



AOS Project Report

‘Experimental Analysis of Mutual
Exclusion Algorithms’

Bhanu Arora (bxa151730)

Divyanshu Paliwal (dxd151630)

Mayur Talole (mnt150230)



Table of Contents

| | |
|--|----|
| Mutual Exclusion Algorithms | 2 |
| Lamport protocol for mutual exclusion | 2 |
| Requesting process..... | 2 |
| Other processes..... | 2 |
| Flow chart of algorithm for Lamport mutual exclusion algorithm | 3 |
| Roucairol-Carvalho Algorithm | 4 |
| Experimental setup..... | 5 |
| Implementation Platform specification..... | 5 |
| Code execution: | 6 |
| Sample Configuration File Format..... | 8 |
| Observations: | 10 |
| Experimental analysis for Roucairol-Carvalho algorithm | 10 |
| Analysis for varying Number of nodes: | 10 |
| Analysis for varying inter-request delay 'd' | 12 |
| Analysis by varying mean Critical execution time 'c' | 14 |
| Message Complexity | 14 |
| Analysis for Lamport Mutual exclusion algorithm | 16 |
| Analysis by varying number of nodes 'n' | 16 |
| Analysis by varying inter-request delay 'd' | 18 |
| Analysis by varying mean Critical execution time 'c' | 21 |
| Experimental comparison of both mutual exclusion algorithms..... | 23 |
| For message complexity | 23 |
| For Average response time..... | 25 |
| For System throughput..... | 27 |
| Conclusion | 29 |
| References..... | 29 |

Mutual Exclusion Algorithms

Lamport protocol for mutual exclusion

Lamport's Distributed Mutual Exclusion Algorithm is a contention-based algorithm for mutual exclusion on a distributed system. Main idea about the Lamport's mutual exclusion protocol is as follows:

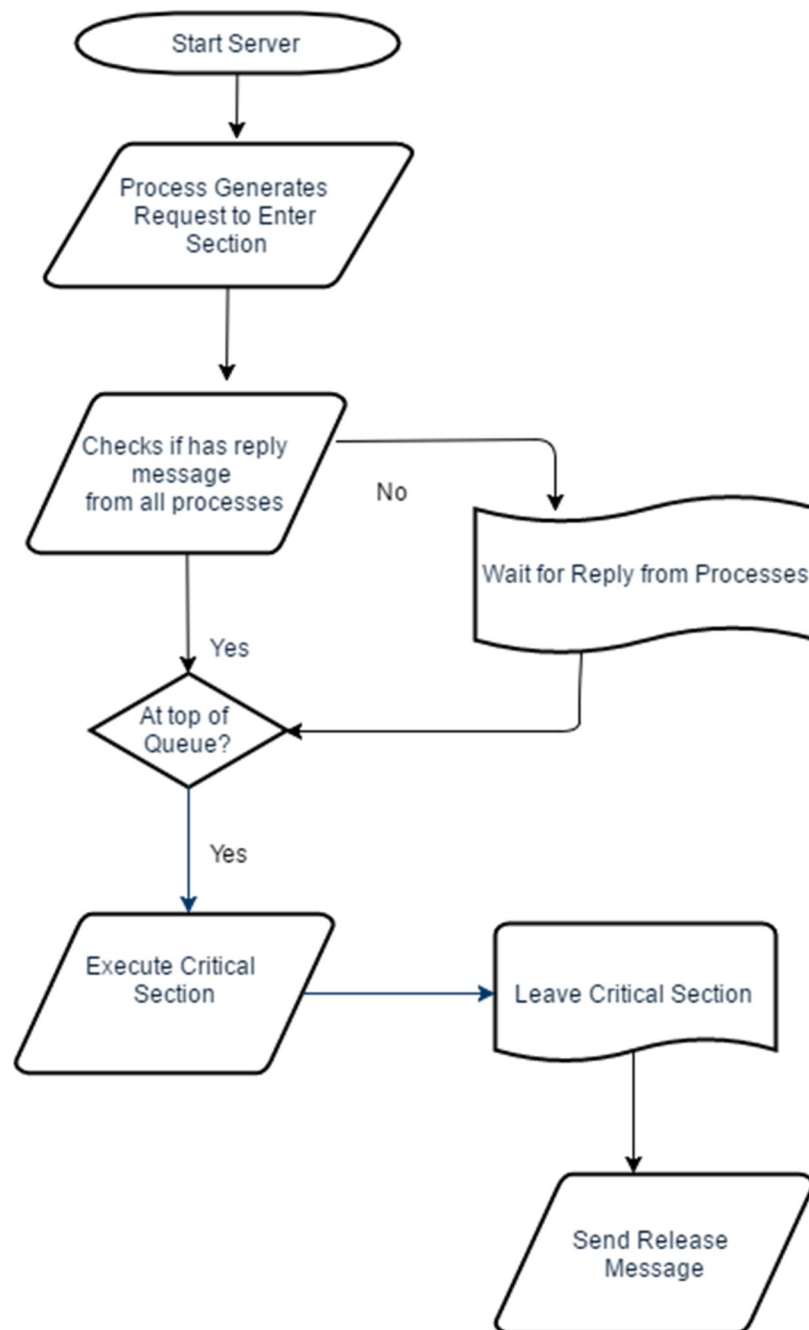
Requesting process

1. Pushing its request in its own queue (ordered by time stamps)
2. Sending a request to every node.
3. Waiting for replies from all other nodes.
4. If own request is at the head of its queue and all replies have been received, enter critical section.
5. Upon exiting the critical section, remove its request from the queue and send a release message to every process.

Other processes

- i. After receiving a request, pushing the request in its own request queue (ordered by time stamps) and reply with a time stamp.
- ii. After receiving release message, remove the corresponding request from its own request queue.
- iii. If own request is at the head of its queue and all replies have been received, enter critical section.

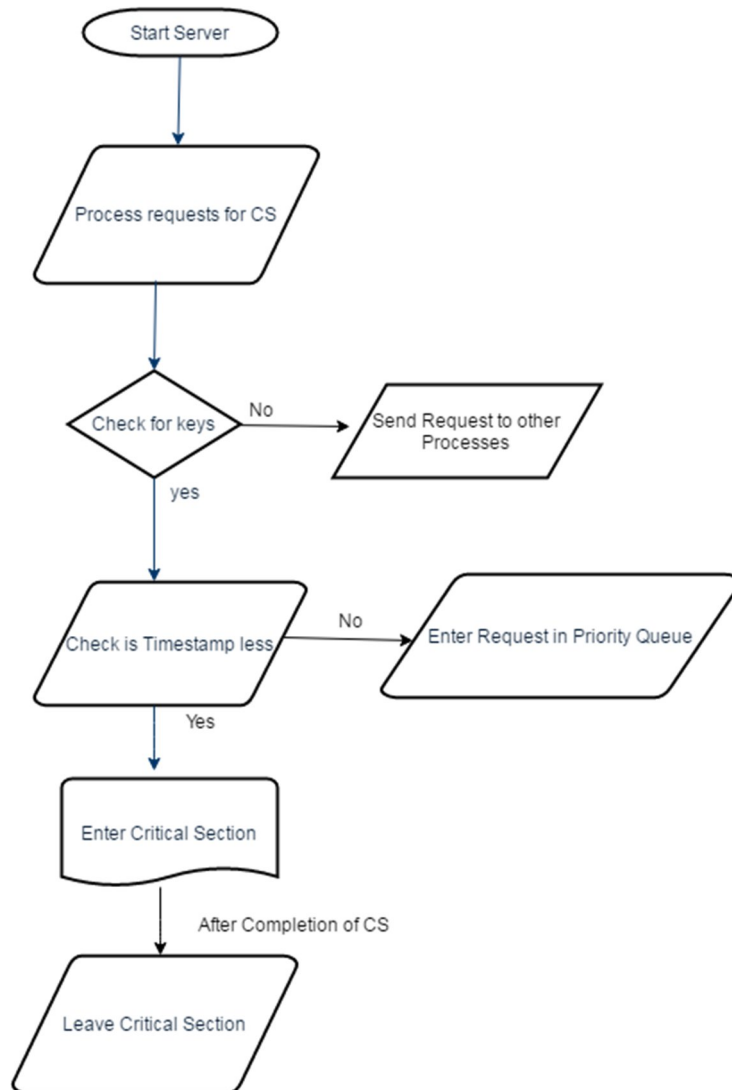
Flow chart of algorithm for Lamport mutual exclusion algorithm



Roucairol-Carvalho Algorithm

Roucairol and Carvalho proposed an improvement to the Ricart-Agrawala algorithm by observing that once a site S_i has received a REPLY message from a site S_j , the authorization implicit in this message remains valid until S_i sends a REPLY message to S_j (which happens only after the reception of a REQUEST message from S_j). Therefore, after site S_i has received a REPLY message from site S_j , site S_i can enter its critical section any number of times without requesting permission from site S_j until S_i sends a REPLY message to S_j .

-Once i has received a REPLY from j, it does not need to send a REQUEST to j again unless it



sends a again unless it sends a REPLY to REPLY to j (in response to a REQUEST from j)

Message complexity varies between 0 and $2(n - 1)$ depending on the request pattern depending on the request pattern.

Experimental setup

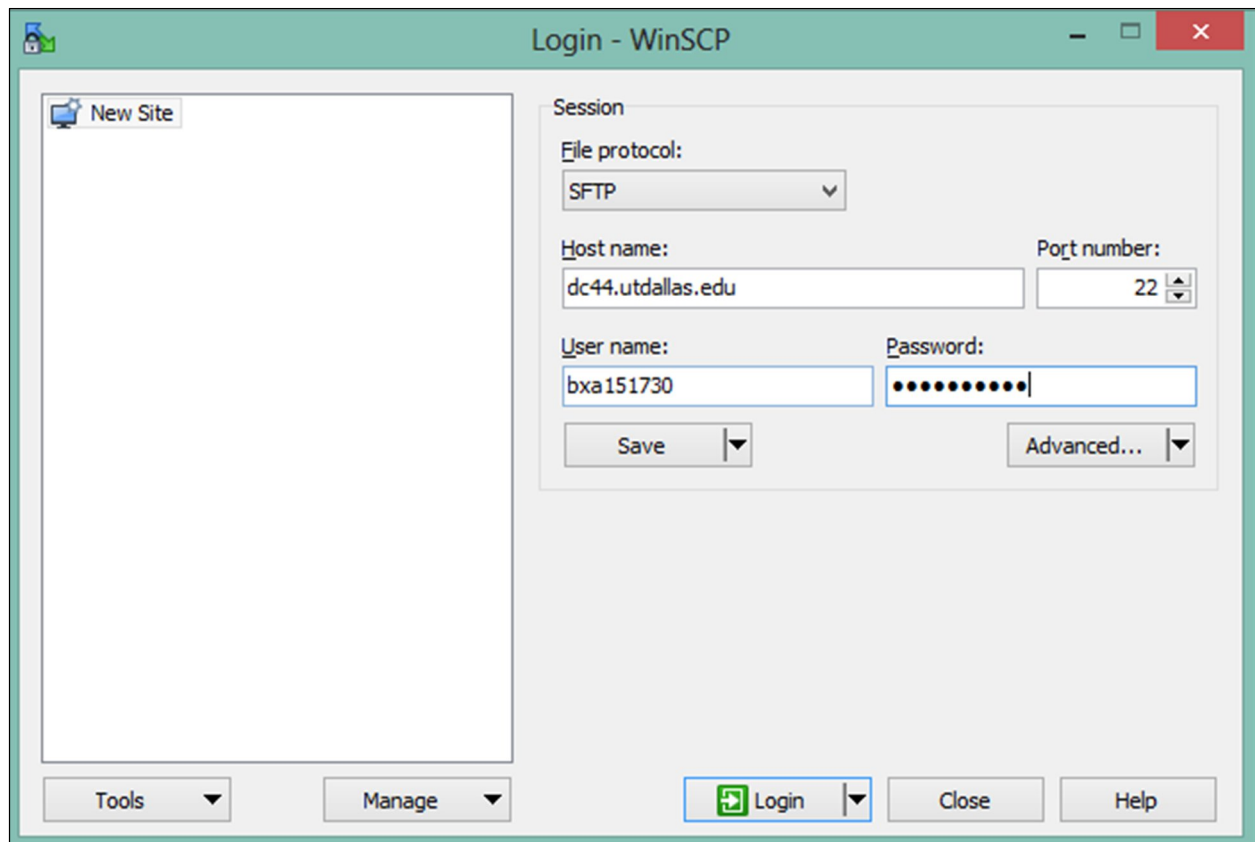
Implementation Platform specification

- Machines used: Toshiba Satellite C55
- PC configuration: intel core i5-2000 CPU 2.20GHz X 4
- Operating System: Ubuntu 15.04 (64-bit)

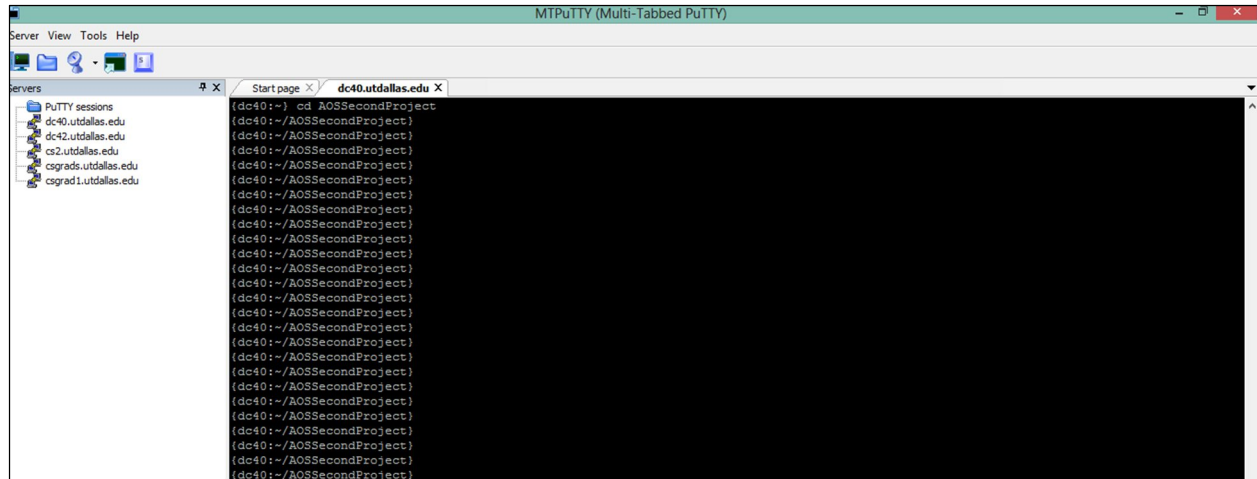
- Hard disk: 243.43 GB
- Memory of machine: 7.7 GiB
- IDE used for coding: Java Eclipse IDE
- Java compiler version: Java 1.8.0_45
- Virtual machine used: dc044@utdallas.edu / cs1.utdallas.edu
- Other machines used in system: dcXX.utdallas.edu

Code execution:

First move the respective code and config.txt file into the server account using Winscp. Login to DC machines using NetID and Password. Login to Winscp and place the code in a folder on DC machine.



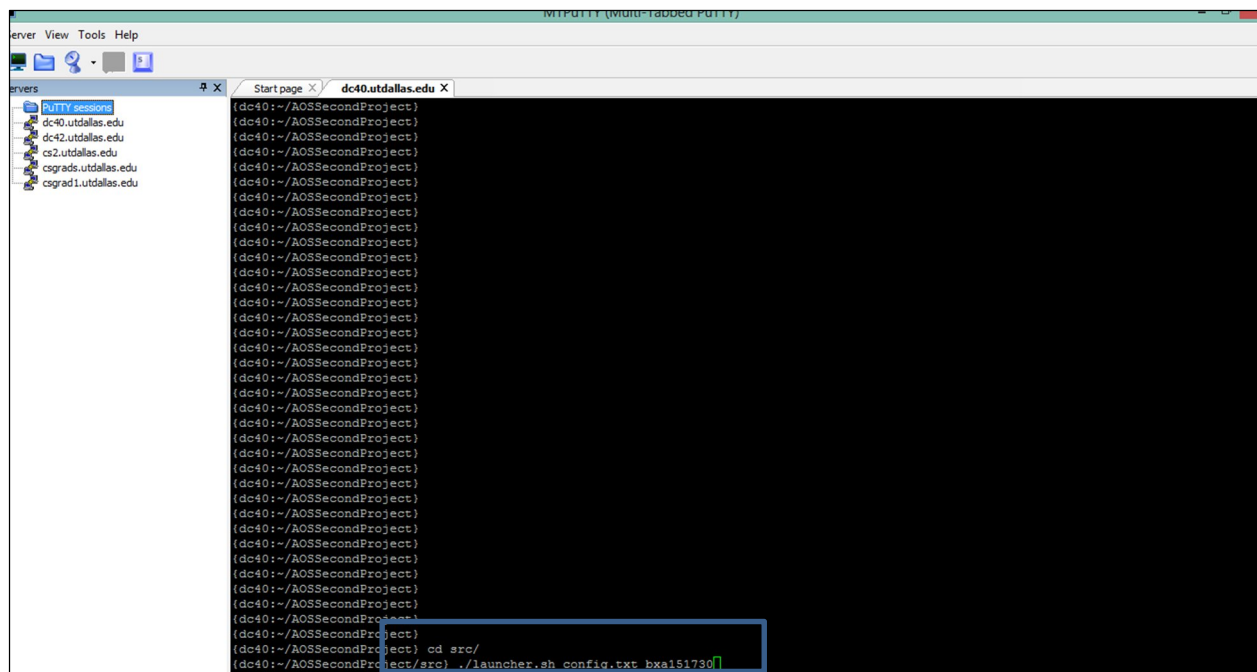
Go to the respective directory of Lamport and roucairol-carvalho protocol where the code is being stored using "cd" command.



A screenshot of the MTPuTTY (Multi-Tabbed PuTTY) terminal window. The left sidebar shows a tree view of 'servers' with 'dc40.utsdallas.edu' selected. The main terminal pane shows the output of a 'cd' command, listing the contents of the directory ~/AOSSecondProject. The output is a long list of directory names, including 'dc40:~/AOSSecondProject' repeated many times.

Once you are inside the directory make changes into config.txt to make changes into number of nodes and number of requests of each node. In order to start the system, run the following command:

```
./launcher.sh config.txt <netID>
```



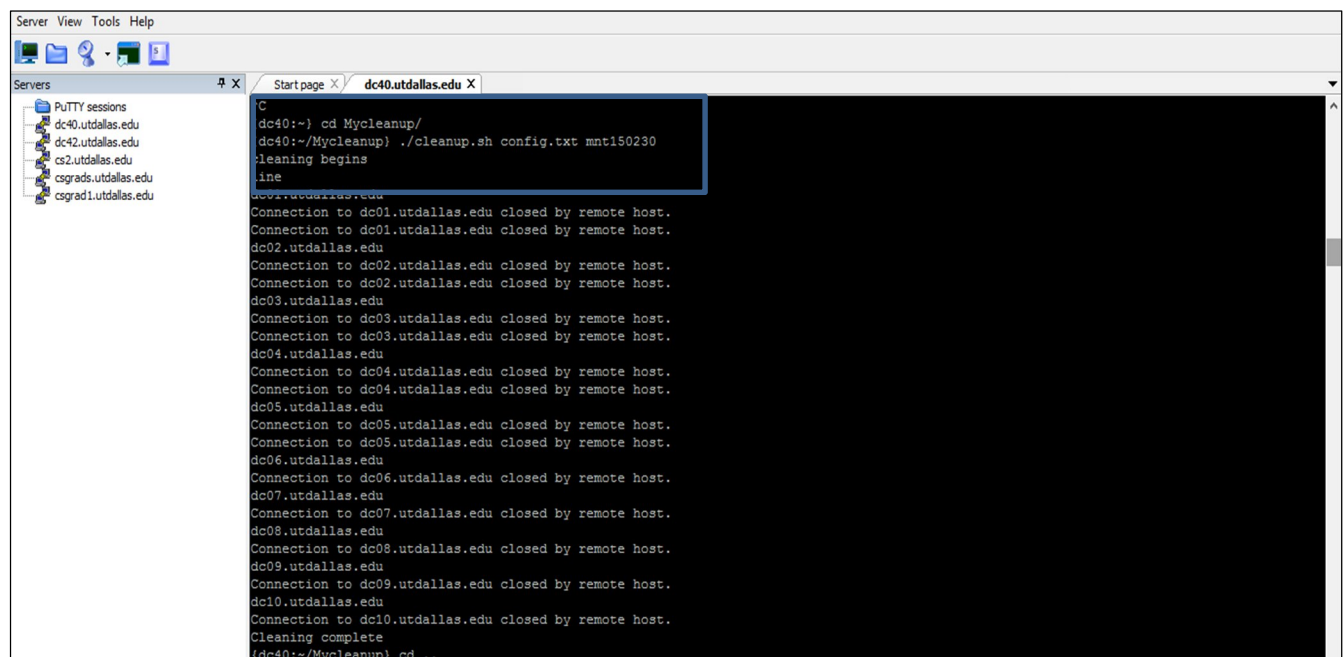
A screenshot of the MTPuTTY terminal window, showing the execution of the script. The terminal output shows the same directory listing as the previous screenshot. At the bottom, a blue box highlights the command being executed: `./launcher.sh config.txt bxa151730`. The terminal also shows the output of the command, which is a long list of directory names, including 'dc40:~/AOSSecondProject' repeated many times.

Sample Configuration File Format

```
# Number of Nodes, D mean in milliseconds, C mean in milliseconds, Number Of Requests.
5 20 10 1000 # This is a comment

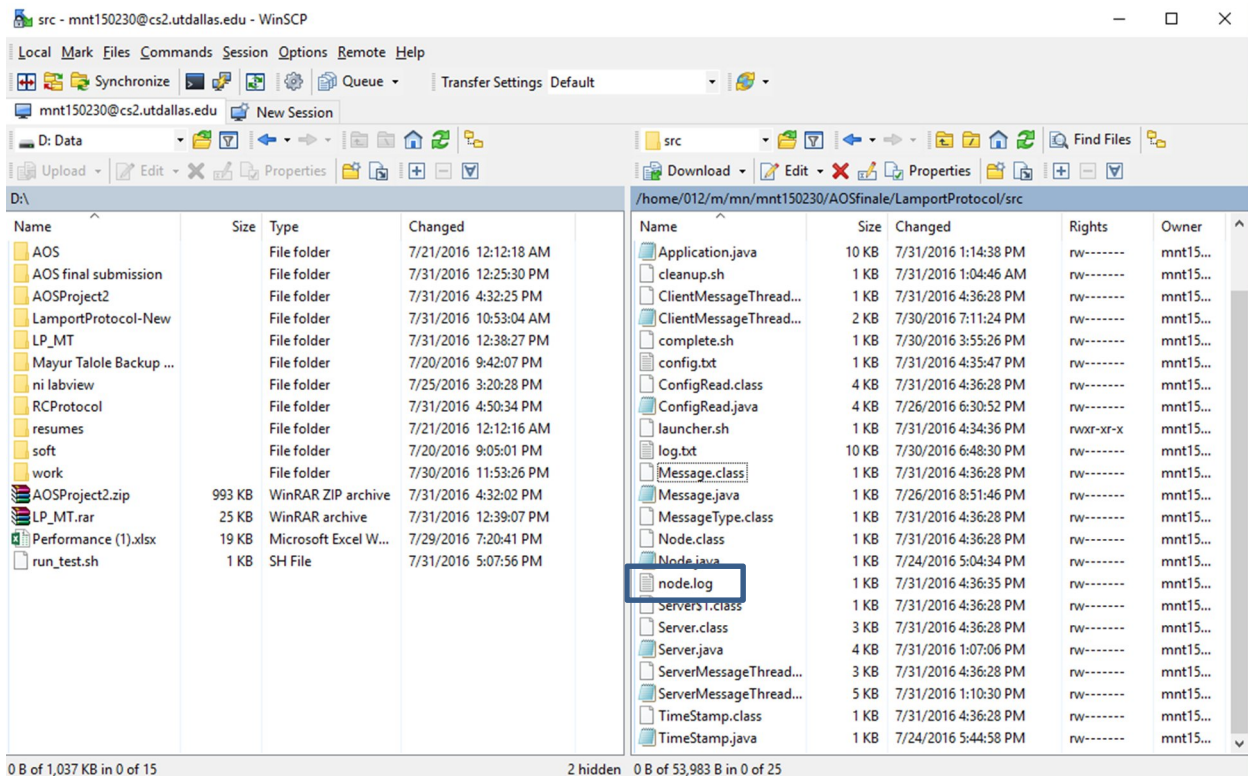
# Location of each node
#
# Format is:
# <Identifier> <Hostname> <Port>
#
# Each node shall have exactly one line in this section. Each line shall start with a unique ID for the node. Nodes
# shall be numbered by integers in the range [0, n-1]. The first token shall be an integer, which consists of the
# unique ID for the node. The second shall be a string, which is the hostname on which your program for the node
# shall be hosted. The third token shall be the port on which the node listens for incoming connections. Each of
# these three tokens shall be separated by one or more whitespace characters (\s*).
#
0 dc01.utdallas.edu 19001 # Node 1
1 dc02.utdallas.edu 19002 # Node 2
2 dc03.utdallas.edu 19003 # Node 3
3 dc04.utdallas.edu 19004 # Node 4
4 dc05.utdallas.edu 19005 # Node 5
```

Running the Cleanup file, after executing the program.



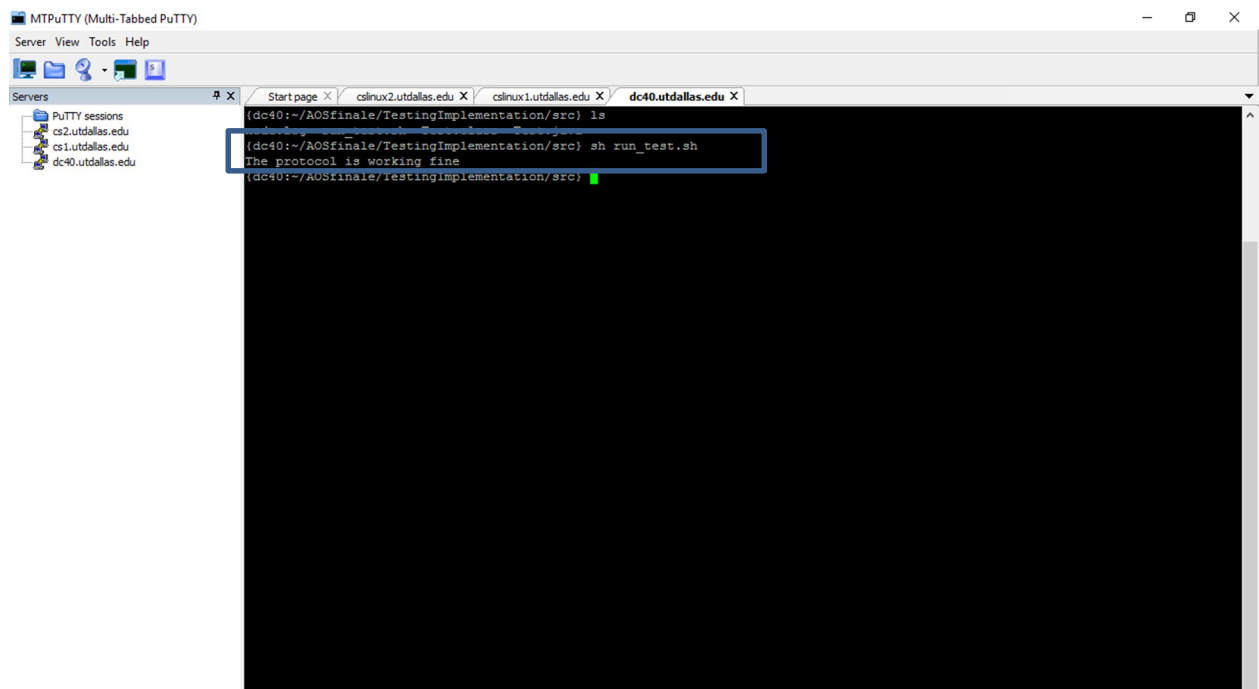
```
Server View Tools Help
Servers
  dc40.utdallas.edu
  dc42.utdallas.edu
  cs2.utdallas.edu
  csgrads.utdallas.edu
  csgrad1.utdallas.edu
  Start page X
  dc40.utdallas.edu X
C
dc40:~/Mycleanup/
dc40:~/Mycleanup/ ./cleanup.sh config.txt mnt150230
Cleaning begins
line
dc40:~/Mycleanup/
Connection to dc01.utdallas.edu closed by remote host.
Connection to dc01.utdallas.edu closed by remote host.
dc02.utdallas.edu
Connection to dc02.utdallas.edu closed by remote host.
Connection to dc02.utdallas.edu closed by remote host.
dc03.utdallas.edu
Connection to dc03.utdallas.edu closed by remote host.
Connection to dc03.utdallas.edu closed by remote host.
dc04.utdallas.edu
Connection to dc04.utdallas.edu closed by remote host.
Connection to dc04.utdallas.edu closed by remote host.
dc05.utdallas.edu
Connection to dc05.utdallas.edu closed by remote host.
Connection to dc05.utdallas.edu closed by remote host.
dc06.utdallas.edu
Connection to dc06.utdallas.edu closed by remote host.
dc07.utdallas.edu
Connection to dc07.utdallas.edu closed by remote host.
dc08.utdallas.edu
Connection to dc08.utdallas.edu closed by remote host.
dc09.utdallas.edu
Connection to dc09.utdallas.edu closed by remote host.
dc10.utdallas.edu
Connection to dc10.utdallas.edu closed by remote host.
Cleaning complete
(dc40:~/Mycleanup) cd ..
```

After running the code, output log file will be generated inside the same folder containing all the CS



enter and CS exit time of each node.

The log file can be tested by writing the following command in window which lets you know that if there



are any violations or not.

Observations:

The behavior of system parameters such as message complexity, system throughput and average system time with varying number of nodes (processes) as follows.

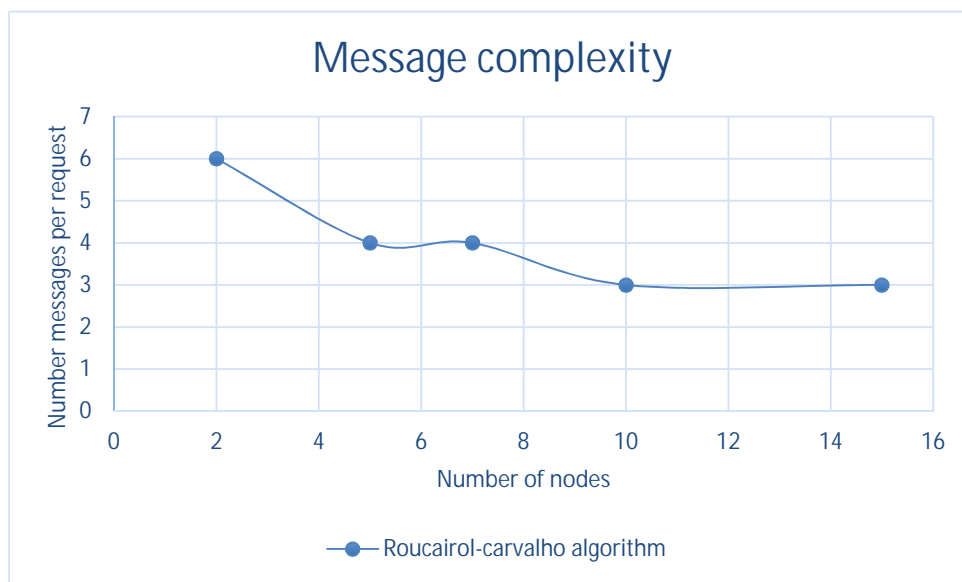
Experimental analysis for Roucairol-Carvalho algorithm

Analysis for varying Number of nodes:

Configuration:

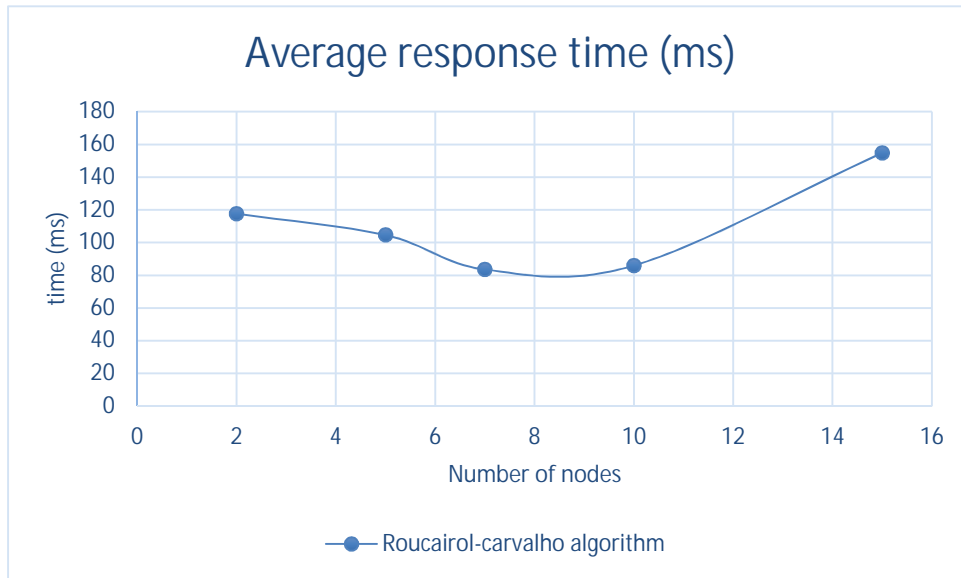
- Number of Nodes=2,5,7,10,15
- Inter Request Delay=50
- Critical Section execution Time=70
- Number of requests per each node=100

Message complexity



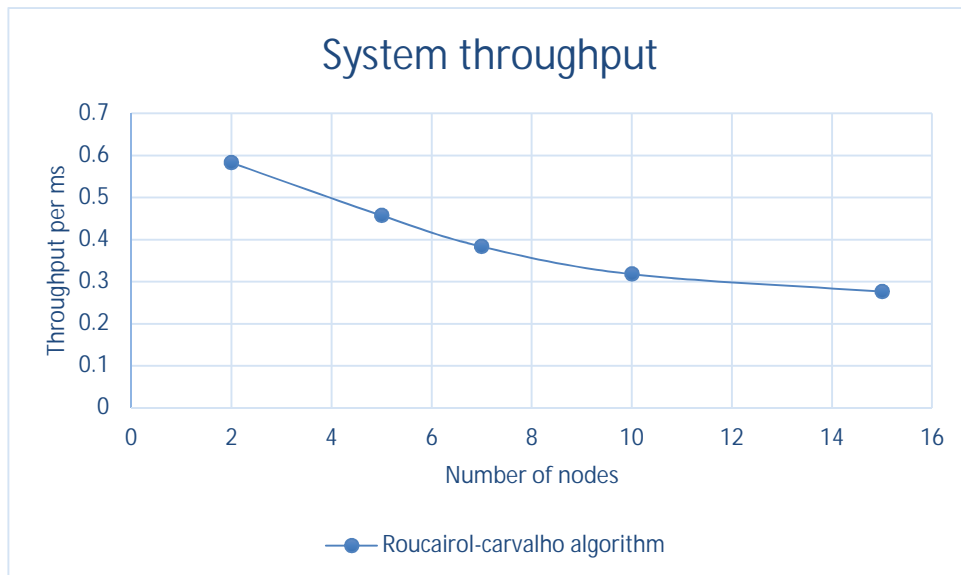
The above graph is for experiment with the message complexity and the number of nodes. Message complexity is the number of messages required per CS execution by a site. Initial number of messages per request were 6, later it reduces to 3 messages and remains constant.

Average response time



The above graph is for experiment with the number of nodes and the response time. Response Time is the time interval a request waits for its CS execution to be over after its request messages have been sent out. Response time for RC algorithm increases gradually as per increase in the number of nodes.

System throughput



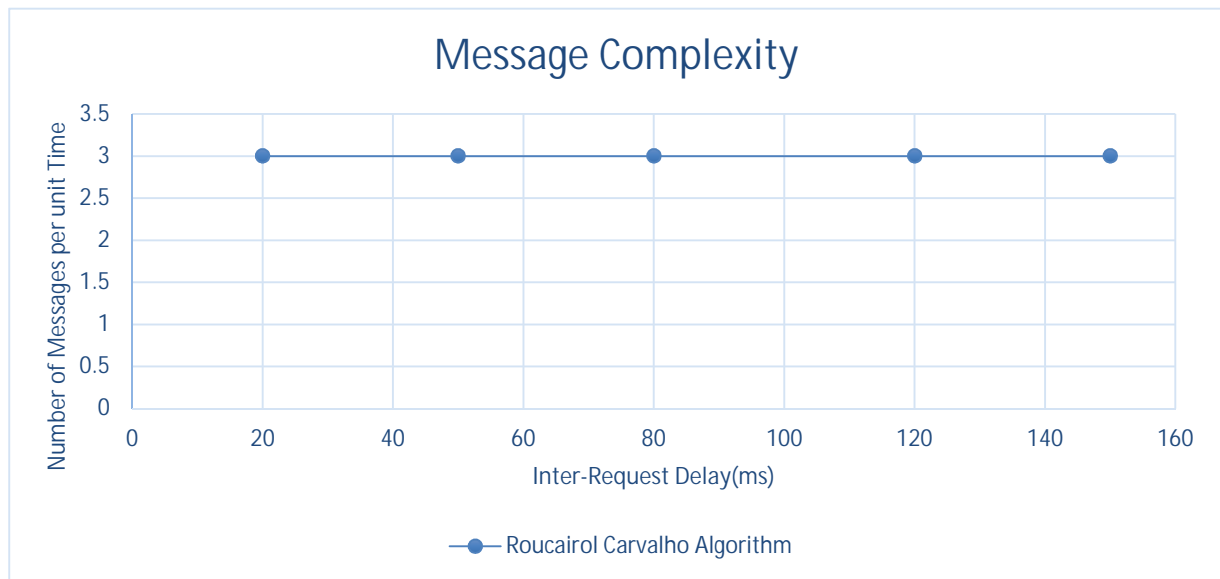
The above graph is for an experiment with the number of nodes and system throughput. System Throughput is the rate at which the system executes requests for the CS. As the number of nodes in system increases the system throughput reduces.

Analysis for varying inter-request delay 'd'

Configuration:

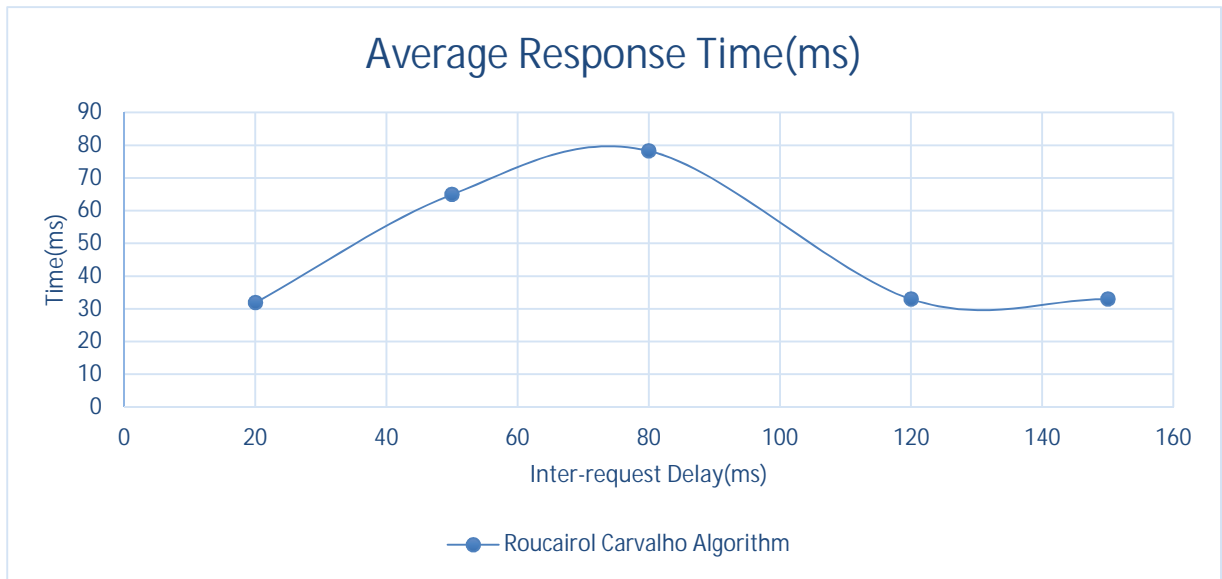
- Number of Nodes=5
- Inter Request Delay=20,50,80,120,150
- Critical Section execution Time=70
- Number of requests per each node=100

Message complexity



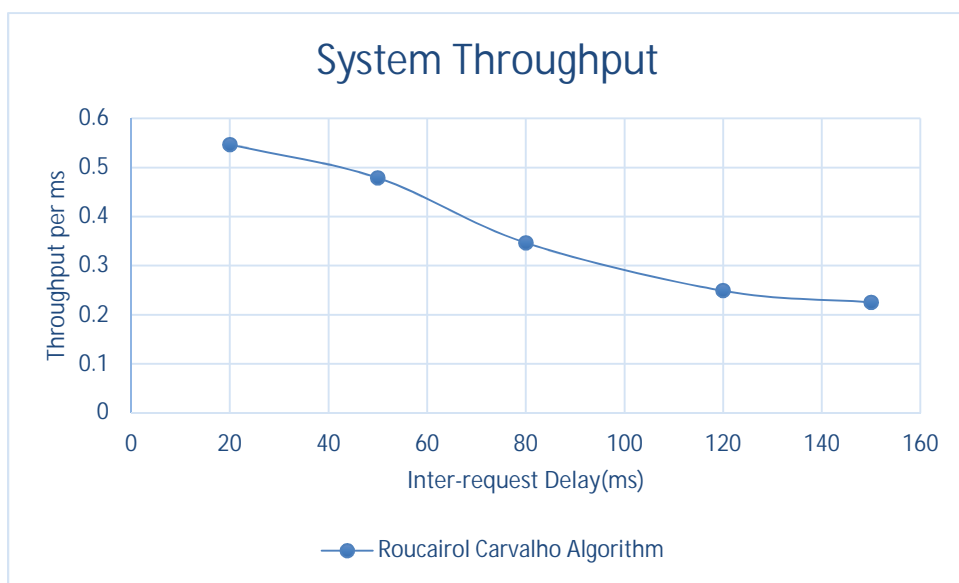
The above graph is for experiment with the message complexity and the Inter Request Delay. Message complexity is the number of messages required per CS execution by a site. The number of messages involved for each request remains same for the system with RC algorithm for varying inter-request delays.

Average response time



The above graph is for experiment with the Inter Request Delay and the response time. Response Time is the time interval a request waits for its CS execution to be over after its request messages have been sent out. The average response time initially increases and later decreases as per the increase in the inter-request delays.

System throughput



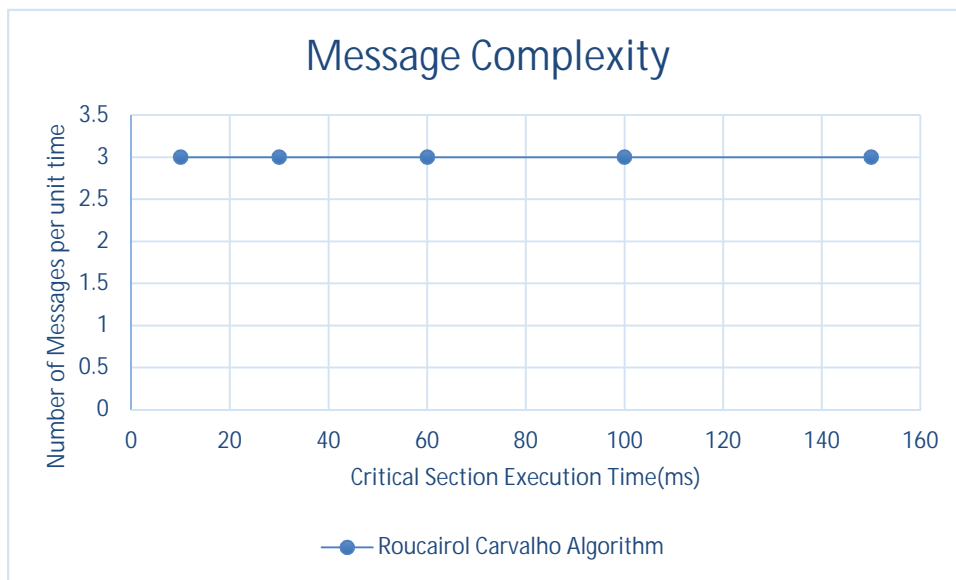
The above graph is for an experiment with the Inter Request Delay and system throughput. System Throughput is the rate at which the system executes requests for the Critical Section. Drastic drop in system throughput is seen when inter-request delay is increased.

Analysis by varying mean Critical execution time 'c'

Configuration:

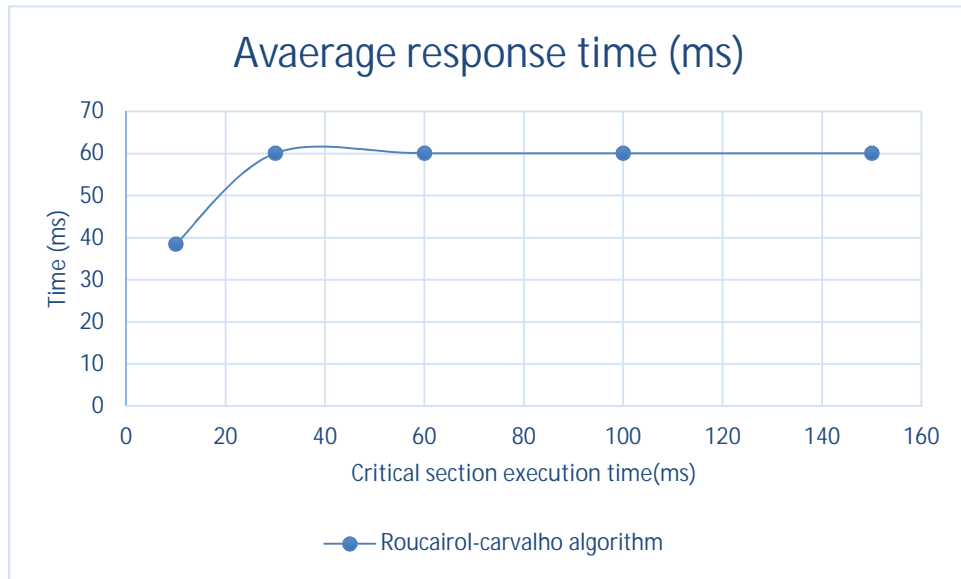
- Number of Nodes=5
- Inter Request Delay=20
- Critical Section execution Time=10,30,60,100,150
- Number of requests per each node=100

Message Complexity



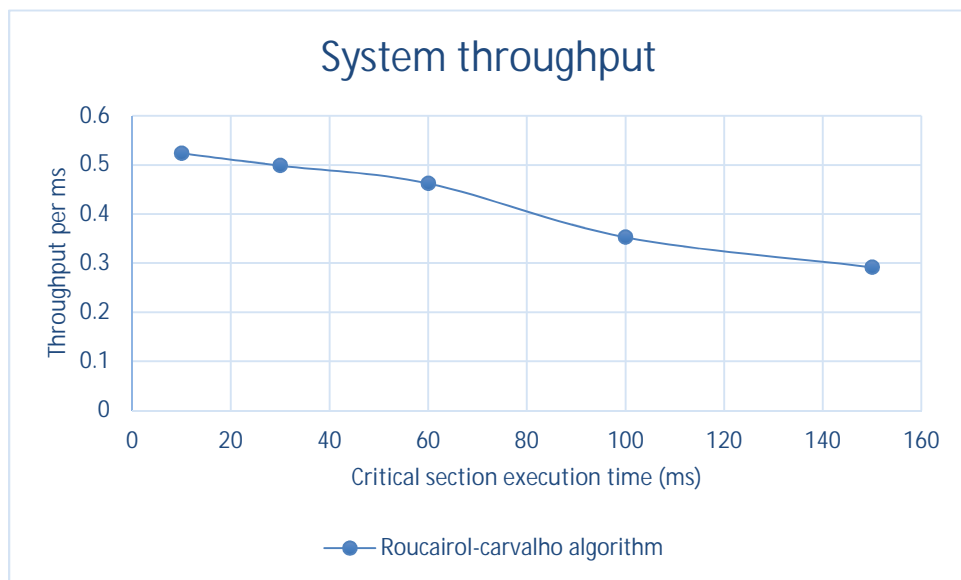
The above graph is for experiment with the Critical Section Execution Time and the Number of messages sent per unit time. Message complexity is the number of messages required per CS execution by a site. The number of messages involved for each request remains same for the system with RC algorithm for varying Critical section execution time.

Average Response time



The above graph is for experiment with the Critical Section Execution Time and the time in milliseconds. Response Time is the time interval a request waits for its CS execution to be over after its request messages have been sent out. Initially the average response time increases and remains same for increasing the critical section execution time.

System throughput



The above graph is for experiment with the Inter Critical Section Execution time and the throughput. It is the rate at which the system executes requests for the Critical Section. There is great fall in system throughput as CS execution time increases.

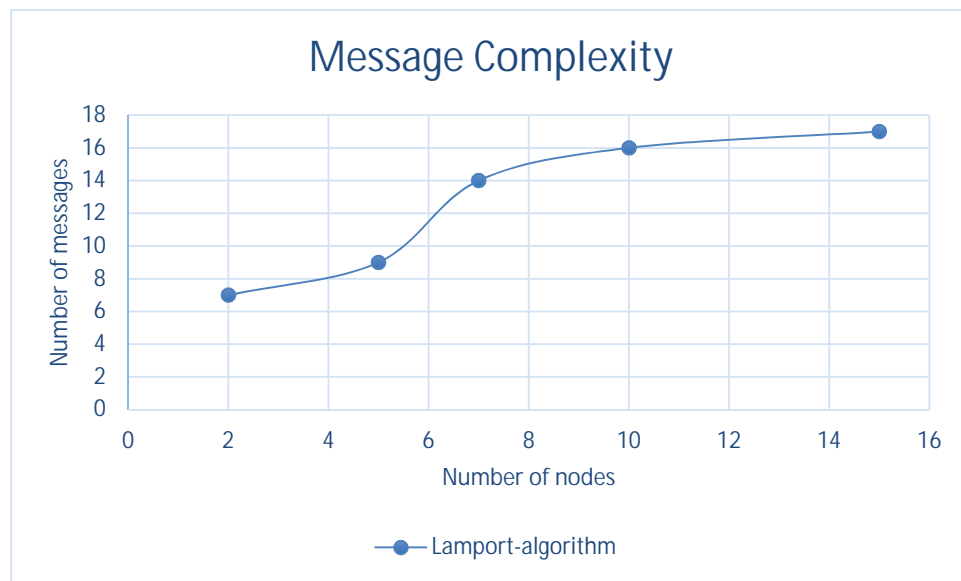
Analysis for Lamport Mutual exclusion algorithm

Analysis by varying number of nodes 'n'

Configuration:

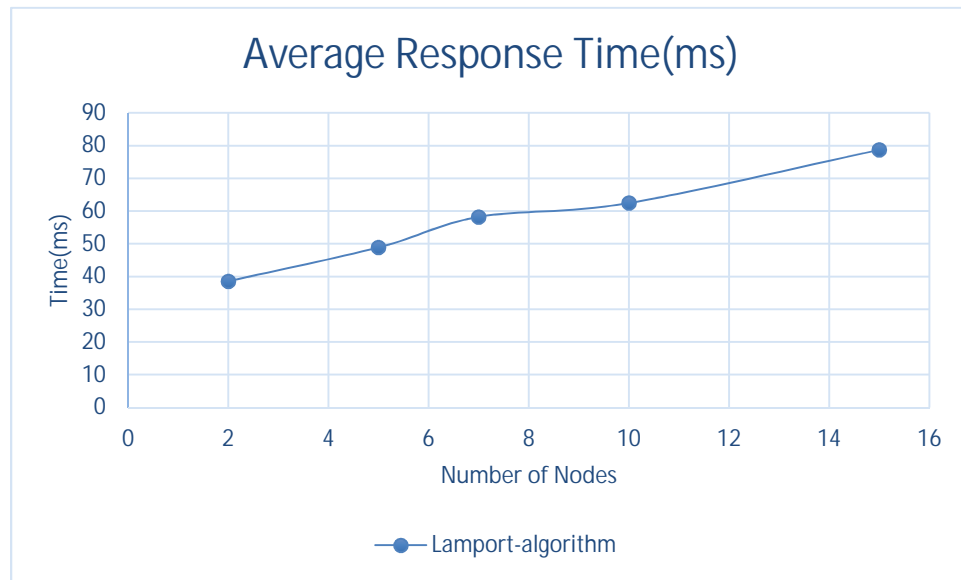
- Number of Nodes=2,5,7,10,15
- Inter Request Delay=50
- Critical Section execution Time=70
- Number of requests per each node=100

Message Complexity:



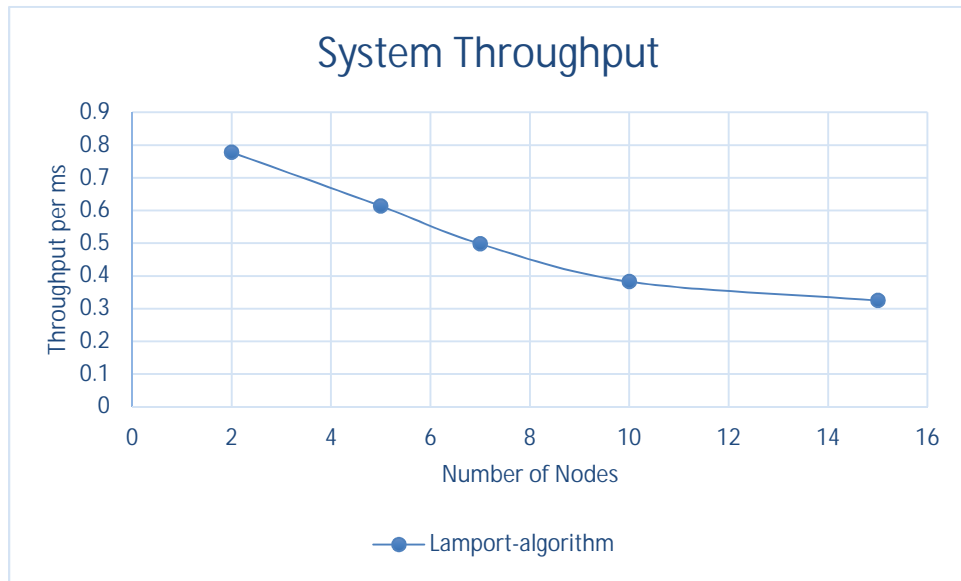
The above graph is for experiment with the message complexity and the number of nodes. Message complexity is the number of messages required per CS execution by a site. For the system with Lamport algorithm the message complexity gradually increases with increase in the number of nodes.

Average Response time:



The above graph is for experiment with the number of nodes and the response time. Response Time is the time interval a request waits for its CS execution to be over after its request messages have been sent out. Average response time increases with number of nodes for Lamport mutual exclusion algorithm.

System throughput



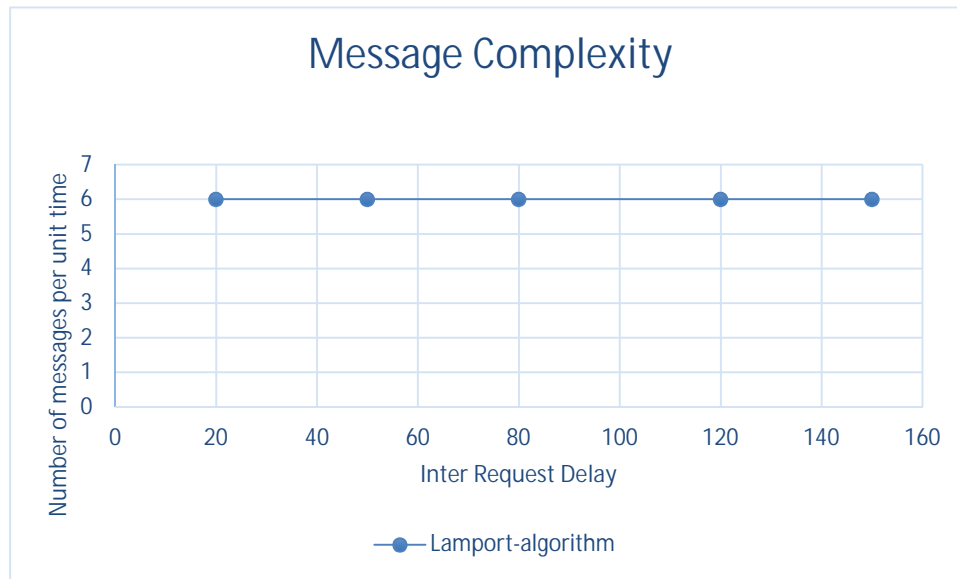
The above graph is for an experiment with the number of nodes and system throughput. System Throughput is the rate at which the system executes requests for the CS.

Analysis by varying inter-request delay 'd'

Configuration:

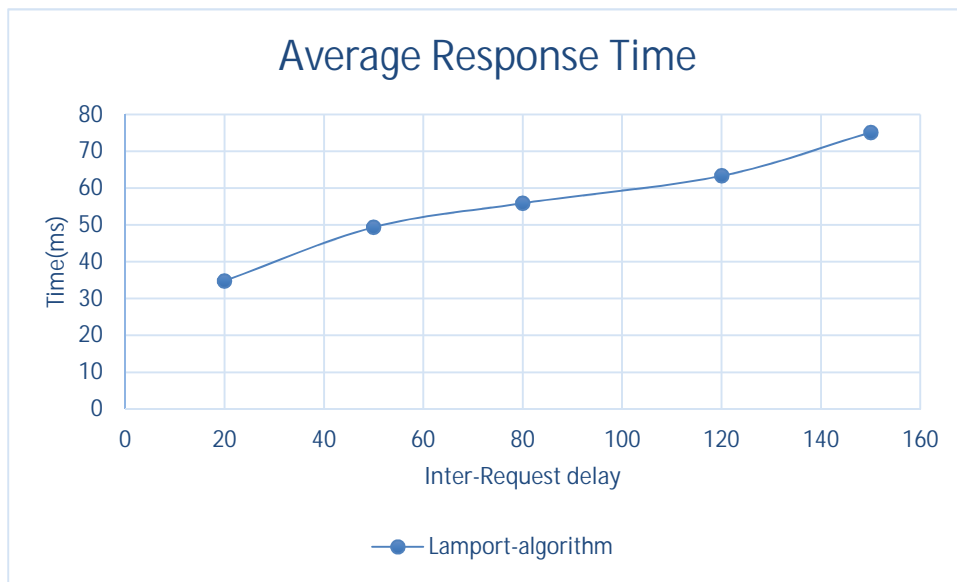
- Number of Nodes=5
- Inter Request Delay=20,50,80,120,150
- Critical Section execution Time=70
- Number of requests per each node=100

Message Complexity:



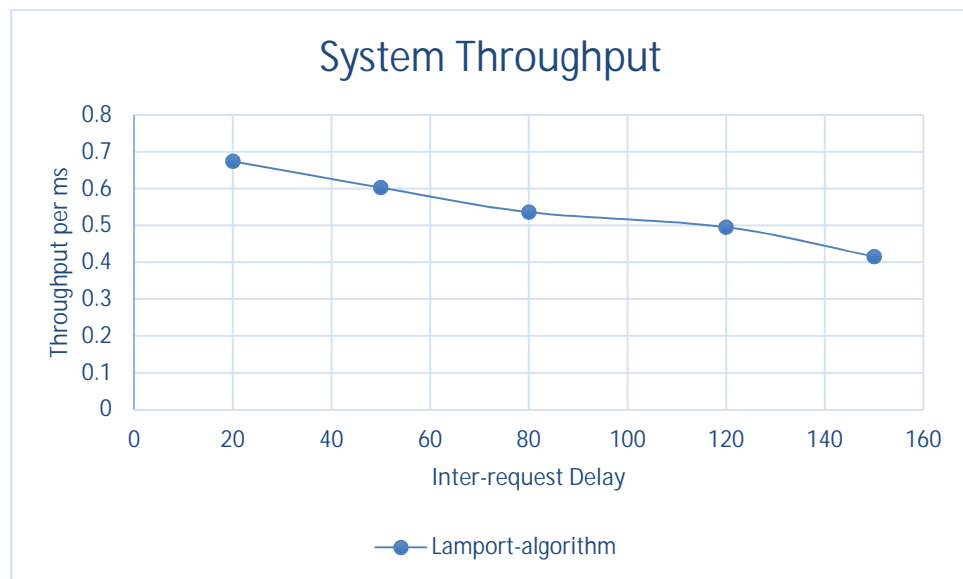
The above graph is for experiment with the message complexity and the Inter Request Delay. Message complexity is the number of messages required per Critical Section execution by a site. The number of messages involved for each request remains same for the system with Lamport algorithm for varying inter-request delays.

Average Response time:



The above graph is for experiment with the Inter Request Delay and the response time. Response Time is the time interval a request waits for its CS execution to be over after its request messages have been sent out. The average response time increases gradually with increase in inter-request delay.

System throughput



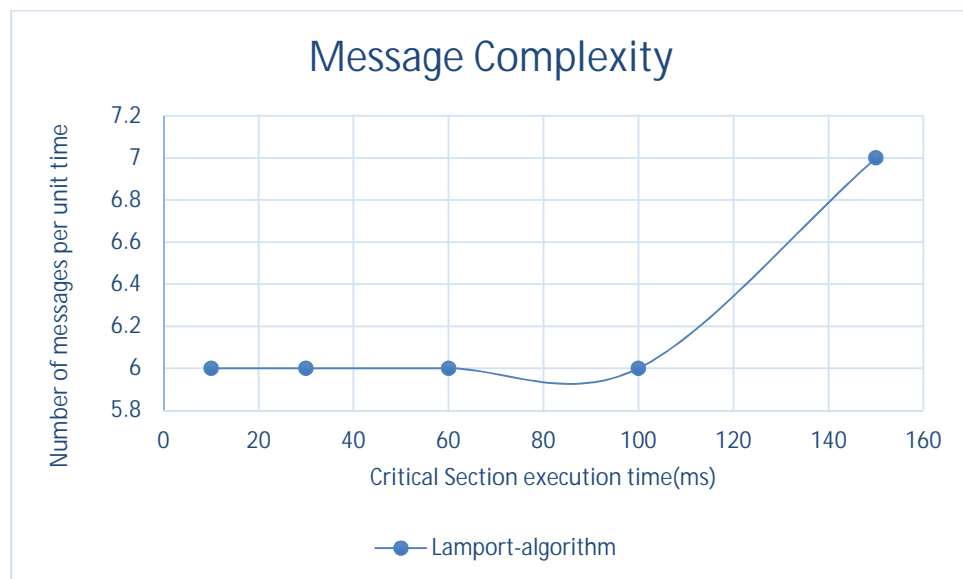
The above graph is for an experiment with the Inter Request Delay and system throughput. System Throughput is the rate at which the system executes requests for the CS. The system throughput reduces per increase in inter request delay.

Analysis by varying mean Critical execution time 'c'

Configuration:

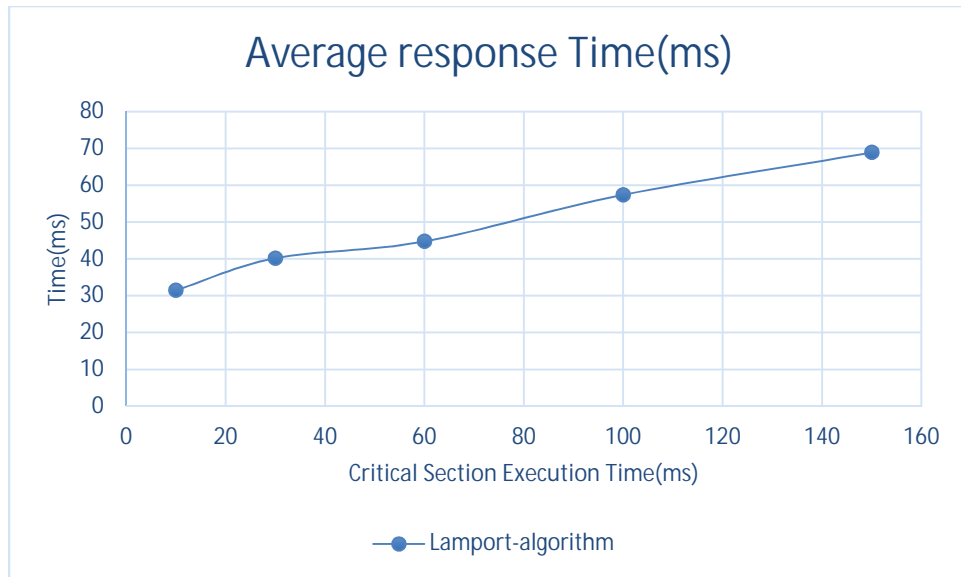
- Number of Nodes=5
- Inter Request Delay=20
- Critical Section execution Time=10,30,60,100,150
- Number of requests per each node=100

Message Complexity:



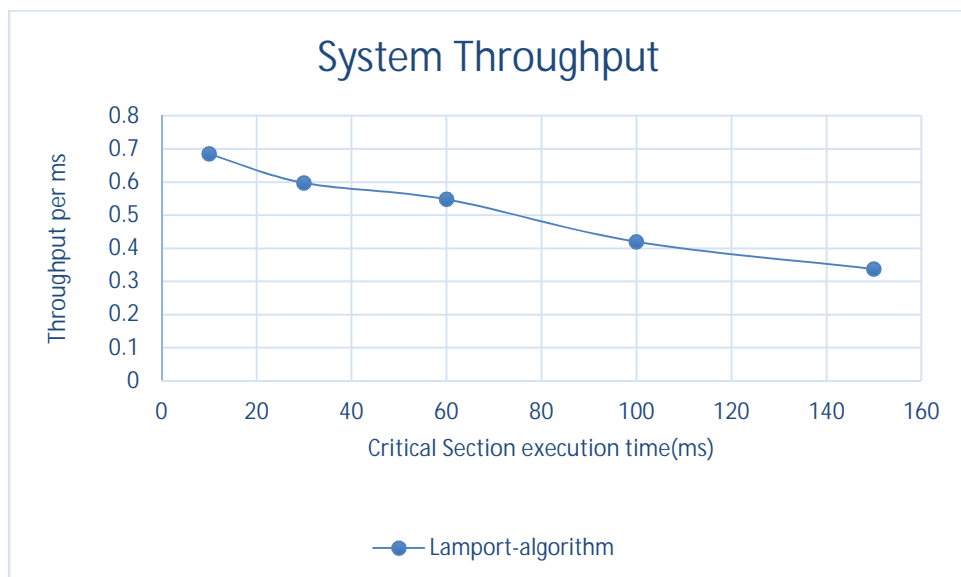
The above graph is for experiment with the Critical Section Execution Time and the Number of messages sent per unit time. Message complexity is the number of messages required per CS execution by a site. Initially the number of messages remains same but it gradually increases to 7 with increase in critical section execution time.

Average Response time:



The above graph is for experiment with the Critical Section Execution Time and the time in milliseconds. Response Time is the time interval a request waits for its CS execution to be over after its request messages have been sent out. Average response time increases with increase in CS execution time.

System throughput



The above graph is for experiment with the Inter Critical Section Execution time and the throughput. It is the rate at which the system executes requests for the Critical Section.

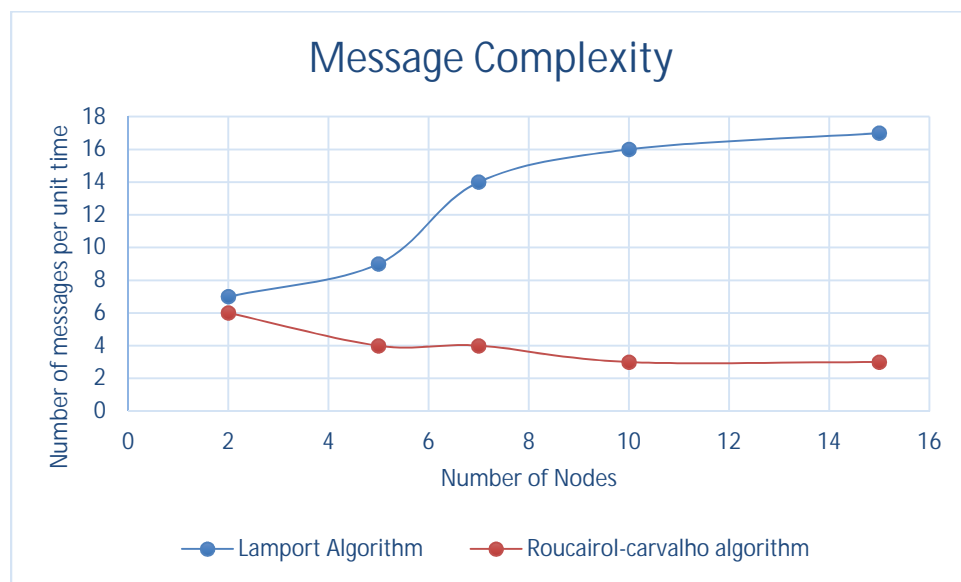
Experimental comparison of both mutual exclusion algorithms

For message complexity

In general, the Message complexity for Lamport Mutual Exclusion Algorithm is $3(N-1)$ in both heavy load and low load situations whereas for Roucairol Carvalho's Algorithm Message complexity is $2(N-1)$ in High load situations and 0 in Low Load situations.

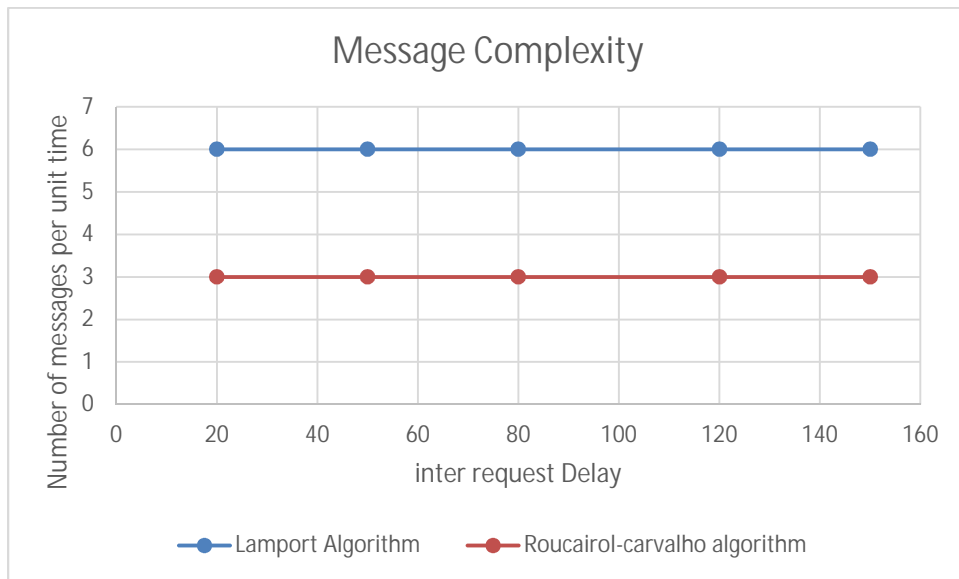
The below graphs are for experiment with the message complexity and the number of nodes for both Algorithms, Roucairol Carvalho and Lamport Mutual Exclusion Algorithm. Message complexity is the number of messages required per CS execution by a site.

i. By varying number of nodes



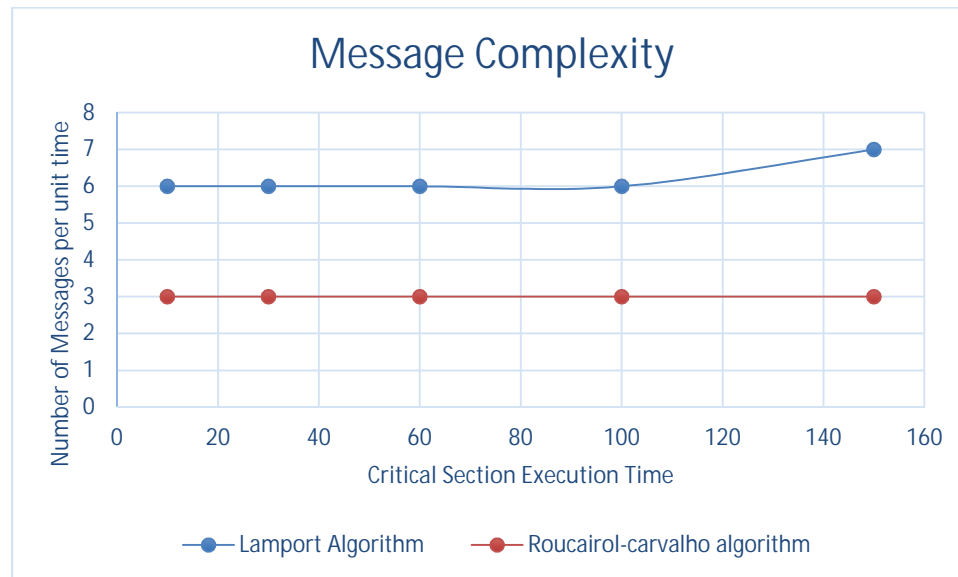
As the number of nodes increases the values for Message complexity steeply increases for Lamport Mutual Exclusion Algorithm. While the message complexity reduces and remains constant with increase in the number of nodes.

ii. By varying the mean inter-request delay



The message complexity remains constant for both algorithms but the message complexity for Lamport is higher than RC algorithm.

iii. By varying the mean critical section execution time



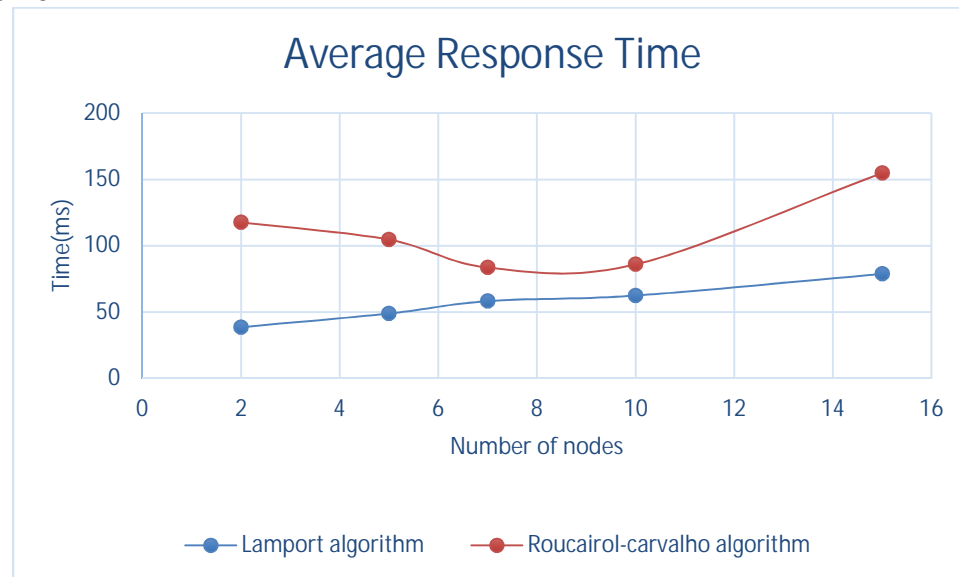
With the increasing critical section execution time the message complexity for Lamport Mutual exclusion increases. The message complexity almost remains same for both algorithms.

For Average response time

Response Time for the Lamport Mutual Exclusion and Roucairol Carvalho Algorithm is $(2T+E)$. Generally, the response time for Lamport Mutual Exclusion Algorithm is better than the Roucairol Carvalho Algorithm

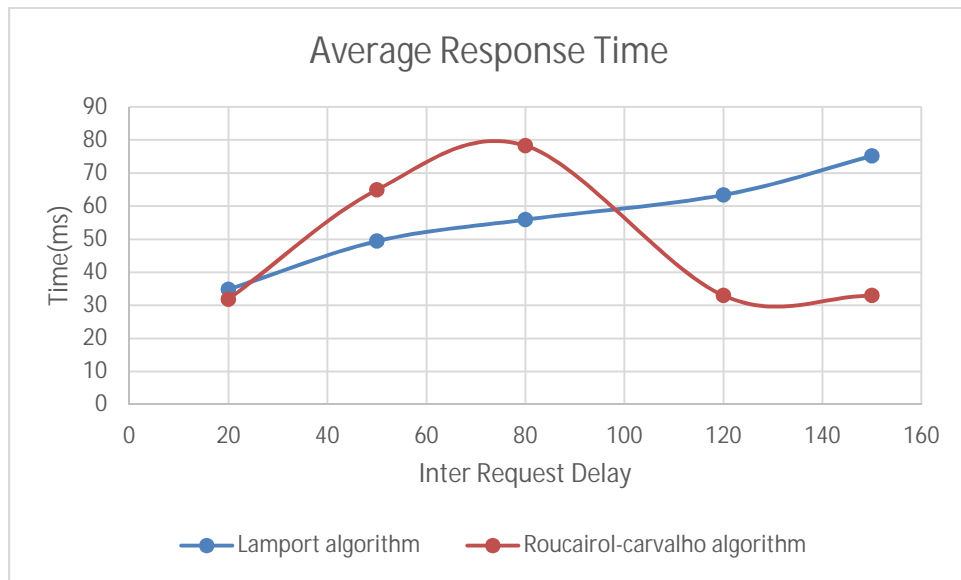
The below graphs are for experiment with the Response Time and the number of nodes. Response Time is the time interval a request waits for its CS execution to be over after its request messages have been sent out.

i. By varying number of nodes



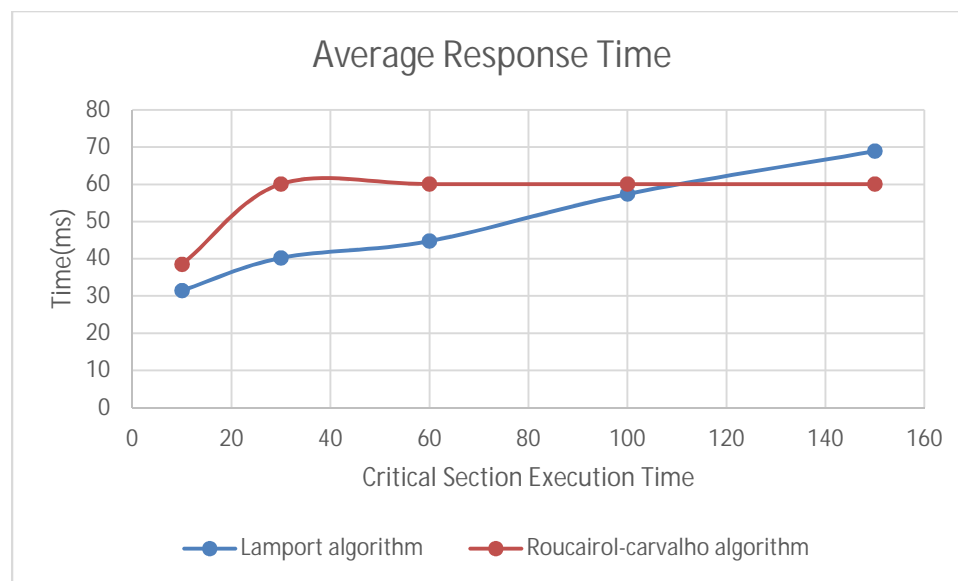
As the number of nodes increases the response time for both protocols increases. But the average response time for RC algorithm is higher than that for Lamport algorithm.

ii. By varying the mean inter-request delay



For Lamport mutual exclusion algorithm, as the Inter Request Delay Increases the response time also increases.

iii. By varying the mean critical section execution time

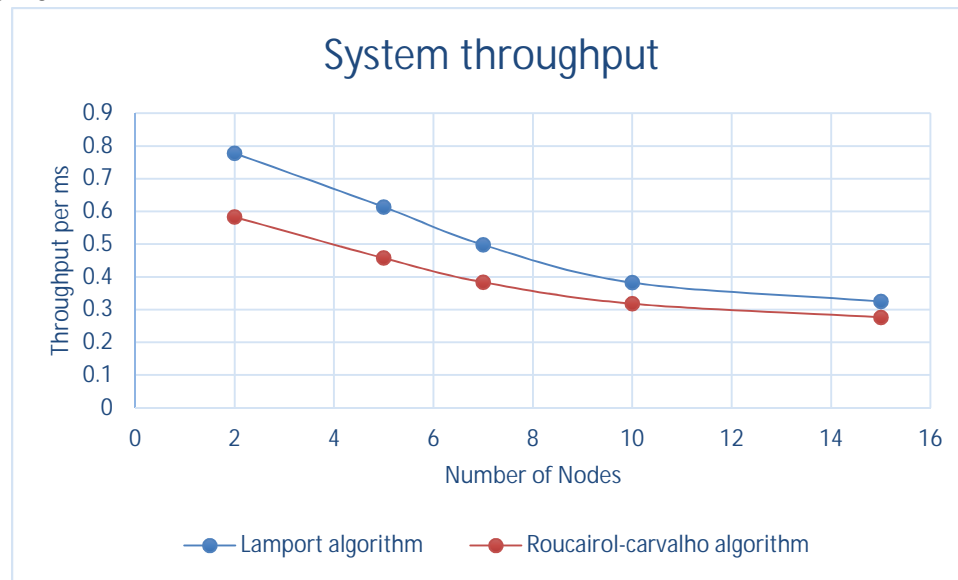


For both Lamport mutual exclusion algorithm and Roucairol Carvalho Algorithm as the Critical Section execution time increases the response time also increases, for Roucairol the response time after some point starts decreasing.

For System throughput

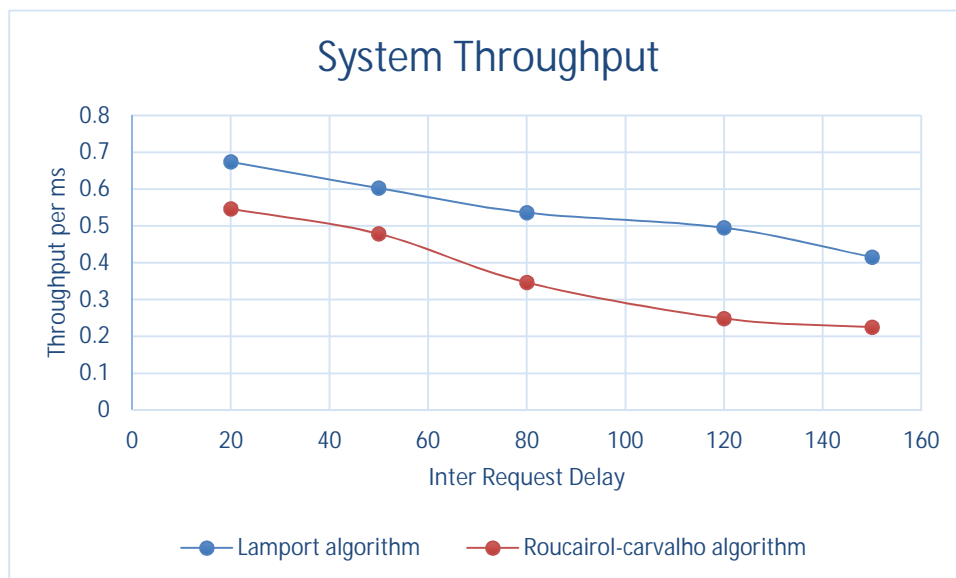
The rate at which the system executes requests for the CS. $\text{system throughput} = 1/(\text{SD} + E)$ where SD is the synchronization delay and E is the average critical section execution time.

i. By varying number of nodes



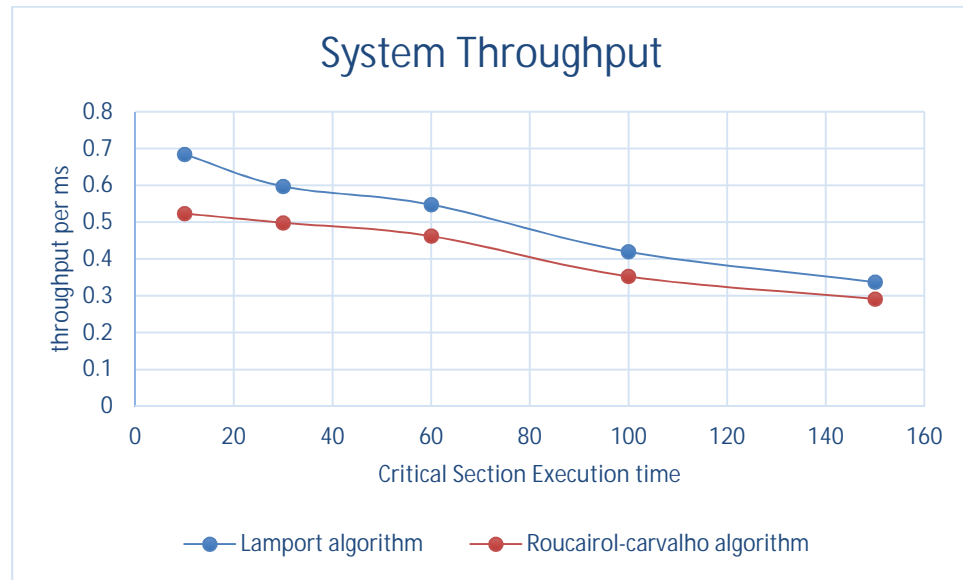
As the number of nodes increases the values for system throughput decreases. While for RC algorithm the throughput reduces greatly than for Lamport.

ii. By varying the mean inter-request delay



System throughput reduces for both protocols with increase in inter-request delays.

iii. By varying the mean critical section execution time



As the critical section execution time increases the values for System throughput decreases for both the algorithms.

Conclusion

In this Project the performance metrics of two algorithms, Roucairol Carvalho and Lamport Mutual Exclusion Algorithm are discussed, each kind of algorithm has its own varying characteristic and performance. A comparative study of their performance based on message complexity, Response Time and System throughput is done. The Roucairol and Carvalho's Algorithm works better for the performance Metric Message Complexity as compared to Lamport Mutual Exclusion Algorithm. The Response Time and System Throughput for Lamport Mutual Exclusion is better as compared to Roucairol and Carvalho's Algorithm.

References

- N. Lynch. Distributed Algorithms. Morgan Kaufman, 1996.
- H. Attiya and J. Welch. Distributed Computing. McGraw-Hill, 1998.
- M. Ben-Ari. Interactive Execution of Distributed Algorithms. ACM Journal of Educational Resources in Computing 1(2), 2001.
- S.Lodha, and A.Kshemkalyani, " A fair distributed mutual exclusion algorithm," IEEE Transactions on Parallel and Distributed Systems, vol. 11, no. 6, June 2000.
- S. Paydar, M. Naghibzadeh and A. Yavari, "A hybrid distributed mutual exclusion algorithm", in IEEE - International Conference on Engineering and Technology, 2006