

# CS5320 – Assignment #3

## Implementing Distributed Mutual Exclusion Algorithm

By Harsh Agarwal  
cs15btech11019

### Implementation Details

Each node is represented by a thread.  
Every node creates another thread for receiving messages at a port number.  
All nodes are running on different port numbers on localhost.  
The port number for every node identified using  $(PORT\_STARTING\_RANGE + nodeID)$ .

### Mutual Exclusion Property

#### Suzuki Kasami Algorithm

```
harsh@harsh-Insiptron-S554:~/btech/sem-7/Distributed-Computing/assignments/Distributed-Computing-Algorithms/Mutual-Exclusion/Suzuki-Kasami$ cat output.txt | grep -nE "enters|exits"
124:2 ENTERS CS for the 1st time at 1008234
133:2 EXITS CS for the 1st time at 1010037
143:1 ENTERS CS for the 1st time at 1010211
167:1 EXITS CS for the 1st time at 1015612
170:4 ENTERS CS for the 1st time at 1016043
220:4 EXITS CS for the 1st time at 1020701
229:5 ENTERS CS for the 1st time at 1020947
254:5 EXITS CS for the 1st time at 1033025
257:6 ENTERS CS for the 1st time at 1033190
300:6 EXITS CS for the 1st time at 1050638
322:7 ENTERS CS for the 1st time at 1067306
324:7 EXITS CS for the 1st time at 1067399
327:2 ENTERS CS for the 2st time at 1067555
347:2 EXITS CS for the 2st time at 1077654
371:3 ENTERS CS for the 1st time at 1099352
372:3 EXITS CS for the 1st time at 1100745
388:0 ENTERS CS for the 1st time at 1102291
391:0 EXITS CS for the 1st time at 1103501
398:1 ENTERS CS for the 2st time at 1103644
420:1 EXITS CS for the 2st time at 1111879
443:8 ENTERS CS for the 1st time at 1129652
444:8 EXITS CS for the 1st time at 1137739
467:4 ENTERS CS for the 2st time at 1143944
468:4 EXITS CS for the 2st time at 1149184
471:5 ENTERS CS for the 2st time at 1151942
479:5 EXITS CS for the 2st time at 1156448
486:9 ENTERS CS for the 1st time at 1156585
516:9 EXITS CS for the 1st time at 1173656
539:6 ENTERS CS for the 2st time at 1185783
540:6 EXITS CS for the 2st time at 1189686
548:7 ENTERS CS for the 2st time at 1197688
549:7 EXITS CS for the 2st time at 1197775
563:2 ENTERS CS for the 3st time at 1217289
577:2 EXITS CS for the 3st time at 1217834
592:3 ENTERS CS for the 2st time at 1220645
593:3 EXITS CS for the 2st time at 1222083
614:0 ENTERS CS for the 2st time at 1225659
616:0 EXITS CS for the 2st time at 1225733
624:1 ENTERS CS for the 3st time at 1225879
627:1 EXITS CS for the 3st time at 1225933
658:8 ENTERS CS for the 2st time at 1233789
684:8 EXITS CS for the 2st time at 1241652
```

All ENTER events are followed by an EXIT event before the next ENTER.  
Thus mutual exclusion is satisfied.

## Kerry Raymond Algorithm

```
harsh@harsh-Inspiron-5554:~/btech/sem-7/Distributed-Computing/assignments/Distributed-Computing-Algorithms/Mutual-Exclusion/Raymond-Tree$ cat output.txt | grep -lE "enters|exits"
17:1 ENTERS CS for the 1st time at 1002266
18:1 EXITS CS for the 1st time at 1002732
17:6 ENTERS CS for the 1st time at 1002970
20:6 EXITS CS for the 1st time at 1003037
32:6 ENTERS CS for the 2st time at 1003122
33:6 EXITS CS for the 2st time at 1003183
37:6 ENTERS CS for the 3st time at 1003266
38:6 EXITS CS for the 3st time at 1003326
42:6 ENTERS CS for the 4st time at 1003407
73:6 EXITS CS for the 4st time at 2003494
82:8 ENTERS CS for the 1st time at 2030472
83:8 EXITS CS for the 1st time at 2031949
88:0 ENTERS CS for the 1st time at 2032153
92:0 EXITS CS for the 1st time at 2037042
100:1 ENTERS CS for the 2st time at 2039162
101:1 EXITS CS for the 2st time at 2040719
113:9 ENTERS CS for the 1st time at 2042508
114:9 EXITS CS for the 1st time at 2042582
120:5 ENTERS CS for the 1st time at 2045008
127:5 EXITS CS for the 1st time at 2047198
137:2 ENTERS CS for the 1st time at 2052613
140:2 EXITS CS for the 1st time at 2059338
152:3 ENTERS CS for the 1st time at 2070951
153:3 EXITS CS for the 1st time at 2076995
162:4 ENTERS CS for the 1st time at 2081179
166:4 EXITS CS for the 1st time at 2081244
170:7 ENTERS CS for the 1st time at 2082324
179:7 EXITS CS for the 1st time at 2086611
186:8 ENTERS CS for the 2st time at 2086880
187:8 EXITS CS for the 2st time at 2090999
200:0 ENTERS CS for the 2st time at 2109344
201:0 EXITS CS for the 2st time at 2113814
209:1 ENTERS CS for the 3st time at 2121708
210:1 EXITS CS for the 3st time at 2126881
222:9 ENTERS CS for the 2st time at 2148886
223:9 EXITS CS for the 2st time at 2149136
235:5 ENTERS CS for the 2st time at 2157567
236:5 EXITS CS for the 2st time at 2157656
240:2 ENTERS CS for the 2st time at 2167382
240:2 EXITS CS for the 2st time at 2169095
261:3 ENTERS CS for the 2st time at 2178692
262:3 EXITS CS for the 2st time at 2178765
```

All ENTER events are followed by an EXIT event before the next ENTER.  
Thus mutual exclusion is satisfied.

## Graphs Section

Topology - Complete Graph

Number of CS executions - 8

Initial Node With Token - 0

Alpha - 2

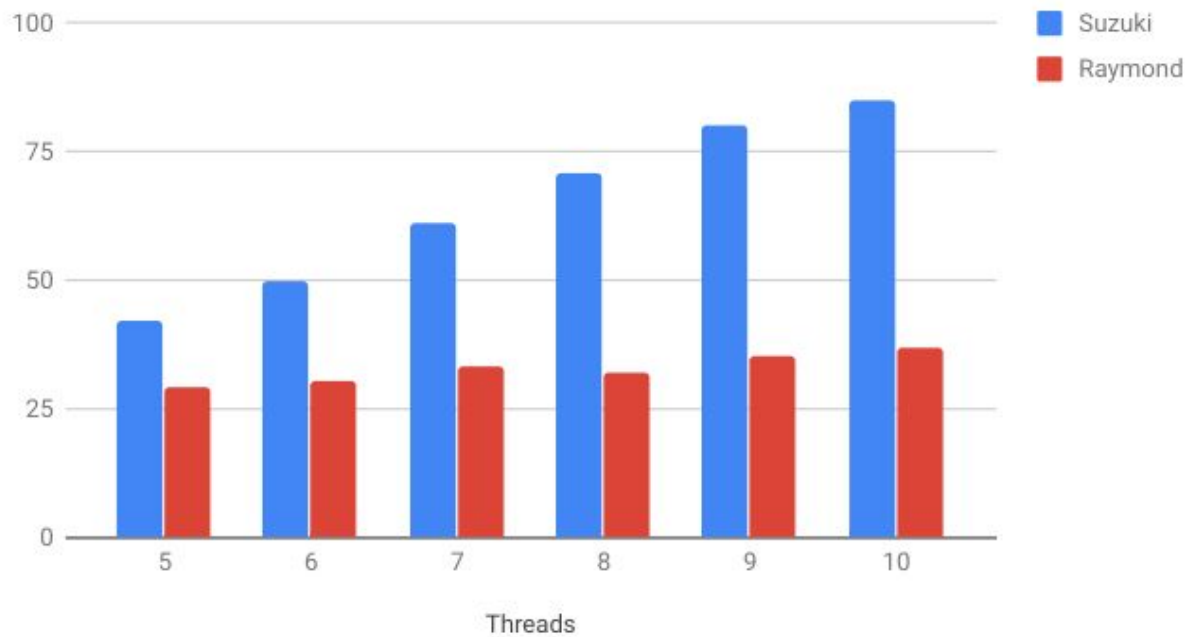
Beta - 1

### Average Message Complexity

#### READINGS

Threads	Suzuki	Raymond
5	42	29.2
6	50	30.6667
7	61	33.1429
8	71	32.25
9	80	35.3333
10	85	36.8

## Average Message Complexity

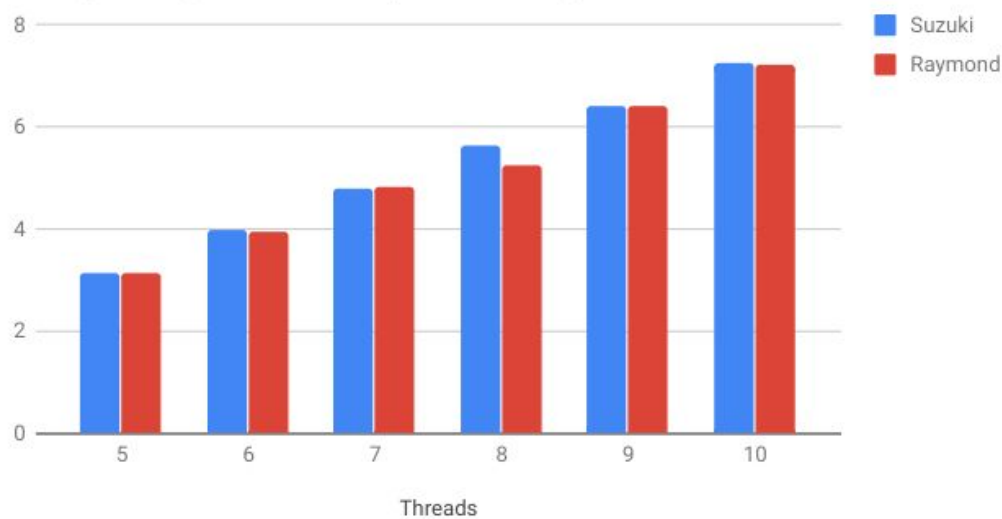


## Average Response Time (in seconds)

### READINGS

Threads	Suzuki	Raymond
5	3.143	3.152
6	3.972	3.964
7	4.797	4.825
8	5.639	5.249
9	6.422	6.410
10	7.275	7.226

Average Response Time (in seconds)



## OBSERVATIONS

- Raymond Algorithm has much lesser message complexity than Suzuki because Raymond sends REQUEST & PRIVILEGE messages just to its neighbors in a spanning tree.  
The added message complexity of forwarding PRIVILEGE message along spanning tree edges in Raymond is superseded by the benefit of sending REQUEST just to HOLDER node. (There are much more REQUEST messages in the system than PRIVILEGE messages.)

But Suzuki Algorithm sends REQUEST messages to all other nodes. Due to this, a lot of messages need to be sent.

- On dividing Raymond message complexity with the number of iterations, we get the average message complexity per CS execution to be almost equal to 4.  
This matches with the fact explained in book “as the number of nodes requesting the privilege increases, the number of messages exchanged per critical section entry decreases. In fact, it requires the exchange of only four messages per CS execution.”  
That’s why the message complexity increases very slowly with the

number of nodes.

- Suzuki algorithm message complexity increases at a faster rate with the number of nodes. This is because more REQUEST messages need to be sent along new  $N-1$  edges.
- Response times are almost the same for both the algorithms. It increases with time.  
This is because:  
For Raymond, more edges need to be traversed to pass on the token from a privileged node to a requesting node.  
For Suzuki, more REQUEST messages need to be sent along new  $N-1$  edges.