Chat Application Project

Project description:

In this project, we developed a chat application that allows the communication between different clients using UDP, each client connects to the centralized server using TCP.

Since the communication between clients s using UDP, messages might be disordered, we manage to solve this problem by creating a different sequence for each message of the same client.

To Improve the security, we encrypted the messages and we add signature to ensure that the message is received from the right client.

Specifications:

At startup:

Server side :

- Initiate connection
- Wait for clients to connect
- Build a list of connected clients

Client side :

- Connect to the server
- Receives the list of connected clients

At steady state:

Server side :

Client side :

- Clients start chatting with each other

  Following the protocol

At change:

Server side :

- When a client connects or disconnects
- Send notification to clients

Client side :

- Clients start chatting with each others

  Following the protocol

Work1:

Project structure:

Controllers:

- **Server**: The main task of the server is to accept connections from clients and then create a thread (serverThread) for each client, and sends the updated list of connected clients.
- **Client**: Sends disconnection request to server and create the SendThread and ListenThread to handle communication between clients.
- *ServerThread*: This thread is created for each client; its job is to receive disconnection requests from clients and notify others about the new list.
- *ListenThreads*: Receives messages from other clients either they were unicast messages or Broadcast ones.
- *SendThreads*: Sends unicast and broadcast messages.
- *UpdateListChatters*: Update the list of chatters if a new client joined or quit the chat.

Modules :

- *Chatter*: Provides the state f the user and it is used by the Client class to act on its behavior.

Protocols :

- *Global Constants* : States global constants.
- *Message*: States the different forms of message used in communication either between clients or between clients and server.
- *Protocol*: represents the protocol followed in this work, it describes the different messages and functions used for communication also the errors managed.

Work2:

In order to create disordered messages, we added a new function delay in “sendThread” to send messages in a random delay.
to deal with the disordered messages, we identify each message by a sequence number related to each client.
At the reception side, we check for each client the sequence of the received message, and we verify if it is the expected sequence, if so, we print it, else we store the messages until we get the right sequence.

Work3:

To ensure the security of this application we used two methods:

- *Encryption*: To verify the message has not changed, each client creates two keys for communication (public key and private key), the sender encrypts the message with the public key of the receiver who will decrypt it with his private key.

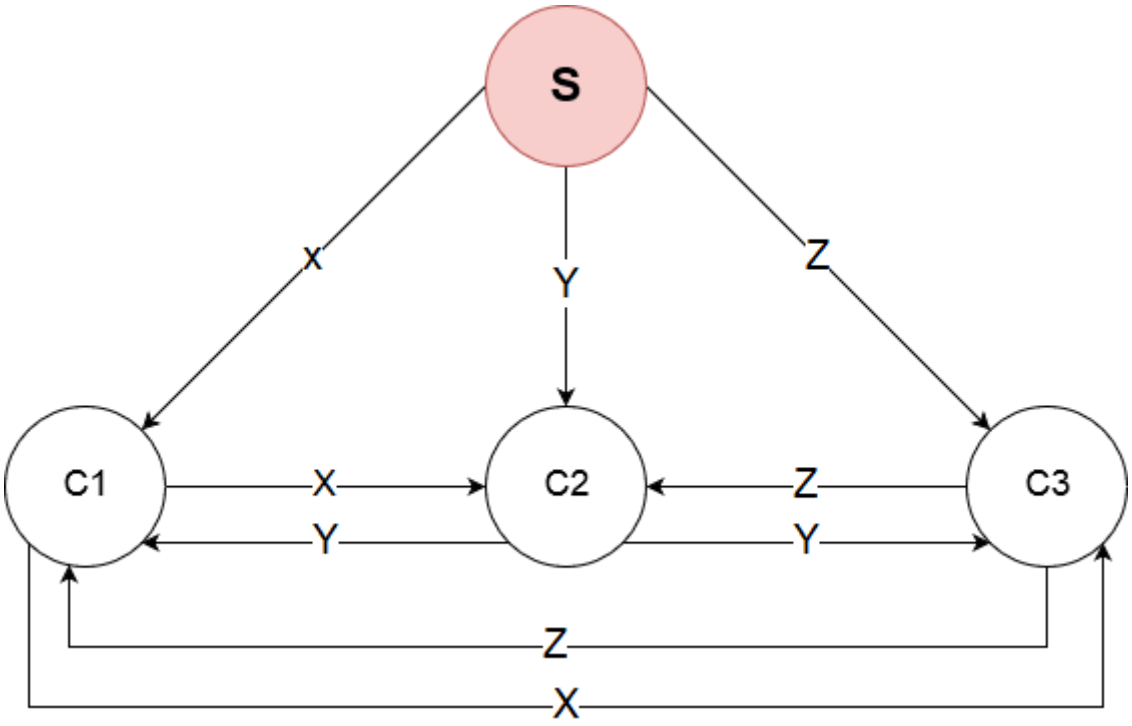
- *Signature*: To ensure that a message is received from the right client, the signature is added to the encrypted message (with the private key of the sender), the receiver will verify the received signed message using the public key of the sender, and decide if it is a malicious client or not.

Work4:

Q1- Let's consider that the client S who broadcasts the message m is designed as a "leader". The specification of the communication between S and correct clients is defined as follows:

*Termination: the algorithm must guarantee that all correct clients reach a decision regarding the value of the message sent by the leader S .
* Agreement : All correct clients have to decide on the same value of the message that the leader sent .
*Validity: If the Leader S is a correct client,all the other clients have to decide on the same value that was originally given by the leader S .

Q2- Let's n be the number of clients.
We consider that n>3
As stated in the The Lamport, Pease and Shostak Algorithm ,the clients arrive to a consensus if the number of clients n is superior to 3*k where k is the number of liars. n>3*k
in the case of 1 liar ,the clients can make a consensus if their number is a least 4 .
The algorithm that solves the problem is then :
1/ The Leader S sends his message to each client.
2/For each i, let vi be the value client i receives from the leader S . The client i acts as the leader and sends the value vi to each of the n-2 other remaining clients(except the leader S).
3/ For each i, and each j≠i, let vi be the value client i received from client j . Client i uses the value majority (v1, v2, … vn) to make a decision .
You can find below an illustrated implementation of the algorithm in the case we have 4 nodes and one liar :
1/if the source S is the liar:



in the figure above:

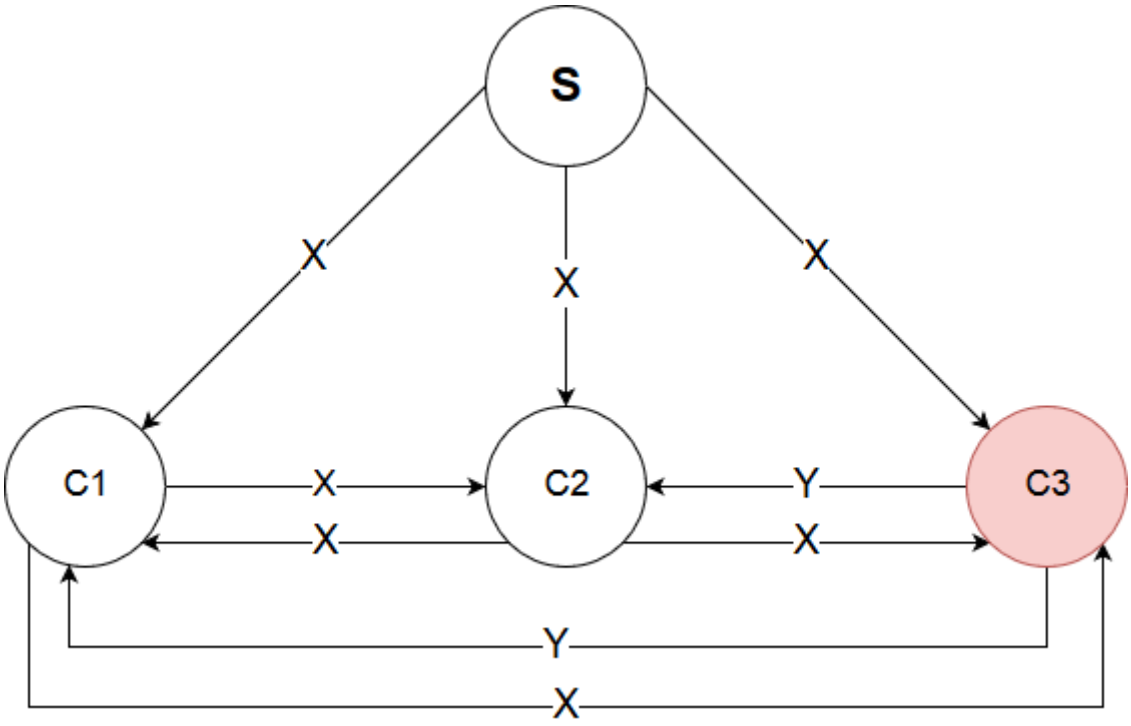
-Leader S sends x, y, z to C1, C2, C3 respectively

-C1 sends x to C2, C3 | C2 sends y to C1, C3 | C3 sends z to C1, C2

-C1 ← majority(x,y,z) | C2 ← majority(x,y,z) | C3 ← majority(x,y,z)

-All the clients have the same value .The consensus is reached. Even if x, y, z are all different, the value of majority(x, y, z) is the same for all 3 Clients(agreement). In the case x,y,z are totally different messages we can assume that the nodes will decide that the leader S is a liar.

2/if the client C3 is the liar:



- Leader S sends x to all the clients
-C1 sends x to C2 | C3 sends y to C2…..

1. -C2 ← majority(x,x,y)=x
2. -The final decision is the majority vote from C1, C2, C3 which is equal to x .The consensus has been achieved.

Q3. If there are k liars. The algorithm that can be used to solve the problem is The Lamport, Pease and Shostak Algorithm .

Following this algorithm,the clients arrive to a consensus if the number of clients n is superior to 3*k where k is the number of liars. (n>3*k)

Depending on the value of k,the algorithm OM can be described as follows:

Algorithm OM(0) :

1. The leader sends his message to every client.
2. Each client uses the value he receives from the leader S

Algorithm OM(m), m > 0

1. The Leader sends his value to each client.
2. For each i, let vi be the value client i receives from the leader. Client i acts as the leader in Algorithm OM(m-1) to send the value vi to each of the n-2 other client.
3. For each i, and each j≠i, let vi be the value client i received from client j in step 2 (using Algorithm (m-1)). Client i uses the value majority (v1, v2, … vn) to take the decision.

Q4. By using the cryptography technics(signature of the source S) we guaranteed that all the other clients except the source S can not be liars.the problem is then reduced to only <one eventual liar(the source S).

The answer to the previous questions will not change but we will use the same algorithm cited above in the case of one liar.