CS 751 Course Project Presentation

Implementation & Demonstration of An Optimal Algorithm for Mutual Exclusion in Computer Networks

Siddharth Saha (170100025)
Saranya C (184057001)

Objective

- To implement Mutual Exclusion Algorithm for Distributed environment.
- **Reference Paper**: An Optimal Algorithm for Mutual Exclusion in Computer Networks authored by Glenn Ricart and Ashok Agrawala.

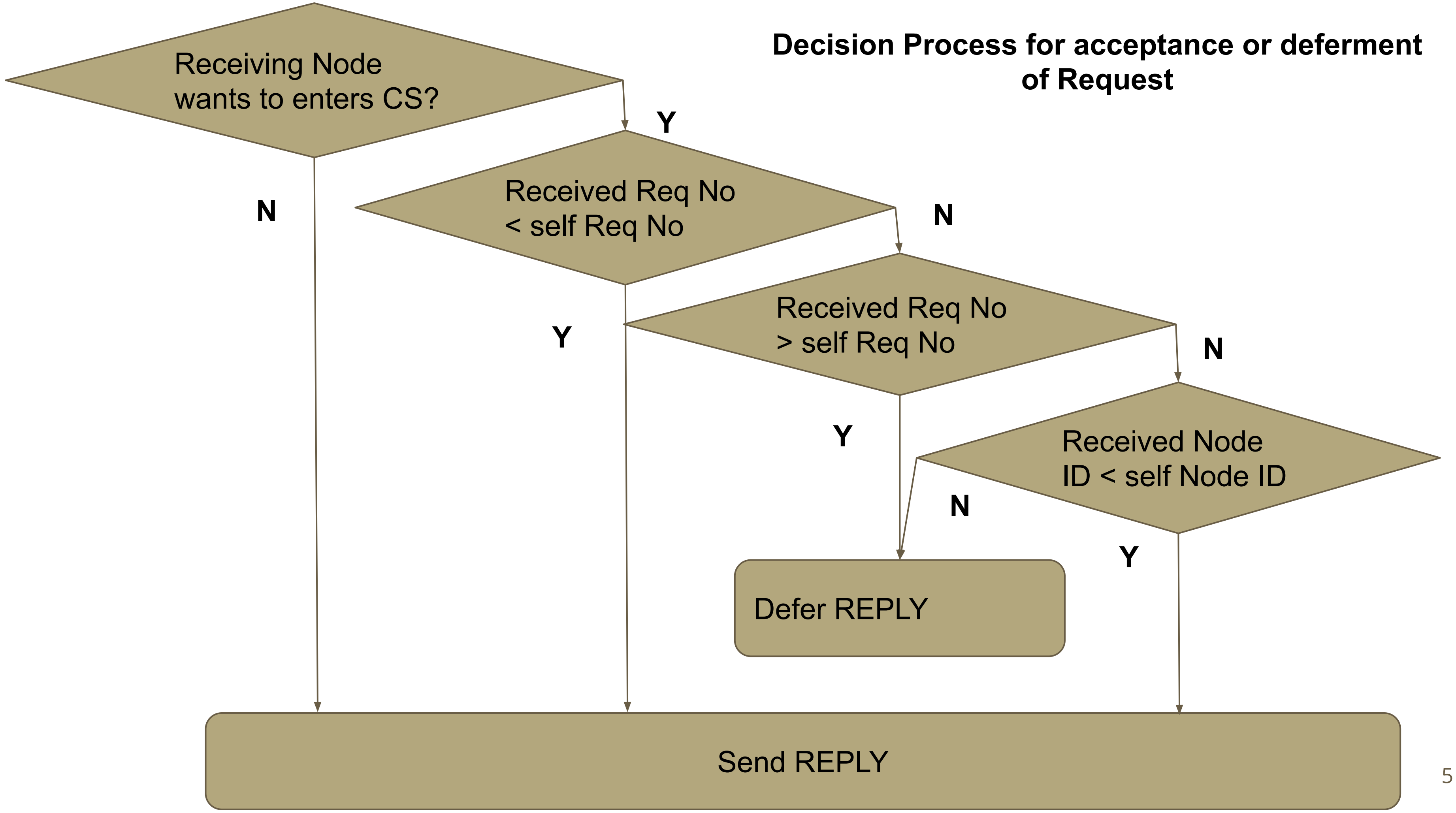
Overview of the Algorithm

- Enables to achieve Mutual Exclusion in a distributed environment
- Communication amongst the Nodes is based on Network Messages
- Crux of the Algorithm:
    - Prerequisite for entering critical section:
        - Send Request message to all other nodes
        - Must have received Reply from all other nodes
    - Post Requisite after exiting critical section:
        - Handle Deferred requests, if any

Steps Involved

- Node which wants to enter the critical section sends REQUEST to all the other Nodes.
- REQUEST message contains the Node ID and a Request Number.
- Other Nodes, on receiving REQUEST shall accept or defer the request.
- Sender on receiving REPLY from all other nodes enters the critical section.
- On exiting from the critical section, it handles the deferred requests.

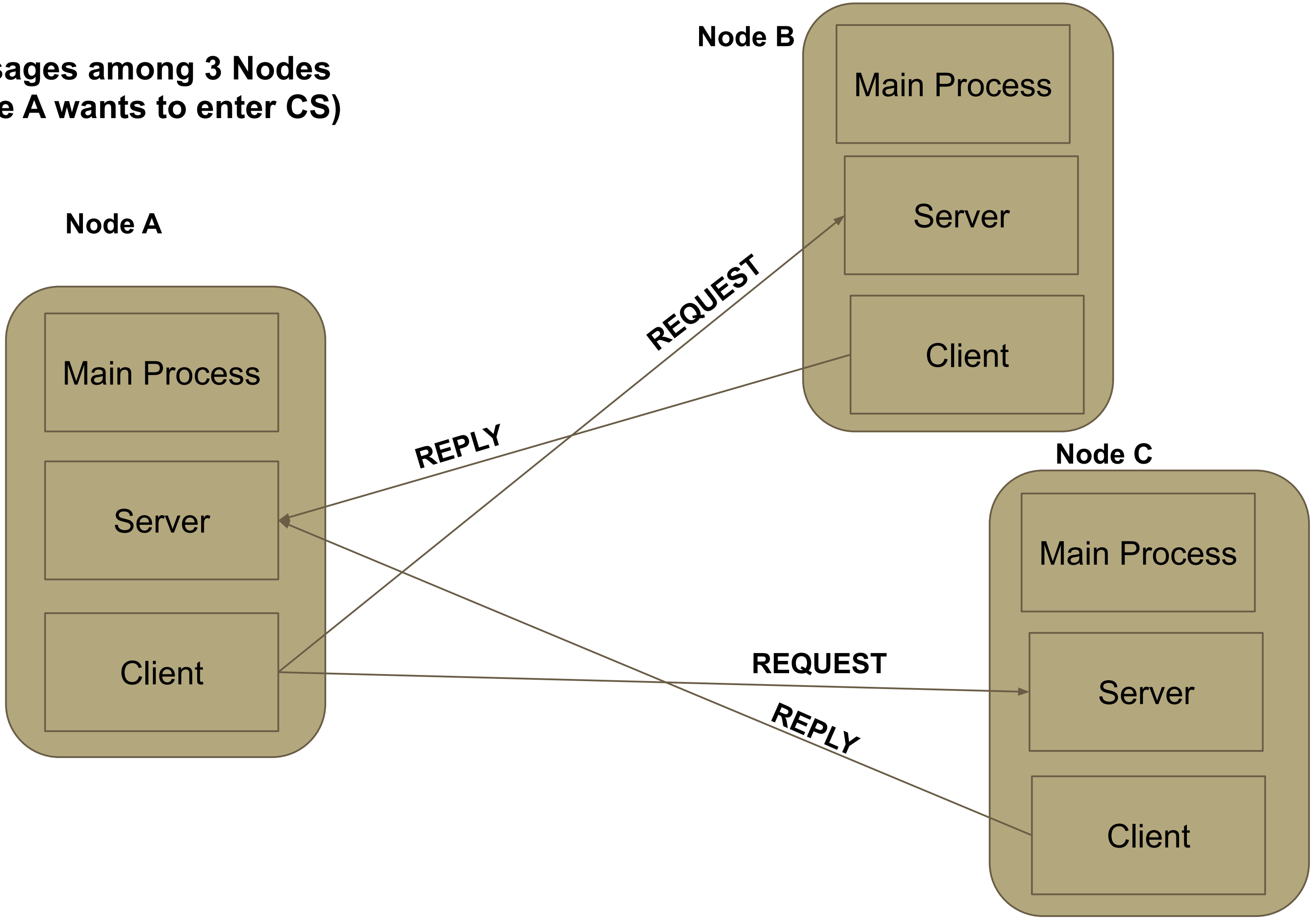
Decision Process for acceptance or deferment of Request


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

Implementation Aspects

- There are N Nodes in the system, numbered from zero to N-1 (these can be pre-defined in the config.txt file)
- The communication is done using TCP/IP socket communication
- Each Node has a Server and Client Component
- On startup, Node starts listening using its Server component
- When a Node wants to communicate it connects to the corresponding Server using its Client component

Messages among 3 Nodes (Node A wants to enter CS)



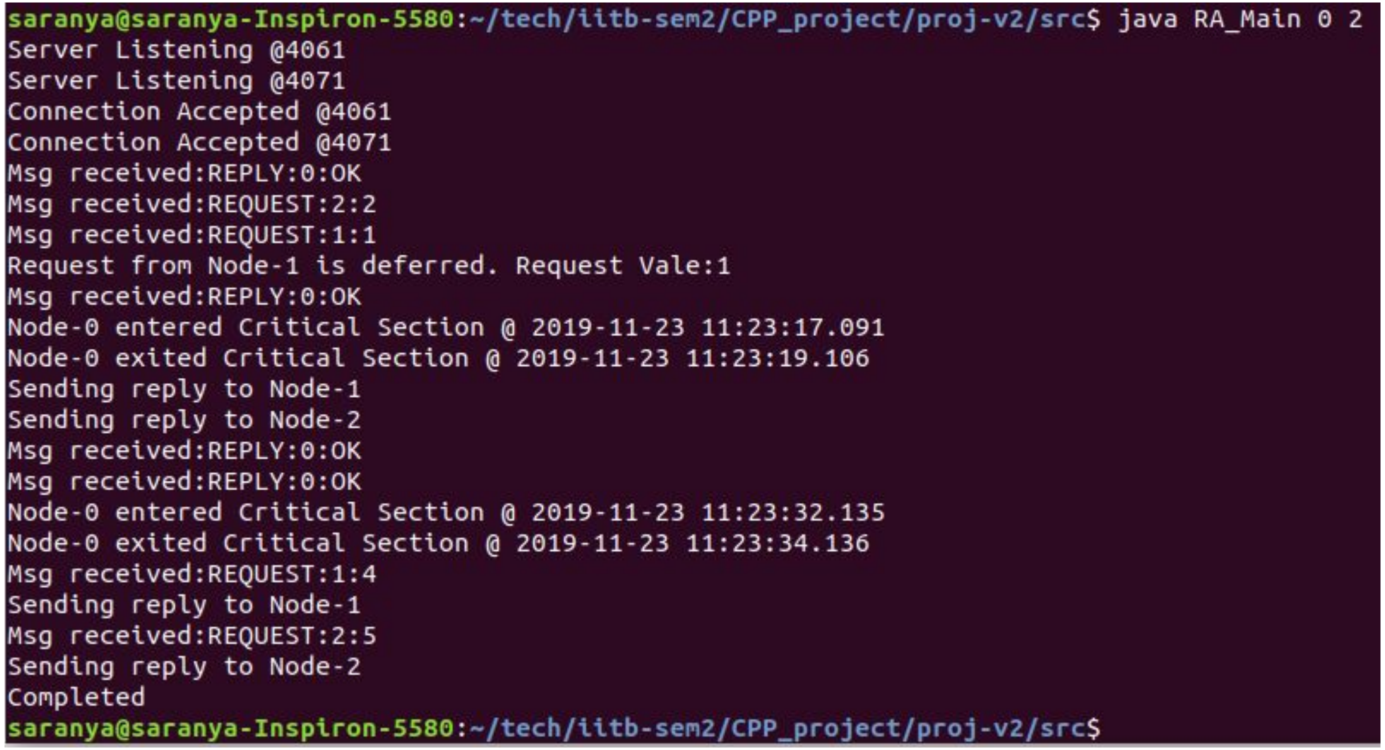
Implementation Details

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

Demo Results

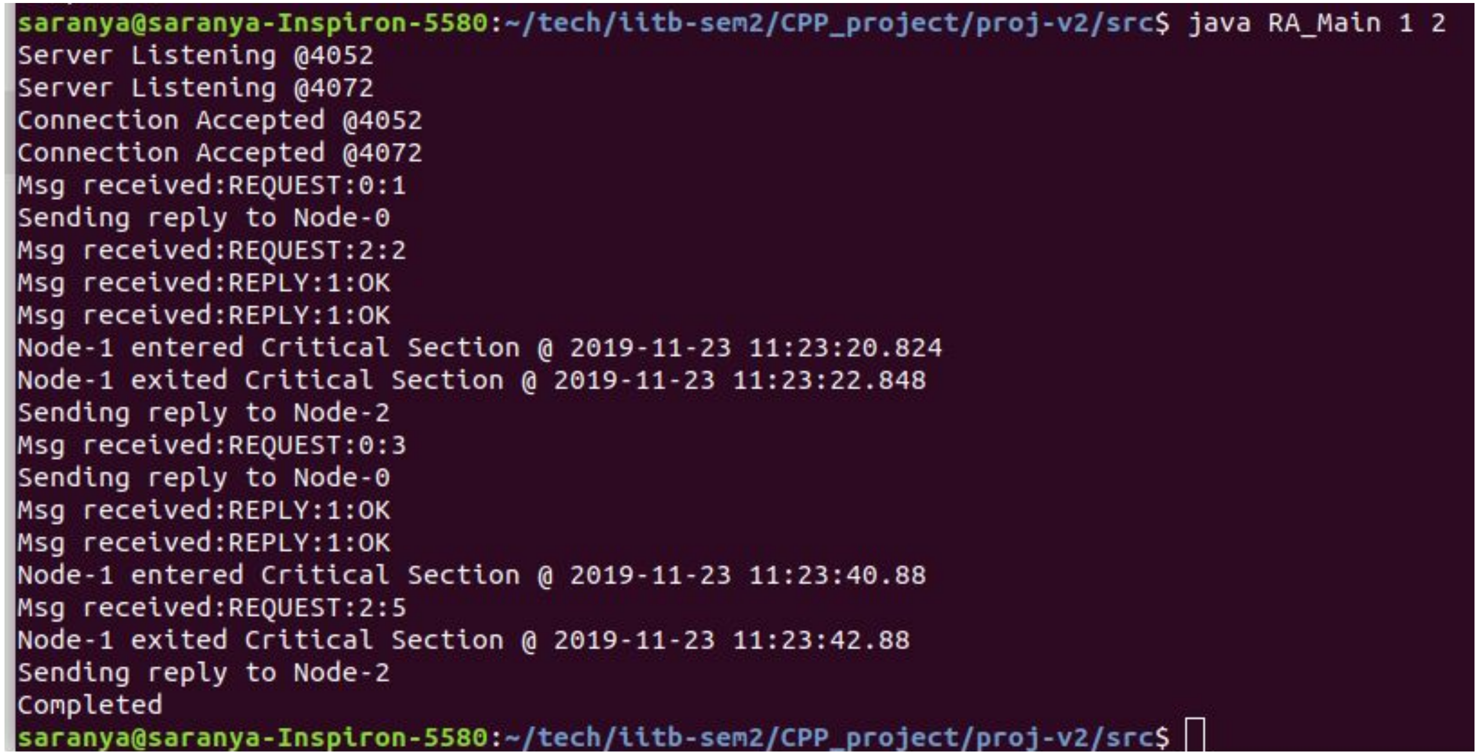
- In the demonstration, 3 Nodes were simulated
- Each of the Nodes shall request for entering the critical section twice.

Demo Results - Node 1



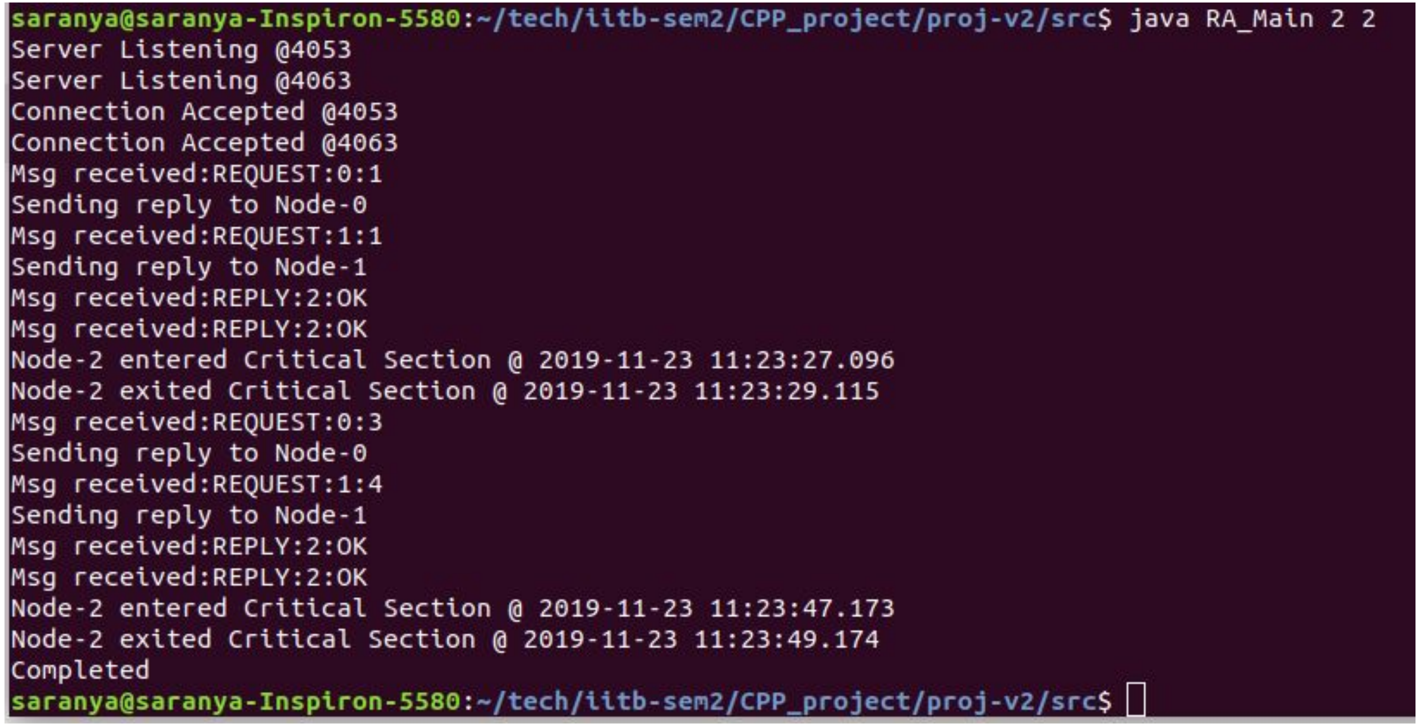
```
saranya@saranya-Inspiron-5580:~/tech/iitb-sem2/CPP_project/proj-v2/src$ java RA_Main 0 2
Server Listening @4061
Server Listening @4071
Connection Accepted @4061
Connection Accepted @4071
Msg received:REPLY:0:OK
Msg received:REQUEST:2:2
Msg received:REQUEST:1:1
Request from Node-1 is deferred. Request Vale:1
Msg received:REPLY:0:OK
Node-0 entered Critical Section @ 2019-11-23 11:23:17.091
Node-0 exited Critical Section @ 2019-11-23 11:23:19.106
Sending reply to Node-1
Sending reply to Node-2
Msg received:REPLY:0:OK
Msg received:REPLY:0:OK
Node-0 entered Critical Section @ 2019-11-23 11:23:32.135
Node-0 exited Critical Section @ 2019-11-23 11:23:34.136
Msg received:REQUEST:1:4
Sending reply to Node-1
Msg received:REQUEST:2:5
Sending reply to Node-2
Completed
saranya@saranya-Inspiron-5580:~/tech/iitb-sem2/CPP_project/proj-v2/src$
```

Demo Results - Node 2



```
saranya@saranya-Inspiron-5580:~/tech/iitb-sem2/CPP_project/proj-v2/src$ java RA_Main 1 2
Server Listening @4052
Server Listening @4072
Connection Accepted @4052
Connection Accepted @4072
Msg received:REQUEST:0:1
Sending reply to Node-0
Msg received:REQUEST:2:2
Msg received:REPLY:1:OK
Msg received:REPLY:1:OK
Node-1 entered Critical Section @ 2019-11-23 11:23:20.824
Node-1 exited Critical Section @ 2019-11-23 11:23:22.848
Sending reply to Node-2
Msg received:REQUEST:0:3
Sending reply to Node-0
Msg received:REPLY:1:OK
Msg received:REPLY:1:OK
Node-1 entered Critical Section @ 2019-11-23 11:23:40.88
Msg received:REQUEST:2:5
Node-1 exited Critical Section @ 2019-11-23 11:23:42.88
Sending reply to Node-2
Completed
saranya@saranya-Inspiron-5580:~/tech/iitb-sem2/CPP_project/proj-v2/src$
```

Demo Results - Node 3



```
saranya@saranya-Inspiron-5580:~/tech/iitb-sem2/CPP_project/proj-v2/src$ java RA_Main 2 2
Server Listening @4053
Server Listening @4063
Connection Accepted @4053
Connection Accepted @4063
Msg received:REQUEST:0:1
Sending reply to Node-0
Msg received:REQUEST:1:1
Sending reply to Node-1
Msg received:REPLY:2:OK
Msg received:REPLY:2:OK
Node-2 entered Critical Section @ 2019-11-23 11:23:27.096
Node-2 exited Critical Section @ 2019-11-23 11:23:29.115
Msg received:REQUEST:0:3
Sending reply to Node-0
Msg received:REQUEST:1:4
Sending reply to Node-1
Msg received:REPLY:2:OK
Msg received:REPLY:2:OK
Node-2 entered Critical Section @ 2019-11-23 11:23:47.173
Node-2 exited Critical Section @ 2019-11-23 11:23:49.174
Completed
saranya@saranya-Inspiron-5580:~/tech/iitb-sem2/CPP_project/proj-v2/src$
```

Conclusions

- The Ricart-Agrawala algorithm for mutual exclusion provides a **deadlock free** and **starvation free** solution.
- The algorithm uses an optimal number of messages, i.e., 2(N-1)

Thank You