CS751: Project Runthrough

An Optimal Algorithm for Mutual Exclusion in Computer Networks

Siddharth Saha 170100025
Saranya C 184057001

1 Introduction:

We aim to implement the algorithm of mutual exclusion in the paper **An Optimal Algorithm for Mutual Exclusion in Computer Networks** authored by Glenn Ricart and Ashok Agrawala.

This algorithm, popularly known as the Ricart-Agrawala algorithm, provides an optimal solution to the problem of mutually exclusive access to a critical section in computer networks, whose nodes communicate only by messages and do not share memory.

2 Overview of the Algorithm:

This Algorithm enables to achieve Mutual Exclusion among Nodes which operate in a distributed environment with no shared memory. The entire communication amongst the Nodes is based on Network Messages.

Whenever a Node wants to enter the Critical Section, it sends a REQUEST message to all the other Nodes. On receiving the REPLY, the Node is entitled to enter the Critical Section.

The steps involved are briefed below:

- Node which wants to enter the Critical Section sends REQUEST to all the other Nodes. The REQUEST consists of its Node ID and a Request Number.
- The Other Nodes, on receiving decides whether it shall allow the Request Sender Node to enter the Critical Section or not based on information contained in the REQUEST i.e Request Number and the Node ID.
- There is no conflict of interest when the receiver Node does not want to enter the Critical Section. When the receiver itself wants to enter Critical Section priority based decision is taken using the Request Number and the Node ID information. Thus the request shall be deferred or replied immediately based on the decision. Deferred requests are processed after the Node exits from the Critical Section. In case of conflict of interest the decision is based on the flowchart presented in Figure-1.
- The Request Sender waits till it receives REPLY from all the other Nodes. It is not aware of the deferment of its request, if any. In case of deferment, the REPLY message is only sent when the conflict of interest is resolved. Once the REPLY messages are received from all the other Nodes, the Request Sender shall enter the Critical Section.

The Algorithm is deadlock free and starvation free. Also the same is achieved with optimum number of messages which is 2(N-1) for each Critical Section entry. The important assumption made here is the faultless Network communication.

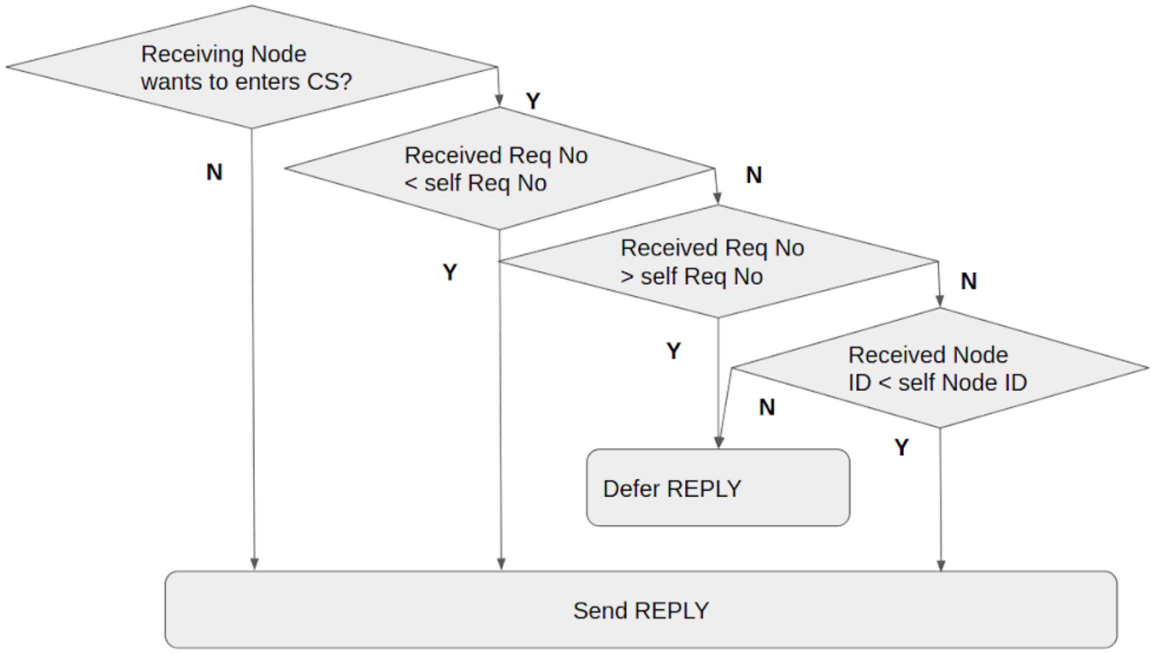
Receiving Node wants to enters CS?
Y
N
Received Req No < self Req No
N
Y
Received Req No > self Req No
N
Y
Received Node ID < self Node ID
N
Y
Defer REPLY
Send REPLY

Figure 1: Flowchart depicting the decision process for sending REPLY message

3 Implementation Aspects:

- There are N Nodes in the system, numbered from zero to N-1. Each node executes on a different machine. The Nodes communicate with each other using TCP socket communication.
- Each Node is designed with a Server and Client Component. On start up, the Node starts listening using its Server component to receive messages from other Nodes. In order to receive messages in concurrent manner, Server components are implemented in N-1 threads.
- When a Node wants to enter its Critical Section it uses the Client component to connect to each of the other Nodes and sends the REQUEST message. Once the REPLY is received from all the other Nodes, it does the Critical Section processing. On exiting from the Critical Section the deferred requests, if any are processed.
- The diagrammatic representation of the transfer of messages among the three nodes is shown in Figure-2. For the sake of demonstration each Node requests for entering the Critical Section for a predefined number of times. Once a Node enters the Critical Section it exists from there after a predefined time duration.

4 Implementation Details:

- **Language:** Java
- **Key Areas of implementation:** Socket programming, Thread programming, Synchronised functions

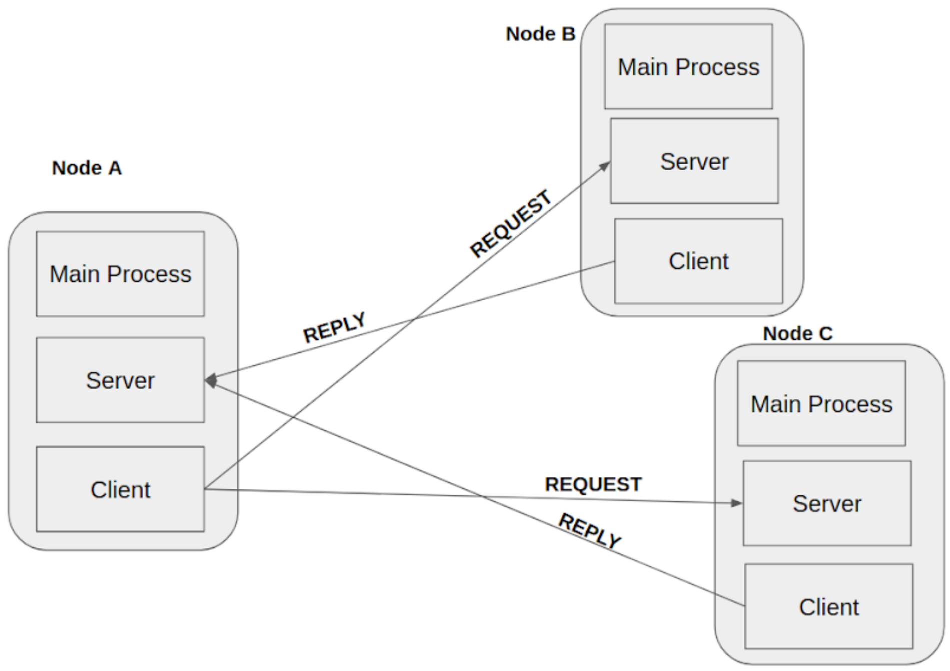

Figure 2: Transfer of REQUEST and REPLY Messages when Node A wants to enter CS

5 Concluding Remarks:

The Algorithm provides efficient deadlock free and starvation free mutual exclusion processing among N nodes in distributed environment. In this implementation we have demonstrated the same. Handling message losses during Network failure and dynamically adding or removing Nodes are some of the extensions which are to be addressed for real life implementations.