Thejasvi Konduru(20171134)
Shreya Terupally(20171175)
Anurag Sahu(2018121004)

DS_Project Report

This is our project video link:-
https://drive.google.com/file/d/1pWipkr4znsu1i9z5lisKCiR8AOhY7JZg/view?usp=sharing

- **Chat application**
  - A chat application is designed using RMI(Remote method Invocation) in Java to simulate FIFO and Non-FIFO channels.
  - **Implementation**
    - A new thread is created for every new client.Every client is asked to enter username and password and if he is a new client then this information i.e username and password are stored otherwise if the client already exists then we check if the password entered by the client is correct or not using the stored information and if the password is incorrect then an error is returned saying that the password is incorrect.
    - The mode of channel is predefined for every pair of clients and the no.of messages client wants to send is taken as an input.The channel can be either FIFO or Arbitrary(Non-FIFO).

- **FIFO implementation**
  - All the required messages are stored in an array and then sent in the same order to the destination client.
- **Arbitrary(Non-FIFO) implementation**
  - The way in which we have implemented Non-FIFO is that every message is executed asynchronously after a time delay of random time in the range of 0 to 10 seconds.
- Clients can also broadcast a message i.e a message can be sent to all the clients at once.

- **Ricart Agrawala Algorithm Implementation**
  - **Requesting Site**
    - It sends a broadcast message and this message includes the clientname and the timestamp of the request and a variable using which the receiving site can determine if the message received is a request for the critical section.
  - **Receiving Site**
    - If it is neither executing nor waiting for CS then it sends a reply message to the requesting site.
    - If it is waiting and if it’s timestamp is greater than the timestamp of the requesting site then it sends the reply message.
    - If it is currently executing CS or if it is waiting for CS and if it’s timestamp is lesser than the

timestamp of the requesting site then it defers the reply message.

- **Impact of channels on Ricart agrawala algorithm**
  - **Fairness:**
    - Fairness is not affected by the type of channel(FIFO or Non-FIFO).Lets take two cases to understand this:-
    - **<u>case:-1(Non-FIFO case)</u>**
    - If we send the reply before the request, but the request reaches before the reply.
    - Let A,B,C denote the events and
    - A=request from the clientA
    - B=reply to the clientA
    - C=request from the clientB
    - We know that
      - timestamp(A)<timestamp(B)
      - timestamp(B)<timestamp(C)
      - From the above two we can tell timestamp(A)<timestamp(C)
    - As timestamp(A)<timestamp(C) clientA will not send the reply message to clientA and when it receives the reply message from clientB and if this is the last reply message clientA is waiting for then it starts executing otherwise it waits until it receives (N-1) reply messages.

        - **<u>case:-2(fifo case)</u>**
        - If we send reply before request and also reply reaches before request then if this is the last reply message receiving site is waiting for then it starts executing CS.Suppose if it is waiting for the reply messages from the other clients then on receiving the request from client B it defers the reply as it's waiting for CS and the timestamp of the received request is greater than its own request's timestamp.
    - **Deadlock**
        - Deadlock is not possible.Lets say we have three clients i.e clientA,clientB,clientC and without loss of generality assume that t(A)>t(B)>t(C) where t denotes timestamp and assume there is a cycle where clientA is waiting for clientB and clientB is waiting for clientC and clientC is waiting for clientA.This cycle is not possible because A will send a reply message to C as t(A)>t(C) so C will start executing CS and then B executes CS as t(A)>t(B) and then A starts executing after B finishes.
    - **Complexity of the algorithm**
        - **Message Complexity** is 2(N-1) per client per CS execution.
        - **Reason:**-A client requires (N-1)REQUEST messages and (N − 1) REPLY messages per CS

execution. Thus, it requires 2(N − 1) messages per CS execution.

- **Node Failures and channel failures**
  - If there is a node failure before sending the reply message then the no.of reply messages received is not equal to the N-1 then the requesting site keeps on waiting for the reply and it will never execute CS.
  - If there is a node failure after sending the reply message then there is no problem.
  - If a channel fails i.e the messages sent by the nodes are lost or the messages are changed then the requesting site keeps on waiting for the reply and it will never execute CS.
  - So the Ricart Agrawala algorithm does not work when there are node failures and channel failures.