

Ricart Agrawala & Maekawa's Algorithms

**Coulouris, Dollimore, Kindberg and Blair
Distributed Systems:
Concepts and Design**

Edition 5, © Addison-Wesley 2012

Ricart and Agrawala's algorithm

On initialization

state := RELEASED;

To enter the section

state := WANTED;

Multicast *request* to all processes;

T := request's timestamp;

Wait until (number of replies received = ($N - 1$));

state := HELD;

} request processing deferred here

On receipt of a request $\langle T_i, p_i \rangle$ at p_j ($i \neq j$)

if (*state* = HELD or (*state* = WANTED and $(T, p_j) < (T_i, p_i)$))

then

 queue *request* from p_i without replying;

else

 reply immediately to p_i ;

end if

To exit the critical section

state := RELEASED;

reply to any queued requests;

Maekawa's algorithm

On initialization

state := RELEASED;

voted := FALSE;

For p_i to enter the critical section

state := WANTED;

Multicast *request* to all processes in V_i ;

Wait until (number of replies received = K);

state := HELD;

On receipt of a request from p_i at p_j

if (*state* = HELD or *voted* = TRUE)

then

 queue *request* from p_i without replying;

else

 send *reply* to p_i ;

voted := TRUE;

end if

For p_i to exit the critical section

state := RELEASED;

Multicast *release* to all processes in V_i ;

On receipt of a release from p_i at p_j

if (queue of requests is non-empty)

then

 remove head of queue – from p_k , say;

 send *reply* to p_k ;

voted := TRUE;

else

voted := FALSE;

end if