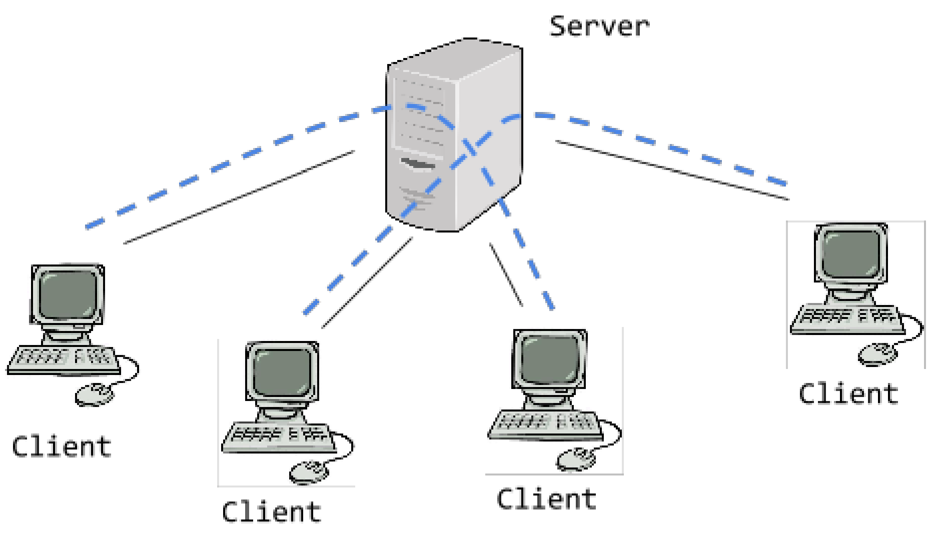
Computer Networks

Server Based Chat

In this project, you will need to design a server and client program for a chat application. The server's purpose will be to mediate chat sessions between clients. There can be multiple clients engaged in chat sessions simultaneously. However you can assume a given client at any particular time will have a chat session with at most one other client.
You may propose to write your code in a language of your choice and submit it for approval.

1. System Overview

In what follows, "client" refers to a machine, while "end user" is the human user of that machine. The server is assumed to be up and running. Each client who is a subscriber to the chat service, gets assigned a unique Client-ID, along with a secret key K at subscription time. The unique Client-ID and key K are permanently stored in the client, and the list of subscriber Client-IDs, along with the corresponding secret keys, is permanently stored at the server. Client-ID could be a string, or a numerical ID (Choose one, but do not mix them in your implementation).



2. Connection to the Server

A client needs to connect first to the server. Connection to the server is initiated when a user logs on.

1. When the user at client A logs on, client A sends a HELLO (Client-ID-A) message to the server, using UDP transport.

2. The server verifies that Client-ID-A is an ID on the list of subscribers. If so, the server retrieves the client's secret key and sends a CHALLENGE (rand) message to the client, using UDP.
3. The client responds with a RESPONSE (Res) to authenticate itself. If authentication is not successful, the server sends an AUTH_FAIL message to the client. Else the server generates an encryption key CK-A, then sends an AUTH_SUCCESS(rand_cookie, port_number) message to the client. The message is encrypted by CK-A.
4. The client generates the same CK-A key, and decrypts the message. From this point on, all data exchanged between client A and the server is encrypted using CK-A.
5. Client A establishes a TCP connection to the port at port_number and sends a CONNECT (rand_cookie) to the server. From this point on, until the TCP connection is closed, all data (signaling messages and chat) is exchanged over the TCP connection.
6. The server sends CONNECTED to the client. The client is connected.
7. The client tears down the TCP connection when the user types "Log off" or when the activity timer expires

3. Client A Initiates Chat Session to Client B

This scenario will go through the following steps. Client A must have already gone through the connection phase and be connected to the server.

1. To start a chat with client B, the end user types "Chat Client-ID-B", and client A sends a CHAT_REQUEST (Client-ID-B). If the server determines client-B is connected and not already engaged in another chat session, the server sends CHAT_STARTED(session-ID, Client-ID-B) to client A, and CHAT_STARTED(session-ID, Client-ID-A) to client B. Client A and Client B are now engaged in a chat session and can send chat messages with each other, through the server. The clients display "Chat started" to the end user at A and B. If client B is not available, the server sends UNREACHABLE (Client-ID-B) to client A.
2. When the end user at A or B wants to terminate the session, the end user types "End Chat", and the associated Client sends END_REQUEST (session-ID) to the server.
3. The server sends an END_NOTIF(session-ID) to the other client. The Clients display "Chat ended" to their respective end users.

4. Security

a. Authentication

The server is assumed secure, and only the client needs to be authenticated. Authentication is based on the challenge/response mechanism, used in cellular networks.
Upon receiving HELLO (Client-ID-A), the server looks up Client A's secret key K_A. Then it generates a random number rand. Rand and K_A are input into an authentication algorithm A3. The output, XRES, is stored by the server. Then the server sends rand in the CHALLENGE (rand). Client A is expected to run the same algorithm A3 with the same inputs rand and K_A, and produce RES, which is sent to the server in the RESPONSE (Res) message. If RES matches XRES, the server has authenticated the client. For this assignment, use a hash function for algorithm A3: RES = hash1(rand + K_A), where + denotes concatenation. Examples are MD5, SHA1, SHA256 etc.

b. Encryption

Concurrently with the above authentication process, the server and client run an A8 algorithm, which takes as input rand and K_A, to generate a ciphering key CK_A. For this assignment, use a hash function for A8: CK_A = hash2(rand + K_A).

c. Integrity protection

For simplicity, integrity protection is not implemented.

5. Chat History

In addition, the server will maintain chat history of each client. At any time in the connected state, client A can see the history of past chat messages with client B by sending the HISTORY (Client-ID-B) message to the server. In response the server sends all the past chat messages exchanged between client A and client B. Client A should display all the chat messages in the history in the following format
<session_id> <from: sending client> <chat message>

6. Protocol Messages

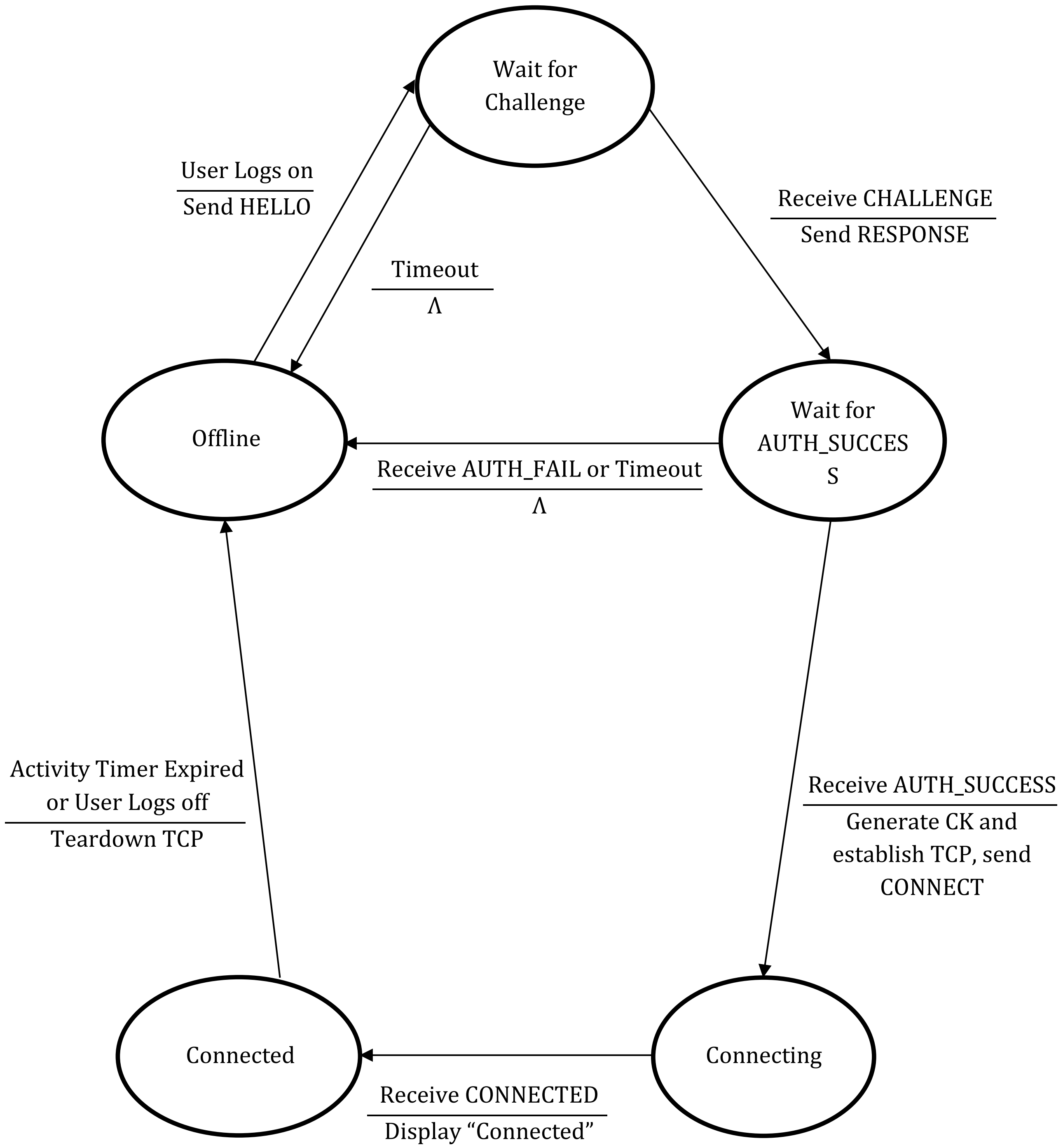
The significant fields of the messages along with their semantics are defined below. As part of the project assignment, you have to define the precise syntax and format of the messages.

- HELLO (Client-ID-A): Initiates the process for Client A to be authenticated and registered with the server.
- CHALLENGE (rand): Sent by the server to challenge the client to authenticate itself. rand is a random number generated by the server. A new rand is generated at every CHALLENGE.
- RESPONSE (client-ID, Res): Response to the challenge, sent by the client to authenticate itself.
- AUTH_SUCCESS(rand_cookie, port_number): Sent by the server to notify the client authentication is successful. rand_cookie is a random number generated by the server, and port_number is a TCP port number assigned by the server for subsequent connection by the client.
- AUTH_FAIL: Sent by the server to notify the client authentication has failed
- CONNECT (rand_cookie): Sent by the client to the server. rand_cookie is the cookie previously sent by the server.
- CONNECTED: Sent by the server to notify the client it has been connected
- CHAT_REQUEST (Client-ID-B): Sent by client A to the server to request a chat session with Client B
- CHAT_STARTED (session-ID, Client-ID-B): Sent by the server to notify client A that a chat session with client B has started. Session-ID is an ID assigned to the session
- UNREACHABLE (Client-ID-B): Sent by the server to client A to notify client B is not available for chat.
- END_REQUEST (session-ID): Sent by any of the clients involved in the chat session to request a termination of the session

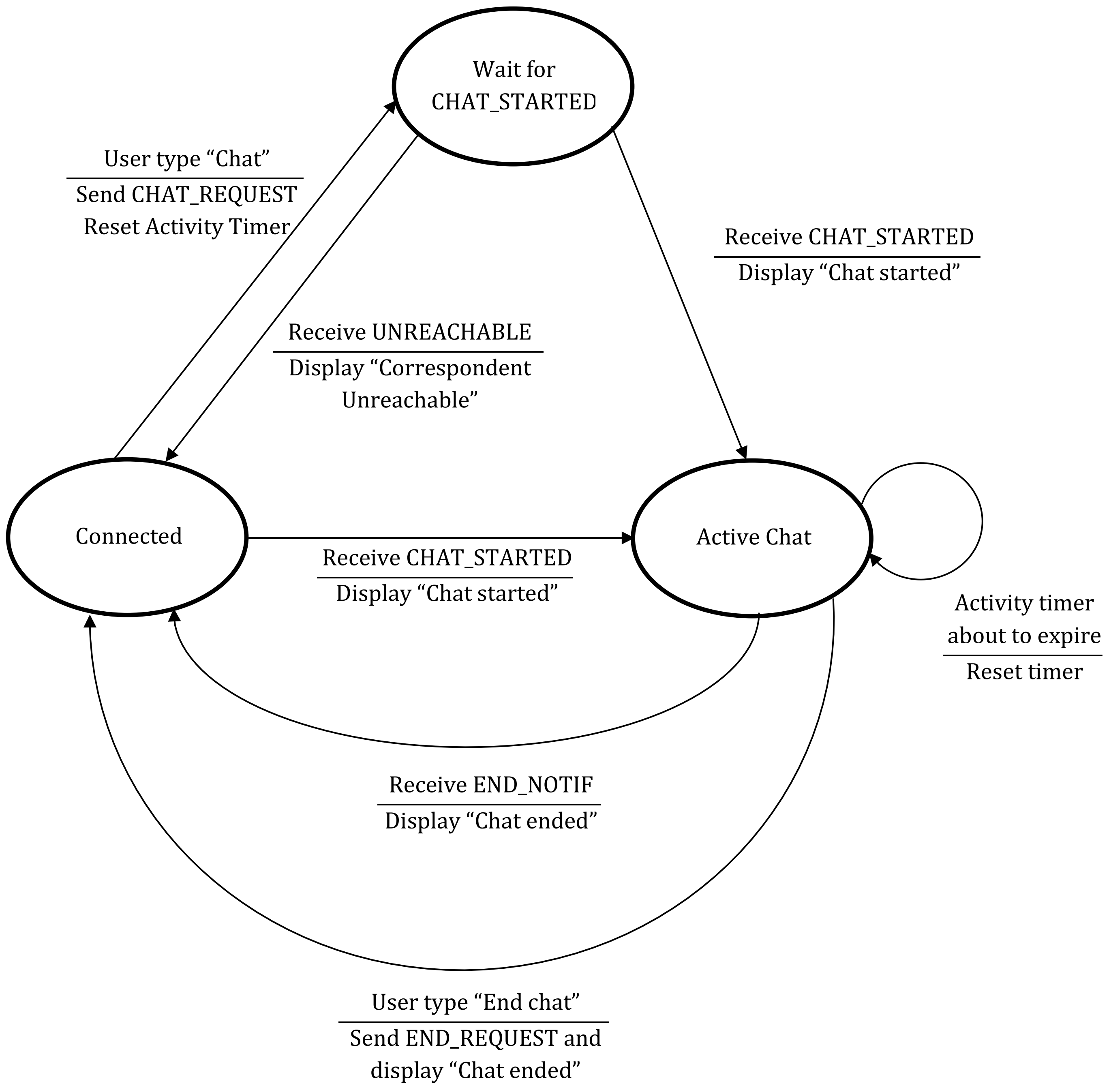
- END_NOTIF (session-ID): Sent by the server to notify a client involved in the session that the session has been terminated by another client.
- CHAT (session-ID, chat message): Exchanged between the clients, relayed by the server. Carries the actual chat message
- HISTORY_REQ (Client-ID-B): Sent by client A to request the history of past chat messages with client B.
- HISTORY_RESP (Sending Client-ID, chat message): Sent by the server to the client who requested the history. Sending Client-ID is the ID of the client who sent the chat message, and chat message is the chat message in the history. There is one HISTORY_RESP message for each chat message in the history.

7. State Diagram (Client)

CONNECTION PHASE:



CHAT PHASE:

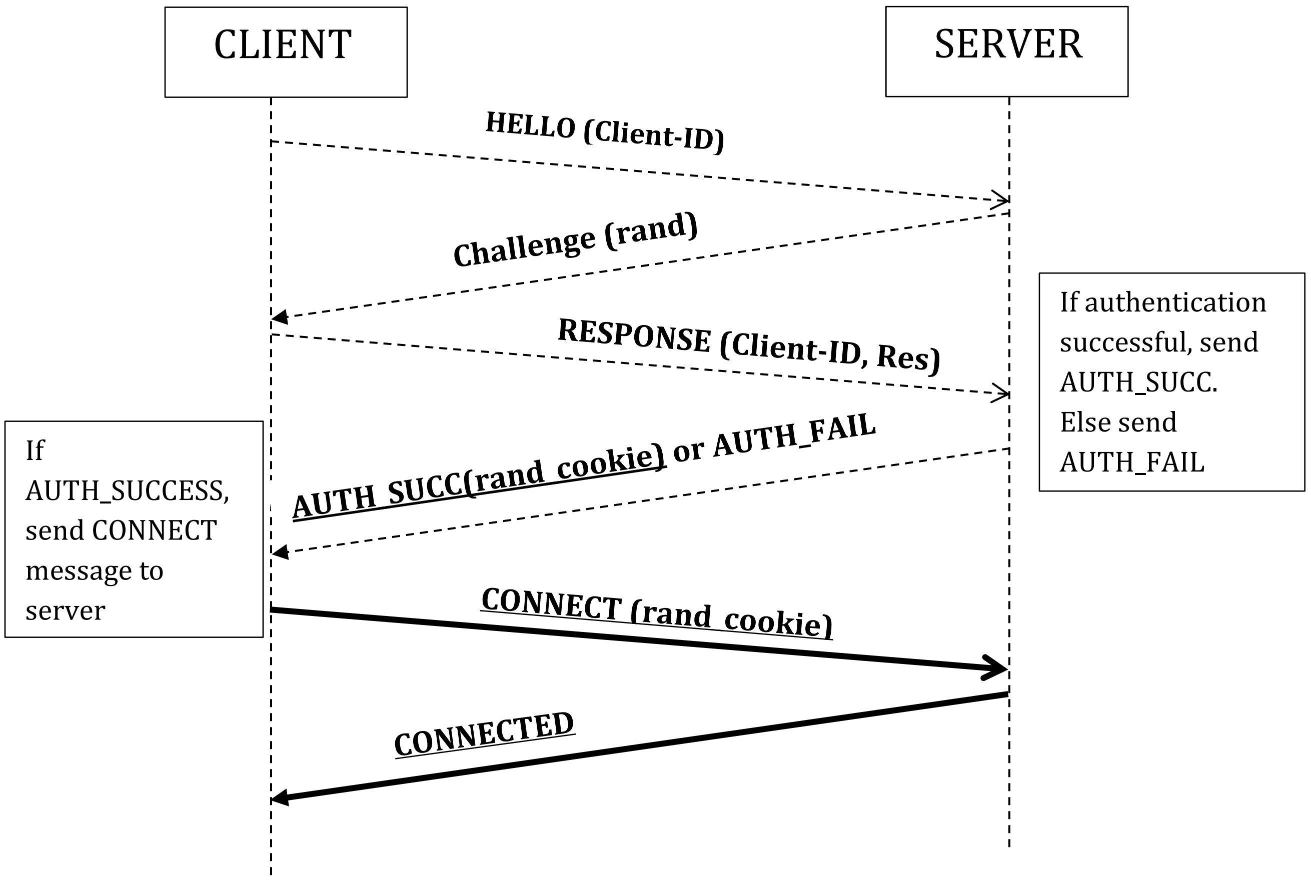


Wait for CHAT_STARTED
Connected
Active Chat
User type "Chat" / Send CHAT_REQUEST Reset Activity Timer
Receive UNREACHABLE / Display "Correspondent Unreachable"
Receive CHAT_STARTED / Display "Chat started"
Receive CHAT_STARTED / Display "Chat started"
Activity timer about to expire / Reset timer
Receive END_NOTIF / Display "Chat ended"
User type "End chat" / Send END_REQUEST and display "Chat ended"

8. State Diagram (Server)

As part of the assignment, you have to develop the state diagrams for the server: Connection phase and chat phase

9. Connection Phase


CLIENT
SERVER
HELLO (Client-ID)
Challenge (rand)
RESPONSE (Client-ID, Res)
If authentication successful, send AUTH_SUCC. Else send AUTH_FAIL
AUTH_SUCC(rand_cookie) or AUTH_FAIL
If AUTH_SUCCESS, send CONNECT message to server
CONNECT (rand_cookie)
CONNECTED

Dashed lines indicate UDP transfer, solid lines indicate TCP transfer, and underlined messages are encrypted.

10. Validation Scenarios

In order to validate your code, use the following scenarios.

a. Basic chat initiated and closed by A

Step	Action	Expected behavior
0	Start from the state where clients A and B are off line	

1	Client A’s user types “Log on”	Client A and server should go through the connection phase and exchange the messages shown in “Connection Phase” Client A should display “Connected”
2	Client B’s user types “Log on”	Client B and server should go through the connection phase and exchange the messages shown in “Connection Phase” Client B should display “Connected”
3	Client A’s user types “Chat Client-ID-B”	Client A should send CHAT_REQUEST, server should send CHAT_STARTED to client B and client A. “Chat started” should be displayed at both clients
4	Client A and client B exchange chat messages	What client A’s user types should be displayed at client B, and vice versa
5	Client A’s user types “End chat”	Client A should send END_REQUEST to server and display “Chat ended” Client B should receive END_NOTIF from server and display “Chat ended”
6	Client A’s user types “Log off"	Client A should close the TCP connection

b. Basic chat initiated by A and closed by B

Step	Action	Expected behavior
0	Go through steps 0 to 4 of “Basic chat initiated and closed by A”	
1	Client B’s user types “End chat”	Client B should send END_REQUEST to server and display “Chat ended” Client A should receive END_NOTIF from server and display “Chat ended”
2	Client A’s user types “Log off"	Client A should close the TCP connection

c. Basic chat initiated by A, but B initially not connected

Step	Action	Expected behavior
0	Go through steps 0 to 1 of “Basic chat initiated and closed by A”	
1	Client A’s user types “Chat Client-ID-B”	Client A should send CHAT_REQUEST, server should reply with UNREACHABLE, client A should display “Correspondent unreachable”
2	Client B’s user types “Log on”	Client B and server should go through the connection phase and exchange the messages shown in “Connection Phase” Client B should display “Connected”

3	Client A's user types "Chat Client-ID-B"	Client A should send CHAT_REQUEST, server should send CHAT_STARTED to client B and client A. "Chat started" should be displayed at both clients
4	Client A and client B exchange chat messages	What client A's user types should be displayed at client B, and vice versa
5	Client B's user types "End chat"	Client B should send END_REQUEST to server and display "Chat ended" Client A should receive END_NOTIF from server and display "Chat ended"
6	Client A's user types "Log off"	Client A should close the TCP connection

d. Basic chat initiated by C, but B is already in another chat

Step	Action	Expected behavior
0	Go through steps 0 to 4 of "Basic chat initiated and closed by A"	
1	Client C's user types "Log on"	Client C and server should go through the connection phase and exchange the messages shown in "Connection Phase" Client C should display "Connected"
2	Client C's user types "Chat Client-ID-B"	Client C should send CHAT_REQUEST, server should reply with UNREACHABLE, client C should display "Correspondent unreachable"

e. History recall

Step	Action	Expected behavior
0	Go through the steps of "Basic chat initiated and closed by A"	
1	Client B's user types "Log on"	Client B and server should go through the connection phase and exchange the messages shown in "Connection Phase" Client B should display "Connected"
2	Client A's user types "History Client-ID-B"	Client A should send HISTORY_REQ and should receive all the chat messages of the last chat session with B, sent in one or more HISTORY_RESP messages

f. Simultaneous chat sessions

Verify you can establish 5 simultaneous chat sessions between 5 pairs of clients.

11. What to Turn In

1. A proposed action plan by the due date specified in “ProjectTimeline”.
2. A team report by by the due date specified in “ProjectTimeline”.
   a) Containing source codes related to client and server. Include a README file that describes how to compile/run the program.
   b) Providing a background summary of challenge/response authentication scheme
   c) Describing your hardware setup and configuration
   d) Including the design document of your code (i. e Protocol State Diagrams, Sequence Diagram or any other document that helps to understand what has been done)
   e) Providing screenshots of chat sessions
   f) Describing what issues, if any, the team encountered during the project, how the team overcame the issues and what the team learned from the project. You can also provide suggestions on how the projects in Computer Networks could be improved in the future.
   g) Including a video clip to demo the code running
3. Individual reports, one for each team member by the due date specified in “ProjectTimeline”. The individual report is confidential and not shared with the other team members.
   a) If you, as an individual team member, have anything specific to add to 1.f) in the team report, please do it in your individual report. Describe what issues, if any, you, as an individual team member, encountered during the project, how you overcame the issues and what you learned from the project (this is not necessarily just about the topic, could be related to teamwork, etc.). You can also provide suggestions on how the projects in Computer Networks could be improved in the future. This complements the team report with any individual viewpoint not included in the team report.
   b) Describing what each team member (including yourself) did and contributed to the project, and assign a numerical score from 1 to 10 to each team member, including yourself. 1 is the poorest, and 10 is the best.

Note: There will be a separate session (taking place outside of lecture hours) for you to demo your code is running. In addition, you will be asked to explain your code. The code demo sessions will take place towards the end of the semester.