Figure 15.7  Model for Mutual Exclusion Problem in Distributed Process Management
Figure 15.8  Example of Operation of Timestamping Algorithm

\[ a, x, b, j \]
Figure 15.9  Another Example of Operation of Timestamping Algorithm

Only logical order is maintained

You must wait until every system receives all the messages.
Figure 14.10  State Diagram for Algorithm in [RICA81]
(a) First Part

if (token_present)
    
    clock++;
    broadcast (Request, clock, i);
    wait (access, token);
    token_present = true;

    token_held = true;
    <critical section>;
    token[i] = clock;
    */ Postlude */
    token_held = false;
    for (int j = i + 1; j < n; j++)
        if (request(j) > token[j] && token_present)
            
            token_present = false;
            send (access, token[j]);

        else if (i = i + 1; j++)
            if (request(j) > token[j] && token_present)
                
                token_present = false;
                send (access, token[j]);

(b) Second Part

if (received (Request, k))

    request (j) = max(request(j), k);
if (token_present && token_held)
    <text of Postlude>;

Notation
send (j, access, token)   end message of type access, with token, by process j
broadcast (request, clock, i) send message from process i of type request, with time-stamp clock, to all other processes
received (request, t, j)  receive message from process j of type request, with time-stamp t

Figure 15.11  Token-Passing Algorithm (for process P_i)