Asher Bolleddu

Simple Chat Protocol Design Document

a. Project Description
    a. The Simple Chat Protocol (SCP) is designed to facilitate real-time text-based communication between two parties over a network. It provides basic chat functionality, including presence detection, message exchange, and connection management.
b. Protocol Stack Position
    a. SCP operates at the application layer of the OSI model, sitting above either UDP or TCP in the protocol stack. For this design, we'll use TCP as the transport layer protocol to ensure reliable, ordered delivery of messages.

Application Layer: SCP
Transport Layer: TCP
Internet Layer: IP
Link Layer
Physical Layer

c. Header Fields
    a. The SCP header will contain the following fields:
        i. Version (4 bits): Protocol version number for future compatibility.
        ii. Message Type (4 bits): Identifies the type of message (e.g., HELLO, MESSAGE, ACK).
        iii. Sequence Number (16 bits): Used for message ordering and acknowledgment.
        iv. Timestamp (32 bits): Unix timestamp of when the message was sent.
        v. Sender ID (32 bits): Unique identifier for the sending client.
        vi. Recipient ID (32 bits): Unique identifier for the intended recipient.
        vii. Payload Length (16 bits): Length of the message payload in bytes.
d. Message Types
    a. Reciever/Client Side
        i. HELLO: Sent to initiate a connection and check if the other party is online.
        ii. MESSAGE: Contains the actual chat message to be delivered.
        iii. ACK: Acknowledges receipt of a message.
        iv. GOODBYE: Indicates the client is going offline.
    b. Sender/Server Side
        i. HELLO_ACK: Acknowledges a HELLO message and confirms the server is online.
        ii. MESSAGE_ACK: Confirms successful delivery of a MESSAGE.
        iii. PING: Periodically sent to check if the client is still connected.
        iv. GOODBYE_ACK: Acknowledges a client's GOODBYE message.
e. Message Handling
    a. HELLO
        i. Client: Sends to initiate connection or check server status.
        ii. Server: Responds with HELLO_ACK if online.
    b. MESSAGE

i. Sender: Constructs message with header and payload, sends to recipient.
ii. Receiver: Processes message, displays to user, sends MESSAGE_ACK.

c. ACK
    i. Receiver: Sends to confirm receipt of any message type.
    ii. Sender: Updates message status, may retransmit if ACK not received.
d. GOODBYE
    i. Client: Sends when intentionally disconnecting.
    ii. Server: Acknowledges with GOODBYE_ACK, closes connection.
e. PING
    i. Server: Periodically sends to verify client connection.
    ii. Client: Responds with ACK to confirm active status.