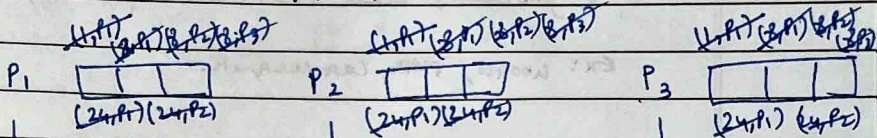


LAMPORT'S DISTRIBUTED MUTEX ALGORITHM:

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



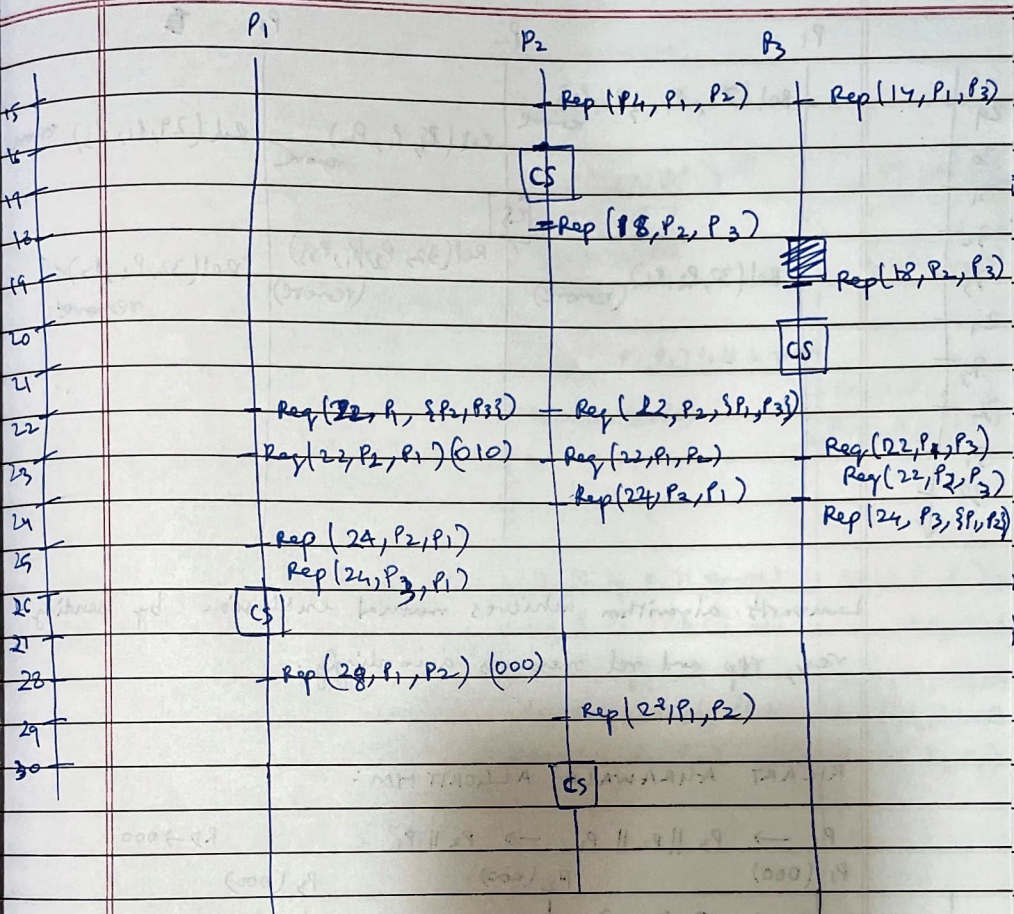
	P ₁	P ₂	P ₃
1	Req(1, P ₁ , {P ₂ , P ₃ }) (Add to queue)		
2		Req(1, P ₁ , P ₂) (add to queue)	
3			Req(1, P ₁ , P ₃)
4	Rep(3, P ₂ , P ₁) Rep(3, P ₃ , P ₁)	Rep(3, P ₂ , P ₁)	Rep(3, P ₃ , P ₁) (add)
5	CS		
6			
7	Rel(6, P ₁ , {P ₁ , P ₂ }) (remove)	Rel(6, P ₁ , P ₂) (remove)	Rel(6, P ₁ , P ₂) (rm)
8	Req(8, P ₁ , {P ₂ , P ₃ }) (add)	Rel(8, P ₂ , {P ₁ , P ₃ }) add	Req(8, P ₃ , {P ₁ , P ₂ })
9	Req(8, P ₂ , P ₁) (add) Req(8, P ₃ , P ₁) (add)	Req(8, P ₂ , P ₂) (add) Req(8, P ₃ , P ₂) (add)	Req(8, P ₁ , P ₃) (add) Req(8, P ₂ , P ₃) add
10			
11		Rep(11, P ₂ , P ₁)	Rep(11, P ₃ , P ₁)
12	Rep(11, P ₂ , P ₁)		
13	Rep(11, P ₃ , P ₁)		
14			
15	CS		
16	Rel(15, P ₁ , {P ₂ , P ₃ }) (remove)	Rel(15, P ₁ , P ₂) remove	Rel(15, P ₁ , P ₃) remove
17		CS	
18		Rel(18, P ₂ , {P ₁ , P ₃ }) remove	
19	Rel(18, P ₂ , P ₁) remove		Rel(18, P ₂ , P ₁) remove
20			
21			CS
22	Rel(21, P ₃ , P ₂) remove	Rel(21, P ₃ , P ₂) remove	
23			Rel(21, P ₃ , {P ₁ , P ₂ }) remove
24	Req(24, P ₁ , {P ₂ , P ₃ }) add	Req(24, P ₂ , {P ₁ , P ₃ }) add	Req(24, P ₁ , P ₃) add
25	Req(24, P ₂ , P ₁) add	Req(24, P ₁ , P ₂) add	Req(24, P ₂ , P ₃) add
26		Rep(26, P ₂ , P ₁)	Rep(26, P ₃ , P ₁) add
27	Rep(26, P ₂ , P ₁) Rep(26, P ₃ , P ₁)		
28	CS		

	P ₁	P ₂	P ₃	Σ
29	Rel(29, P ₁ , {P ₂ , P ₃ }) remove			
30		Rel(29, P ₁ , P ₂) remove	Rel(29, P ₁ , P ₃) remove	
31				
32		CS		
33	Rel(32, P ₂ , P ₁) (remove)	Rel(32, P ₂ , {P ₁ , P ₃ }) (remove)	Rel(32, P ₂ , P ₃) remove	
34				
35				
36				
37				
38				
↓				

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

RICART AGRAWALA ALGORITHM:

	$P_1 \rightarrow P_3 \parallel P_2 \parallel P_1 \rightarrow P_2 \parallel P_1$	$RD \rightarrow 000$	
	$P_1(000)$	$P_2(000)$	$P_3(000)$
	Req(1, P_1 , $\{P_2, P_3\}$)		
		Req(1, P_1 , P_2)	Req(1, P_1 , P_3)
		Rep(3, P_2 , P_1)	Rep(3, P_3 , P_1)
	Rep(3, P_2 , P_1)		
	Rep(3, P_3 , P_1)		
	CS		
	Req(7, P_1 , $\{P_2, P_3\}$)	Req(7, P_2 , $\{P_1, P_3\}$)	Req(7, P_3 , $\{P_1, P_2\}$)
	Req(7, P_2 , P_1) 010	Req(7, P_1 , P_2)	Req(7, P_1 , P_3)
	Req(7, P_3 , P_1) 011	Req(7, P_3 , P_2)	Req(7, P_2 , P_3)
		Rep(10, P_2 , P_1) (001)	Rep(10, P_3 , $\{P_1, P_2\}$) (000)
	Rep(10, P_2 , P_1)	Rep(10, P_3 , P_2)	Rep(10, P_3, P_2)
	Rep(10, P_3 , P_1)		
	CS		
	Req(14, P_1 , $\{P_2, P_3\}$)		
	Rep(14, P_1 , $\{P_2, P_3\}$)		



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