, TXT, CLOSE and QUIT.

- REG: REG stands for registration. REG sent by a client suggests the client wants to register an account for the user who is using this client. A REG message should contain a userID and a Password as the header fields. After sending REG messages, the client is expecting a REG message returned by the server, which should contain a header field of Status and userID indicating whether the registration is successful. The Status could either be Successful or Failed.

- SIGN: SIGN stands for sign in. When a user wants to sign in with their account(s), the client will send a SIGN message, which should contain header fields of userID and Password. The client is expecting a SIGN message returned by the server, which should contain a header field of Status. The Status could either be Successful or Failed.

- INVT: INVT stands for the invitation for chatting. When a user wants to request for establishing a connection with another user, the client sends an INVT message including a header field of

sourceUser and a header field of targetUser. A message-body could be included in the INVT message, which would be used as the reason for this request and will finally be transferred to the other user.

- RSP: When a client receives an INVT, the user can either accept it or refuse it. Whatever the user responses, the client will send an RSP message with a header field of sourceUser, targetUser, and Status back to the server. The <Value> of Status can be: Accepted or Refused. The sourceUser should always be the user who initiates the invitation.

- TXT: TXT is the message that contains the text from the user. The header fields consist of the sourceUser, the targetUser, and the sessionID. The TXT message must contain a message-body since a user must send something to another.

- CLOSE: CLOSE stands for the close of a session. If a user ends a certain session, the client will send a CLOSE message to the server. Then, the server will kill the connection between these two users. A CLOSE message should consist of header fields of a sessionID, the sourceUser, and the targetUser.

- QUIT: When a user wants to sign out, the client must send a QUIT message consisting of headers of userID and Password to the server.

3.2. Messages sent by server

There are 6 kinds of start-line sent by server: REG, SIGN, INVT, RSP, TXT, and CLOSE.

- REG: When a client sends a REG, the server should build an account for the user. Then, the server returns a REG including header fields of Status and userID. The <Value> of the Status can either be Successful or Failed.

- SIGN: When a client sends a SIGN, the server should check if the account exists. If so, then, if the Password is correct. Then, the server returns a SIGN message including header fields of userID and Status. If the account doesn’t exist, the <Value> of Status will be NotExist. If the Password is incorrect, then the <Value> would be Failed. If correct, then Successful.

- INVT: When server receives an INVT from a client, the server first checks if the targetUser exists and is online. If so, the server

forwards the INVT to the targetUser. (See 2.2.3. Forwarding Requests)

- RSP: After receiving an INVT from a client, the server will check whether the targetUser exists. If not, the server will instantly send an RSP whose <Value> of Status would be NotExit. If the targetUser exists, the server will forward the INVT to the targetUser and wait for the RSP from the targetUser. If the Status of RSP from the targetUser is Accepted, the server will try to establish a connection between them. If the server successfully builds the connection, it will add one header field of the sessionID to the RSP message, then, sends the modified RSP to both clients so that both will know the status of the connection and the sessionID of the session. If the connection is failed, then the Status would be Failed, and no sessionID will be added to the header.

- TXT: If the session, sourceUser, and targetUser are all correct, the server will directly forward the TXT message to the targetUser. If not, the server just drops the TXT message.

- CLOSE: There are two usages of CLOSE. (1) If the server receives a CLOSE from a client, the server will forward the CLOSE message to the other client then will destroy the session that corresponding to the sessionID mentioned in the CLOSE message. (2) If the server receives a QUIT from a client, the server will send CLOSE to all users that have a session with the quitted client.

- QUIT: While a client tries to login an account which has already been logged in by another clients, the server send a QUIT message to the prior clients informing it there’s another trying to login the account and commanding the first clients to quit.

3.3. Issues concerning TCP connection

The server is always listening on TCP Port 10086, which is used for receiving REG, SIGN, INVT, RSP, TXT, CLOSE, and QUIT. The client establishes one and only one socket connecting to the server once the it starts. All kinds of messages are transferred by this socket.

4. Security Considerations

This section is to inform application developers that the Chat Room Protocol (CRP) does not provide any security guarantee that all messages are not encrypted. The developers should let the users be aware of the risk of their messages being eavesdropped.

5. Informative References

[1] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, DOI 10.17487/RFC0793, September 1981, <https://www.rfc-editor.org/info/rfc793>.

[2] Cerf, V., "ASCII format for network interchange", STD 80, RFC 20, DOI 10.17487/RFC0020, October 1969, <https://www.rfc-editor.org/info/rfc20>.

6. Acknowledgments

This document was prepared using 2-Word-v2.0.template.dot.

Authors' Addresses

Jiuzhi Qi
The George Washington University
Science and Engineering Hall
800 22nd St. NW
Washington, DC 20052

Phone: (202)-710-3950
Email: qijiuzhi@gwu.edu