

Name: Subasiravan.V

Class - CSE-B

Reg No: 205001085

## Lamport's Distributed Mutex Algorithm

$P_1 \rightarrow P_3 \parallel P_2 \parallel P_1 \rightarrow P_2 \parallel P_1$

$P_1$   $(1, P_1) (2, P_1) (3, P_1)$   
 $(24, P_1) (24, P_2)$

Req (1, P<sub>1</sub>, {P<sub>2</sub>, P<sub>3</sub>})  
(Add to queue)

Rep (3, P<sub>2</sub>, P<sub>1</sub>)  
Rep (3, P<sub>3</sub>, P<sub>1</sub>)

Rel (6, P<sub>1</sub>, {P<sub>1</sub>, P<sub>2</sub>})  
(remove)

Req (8, P<sub>1</sub>, {P<sub>2</sub>, P<sub>3</sub>})

Req (8, P<sub>2</sub>, P<sub>1</sub>)

Req (8, P<sub>3</sub>, P<sub>1</sub>)

Rep (11, P<sub>2</sub>, P<sub>1</sub>)

Rep (11, P<sub>3</sub>, P<sub>1</sub>)

CS

Rep (15, P<sub>1</sub>, P<sub>3</sub>)  
(remove)

CS

Rel (18, P<sub>2</sub>, P<sub>1</sub>)  
(remove)

Rel (P<sub>1</sub>, P<sub>3</sub>, P<sub>2</sub>)  
(remove)

Rep (26, P<sub>2</sub>, P<sub>1</sub>)

Rep (26, P<sub>3</sub>, P<sub>1</sub>)

CS

$P_2$   $(1, P_1) (8, P_2)$   
 $(24, P_1) (24, P_2)$

Req (1, P<sub>1</sub>, P<sub>2</sub>)  
(add)

Rep (3, P<sub>2</sub>, P<sub>1</sub>)

Rel (6, P<sub>1</sub>, P<sub>2</sub>)  
(remove)

Req (8, P<sub>2</sub>, {P<sub>1</sub>, P<sub>3</sub>})  
(add)

Req (8, P<sub>1</sub>, P<sub>2</sub>) (add)

Req (11, P<sub>2</sub>, P<sub>1</sub>)

Rel (15, P<sub>1</sub>, P<sub>2</sub>)  
(remove)

Rep (18, P<sub>2</sub>, {P<sub>1</sub>, P<sub>3</sub>})  
(remove)

Rep (26, P<sub>2</sub>, P<sub>1</sub>)

$P_3$   $(1, P_1) (2, P_1)$   
 $(24, P_1) (24, P_2)$

Req (1, P<sub>1</sub>, P<sub>3</sub>)

Rep (3, P<sub>3</sub>, P<sub>1</sub>)  
(add)

Rel (6, P<sub>1</sub>, P<sub>3</sub>) (rm)

Req (8, P<sub>3</sub>, {P<sub>1</sub>, P<sub>2</sub>})

Req (8, P<sub>1</sub>, P<sub>3</sub>) (add)

Rep (11, P<sub>3</sub>, P<sub>1</sub>)

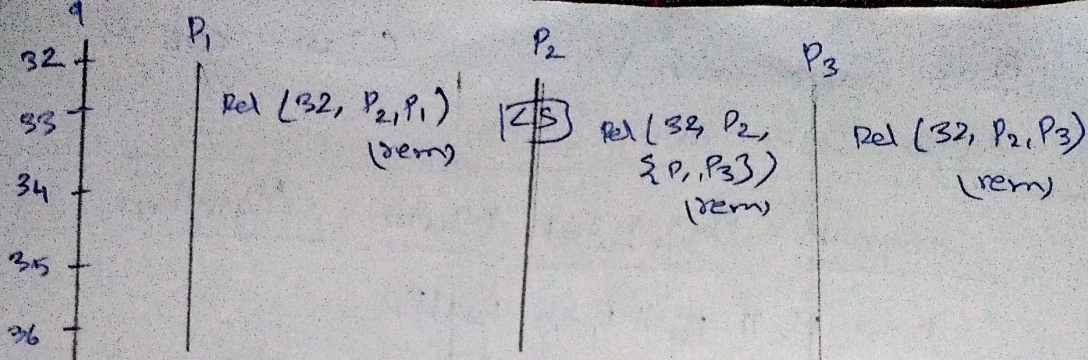
Rel (15, P<sub>1</sub>, P<sub>3</sub>)  
(remove)

Rel (18, P<sub>2</sub>, P<sub>1</sub>)  
(rm)

Rel (21, P<sub>3</sub>, {P<sub>1</sub>, P<sub>2</sub>})  
(rm)

Rep (26, P<sub>3</sub>, P<sub>1</sub>)

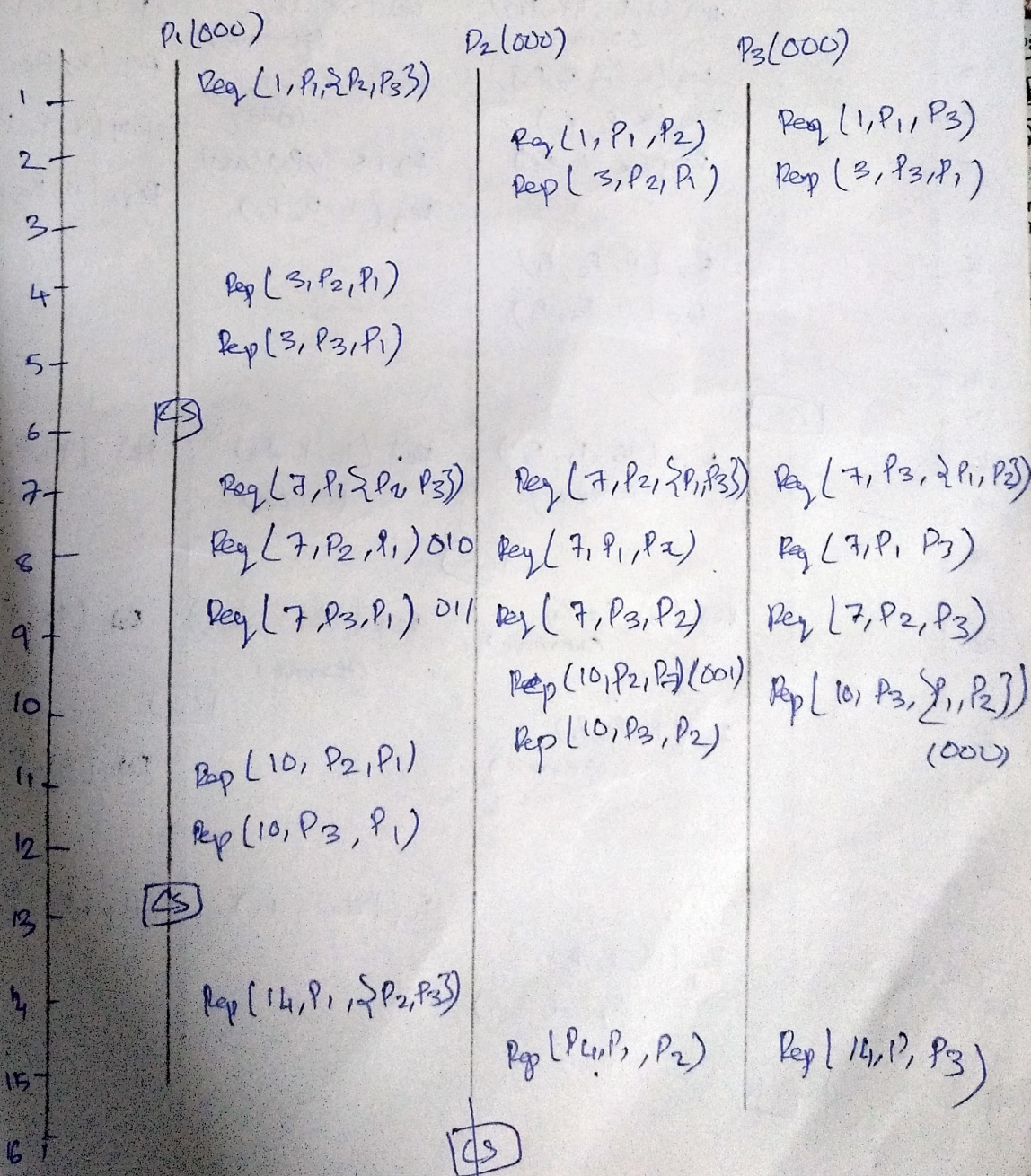




Lamport's algorithm achieves mutual exclusion by sending req, rep and rel messages accordingly.

### Rechart Agarwala Algorithm

P<sub>1</sub> → P<sub>3</sub> || P<sub>2</sub> || P<sub>1</sub> → P<sub>2</sub> || P<sub>1</sub>      RD = 000





18			
19		Rep (18, P <sub>2</sub> , P <sub>3</sub> )	Rep (18, P <sub>2</sub> , P <sub>3</sub> )
20			CS
21			
22	Req (22, P <sub>1</sub> , {P <sub>2</sub> , P <sub>3</sub> })	Req (24, P <sub>2</sub> , {P <sub>1</sub> , P <sub>3</sub> })	
23	Req (22, P <sub>2</sub> , P <sub>1</sub> ) (D10)	Req (22, P <sub>1</sub> , P <sub>2</sub> )	Req (22, P <sub>1</sub> , P <sub>3</sub> )
24		Rep (24, P <sub>2</sub> , P <sub>1</sub> )	Req (22, P <sub>2</sub> , P <sub>3</sub> )
25	Rep (24, P <sub>2</sub> , P <sub>1</sub> )		Rep (24, P <sub>3</sub> , {P <sub>1</sub> , P <sub>2</sub> })
26	Rep (24, P <sub>3</sub> , P <sub>1</sub> )		
27	CS		
28	Rep (28, P <sub>1</sub> , P <sub>2</sub> ) (D00)	Rep (26, P <sub>1</sub> , P <sub>2</sub> )	
29		CS	
30			

Richard Agrawala algorithm reduces number of messages sent (request, reply) by checking which replies have been deferred.