# Project Report

## CS 571

## Operating Systems

Implement DME using Ring-Based algorithm

Group Details

No of members - 1

Name: Gaurav Desai

The problem statement requires us to implement a solution to DME (Distributed Mutual Exclusion) problem. Among various DME algorithms, we are required to implement Ring based DME algorithm. The project is meant to apply various concepts we studied throughout our lectures. A simple distributed system needs to be designed to implement concepts like distributed coordination, communication and fault tolerance.

The use case is to reserve University Auditorium by various departments. When a client (An instructor) is willing to reserve the auditorium for a specific day, they will contact the concerned university department. Each department will serve as a separate node. It will try to update the shared data structure or file "Schedule.txt" while servicing the client request.

To enforce distributed mutual exclusion, one needs to satisfy minimally two requirements i.e safety and bounded waiting. There are different algorithms available to solve DME like Raymond's Tree Algorithm, Central-Server Algorithm, Ring-Based DME algorithm etc. We require implementing Ring-Based DME algorithm.

Ring-Based DME algorithm considers a virtual ring to be maintained among the server nodes. Whenever a server receives a request from a client, it needs to enter critical section to serve the request. To enter Critical Section, it requires a token which is a unique identifier involved among all the server nodes. After a specified time, token will be passed to the next node. The node needs to enter critical section needs to request token and only after receiving token it can enter critical section and complete client request. Once the node exits from critical section, it needs to pass the token to the next node. This means that at a particular time only one critical section can be executed and hence only one client can be served. Because of this, it can be said that it satisfies safety requirement. The bounded-waiting requirement too gets satisfied as the token is guaranteed to reach every node eventually with the assumption that each node exits from critical section in finite amount of time.

Implementation details are specified as follows:

There will be a shared data file called “Schedule.txt” which will have information about the reservation dates of auditorium.

There will be one topology file per department(Server) which will have information about whether to generate token from this node, next node and schedule file path. It will have following kind of structure (We are assuming the ring of five department nodes):

- FirstTokenGeneration=true
    - If true this node will generate the starting token, So for one of the five topology files, it will be true and will be false for remaining
- DepartmentName=Housing Department
    - This reflects the department name of the node which will be used in messaging to give real feel
- ScheduleFileLocation=G:/Schedule.txt
    - This is where you have to provide the shared data file “Schedule.txt” path
- CurrentNode=4001
    - Port number of this node
- NextNode=4002
    - Port number of next node
- PreviousNode=4005
    - Port number of previous node

Say for example to create a ring of five department nodes, we will run five nodes as follows:

1) java edu.gmu.os.**Department** ./Dept_Housing.txt 5 6
2) java edu.gmu.os.**Department** ./Dept_Education.txt 5 6
3) java edu.gmu.os.**Department** ./Dept_Engineering.txt 5 6
4) java edu.gmu.os.**Department** ./Dept_History.txt 5 6
5) java edu.gmu.os.**Department** ./Dept_Science.txt 5 6

Sample topology files have been provided which will make a ring functional with the token generated from one of the five nodes and then passed to the next node in d2 seconds here we have 6 seconds. Also d1 is specified as 5 seconds which is the time by which critical section should get executed.

Now we can run the client program(RMIClient.java) which will ask for the port number of the node to which it should connect to. Then it will make a connection using Java RMI. Then it will ask following options:

```
==========================================================
Select the operation from options below:
Look-up (press 1)
Reserve (press 2)
Exit    (press 3)
Your Option: <Enter your desire option> 2
Enter the date for the month of december: <Enter day of month in dd format> 5
<Here it will show the response received from the department server>
==========================================================
```

If for the entered date (here 5th) auditorium is available then reserve action will update the Schedule file with the name of the department who has made the reservation against that date (like 5, Housing Department). This means that if you make the same request again for the same date then it should the auditorium unavailable.

**Gaurav Desai**
gdesai3@masonlive.gmu.edu