Time, Clocks, and the Ordering of Events in a Distributed System

Leslie Lamport

Presented by Mark Colbert

Overview

- Why Order?
- Partial Ordering
- Logical (Lamport) Clocks
- Mutual Exclusion
- Physical Clocks
- References

Why Order?

- Distributed Systems
  - Mutual Exclusion
  - Database Serialization

Partial Ordering

- usually we order by physical time
  - cannot perfectly synchronize clocks
- a “happened before” b
- what if b “happened before” a too?
  - concurrent

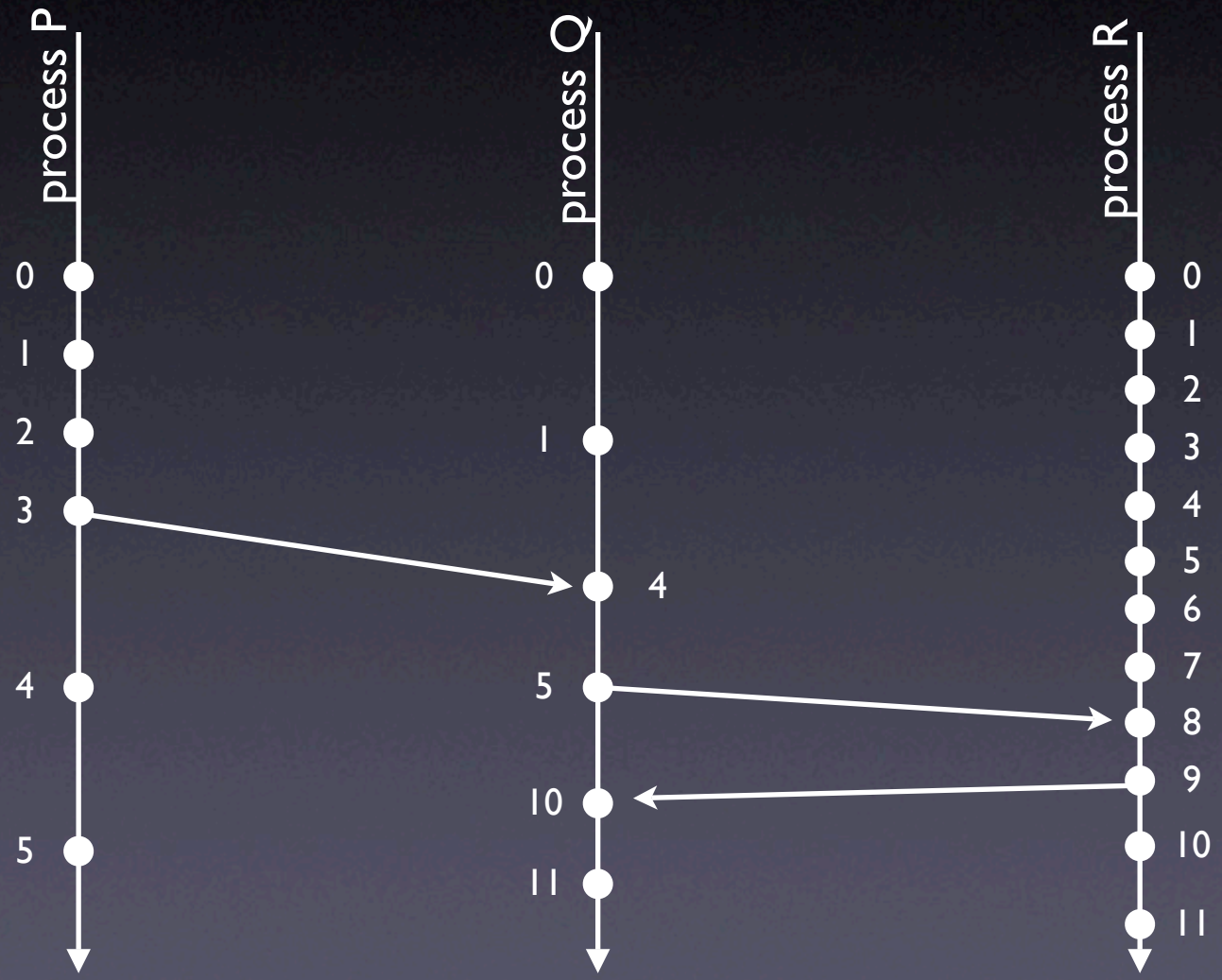
Logical (Lamport) Clocks

- use notion of partial ordering to find an arbitrary total ordering
- logical clocks do not require any basis in physical time, just a counter
- clock ticks represented by passing events
- processes send messages to one another with timestamps (current time for process)

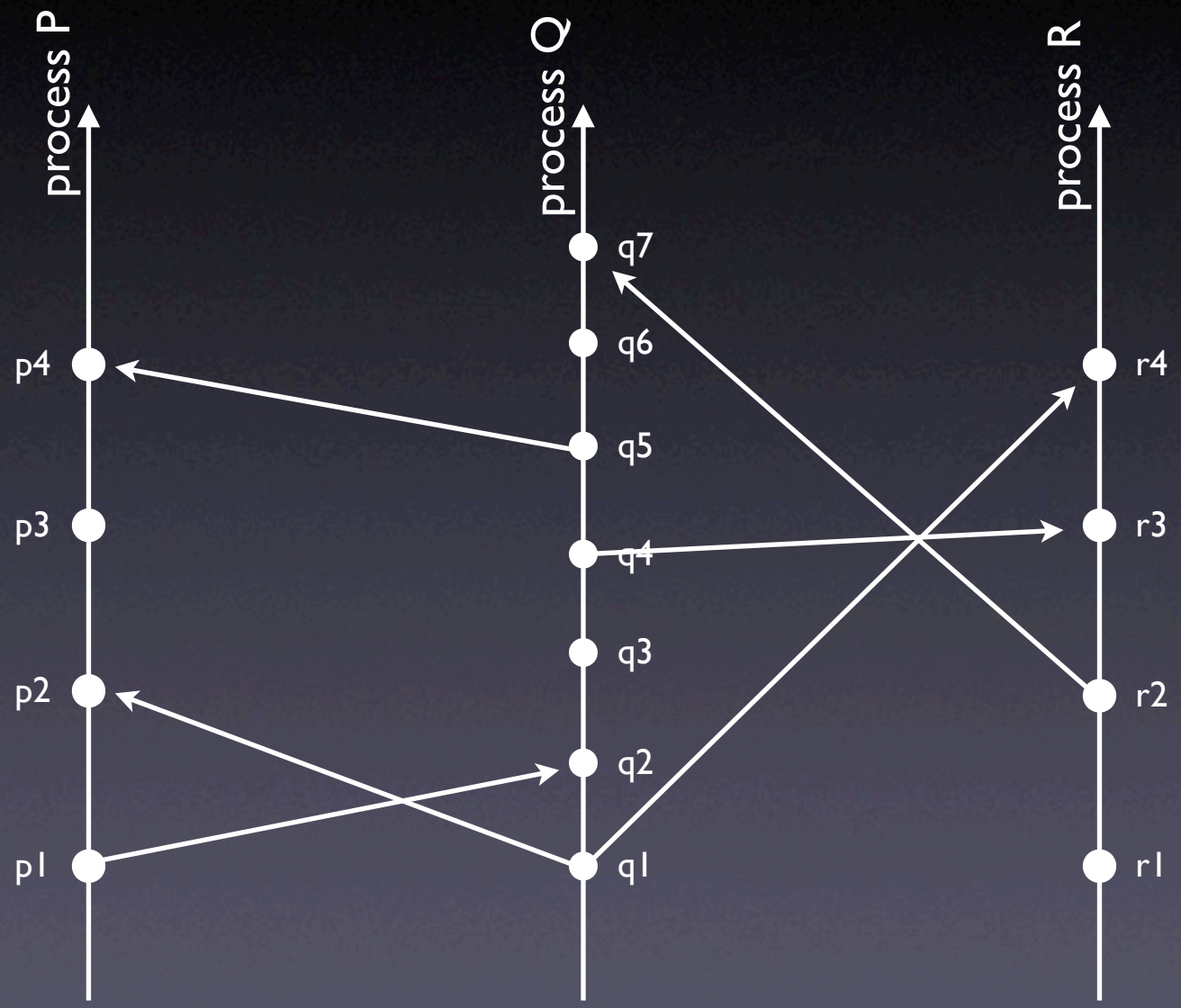
Logical (Lamport) Clocks

- rules for logical clocks
  - each process increments its counter (clock) after every event
  - if an event is a message, set the counter, c, to:
    c = max(timestamp + 1, c + 1)

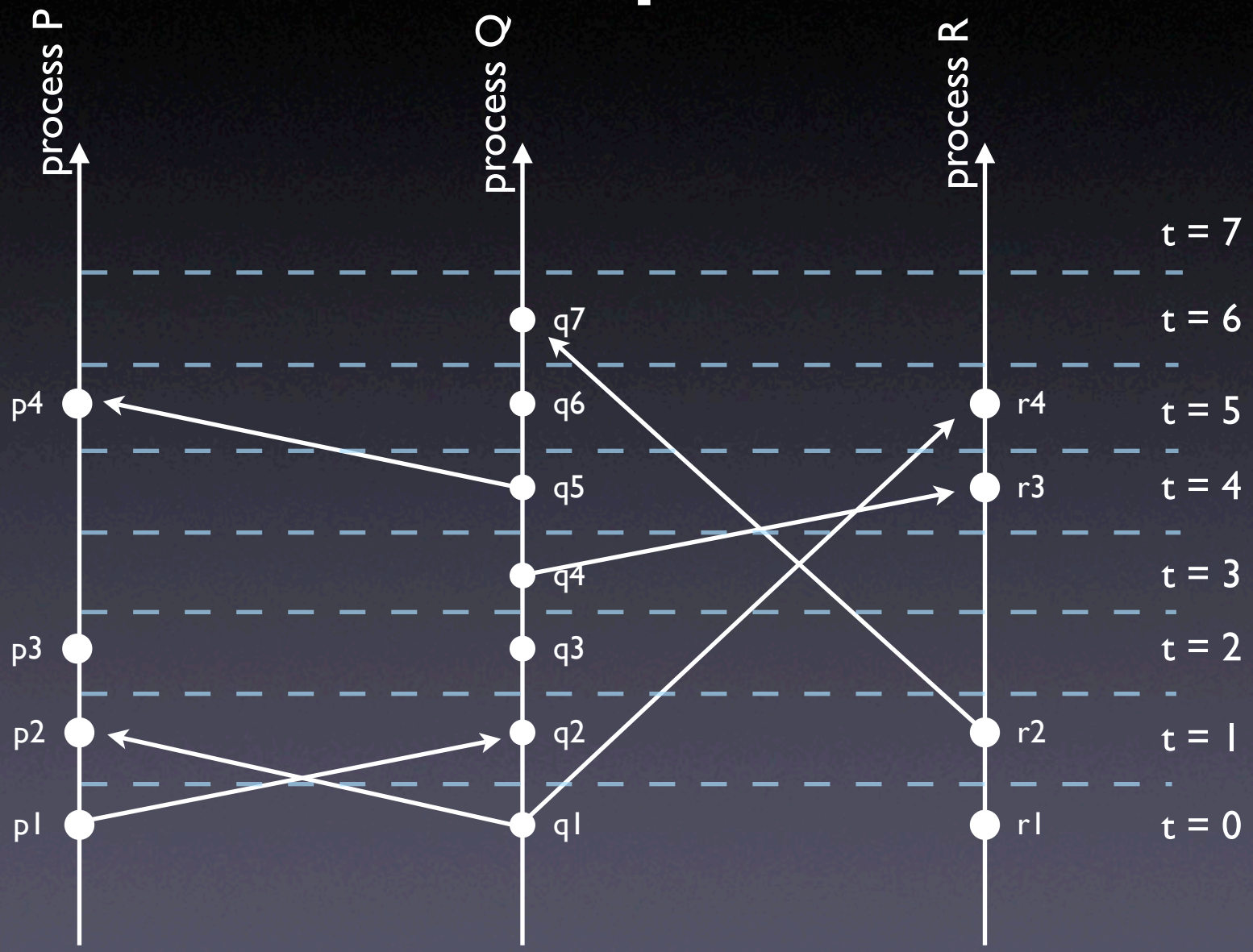
Using Logical Clocks



Example


process P
process Q
process R
p4
p3
p2
p1
q7
q6
q5
q4
q3
q2
q1
r4
r3
r2
r1

Example

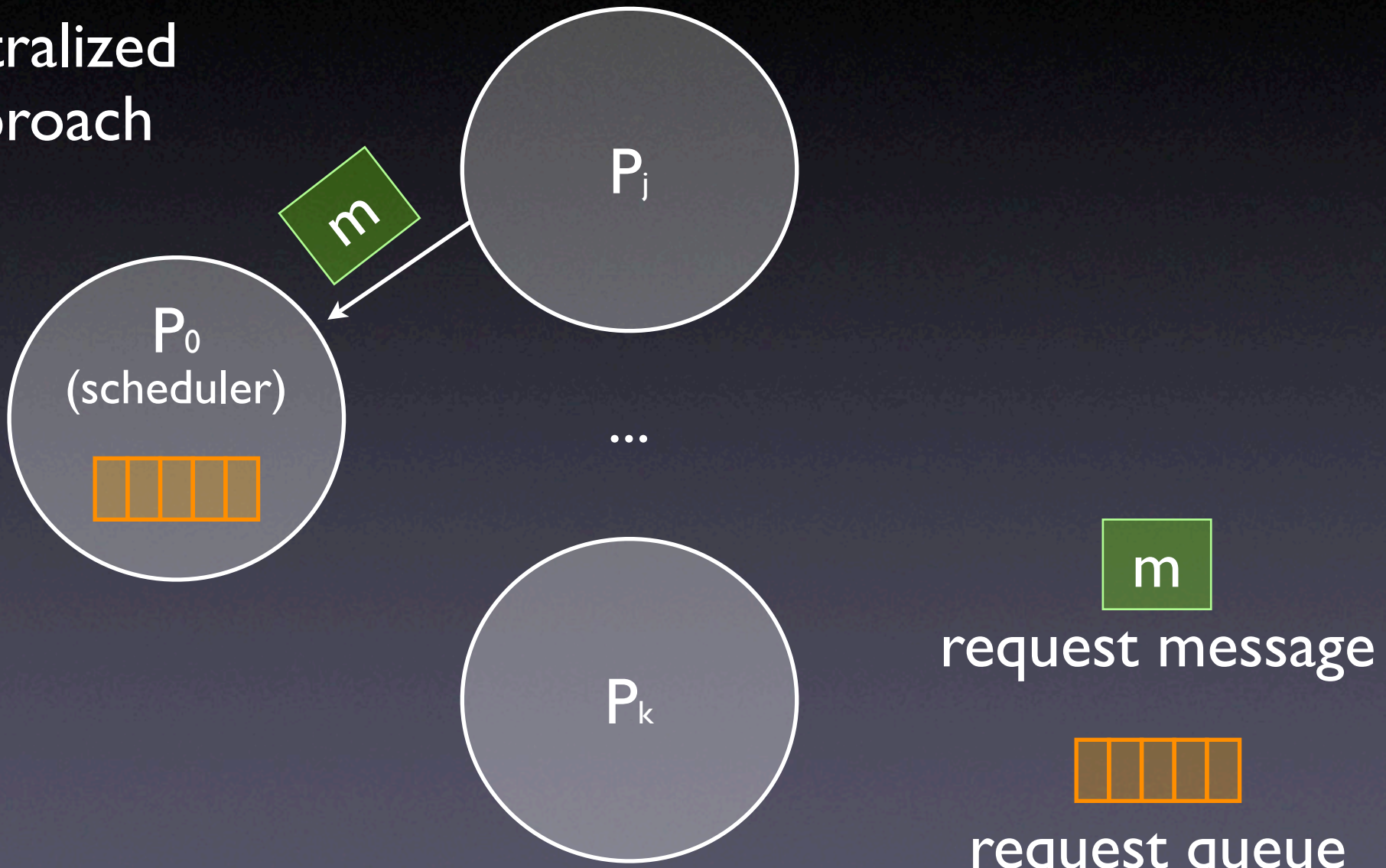

process P
process Q
process R
t = 7
t = 6
t = 5
t = 4
t = 3
t = 2
t = 1
t = 0
p4
p3
p2
p1
q7
q6
q5
q4
q3
q2
q1
r4
r3
r2
r1

Mutual Exclusion

- Shared resource between processes
- Requirements
  - A process must release a resource before another can obtain
  - Different requests must be granted in the order they were requested
  - Every process which is granted the resource must release it

Mutual Exclusion

Centralized Approach

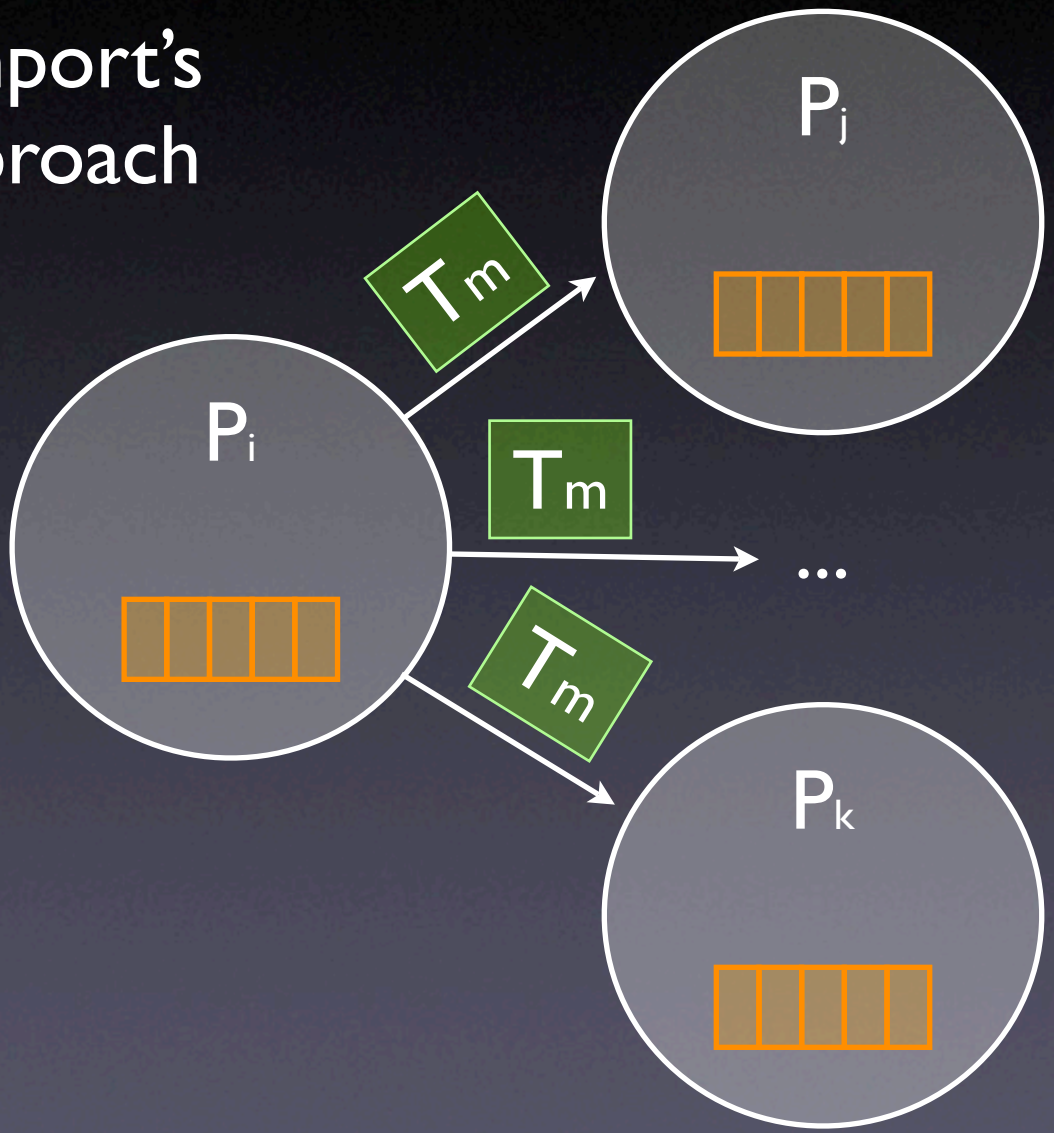


Mutual Exclusion

- Centralized Approach
  - does not guarantee order
    - suppose P_j sends a request to P_0 then sends a message to P_k
    - upon receiving the message, P_k sends a request that arrives before P_j's request
    - order is broken

Mutual Exclusion

Lamport's Approach



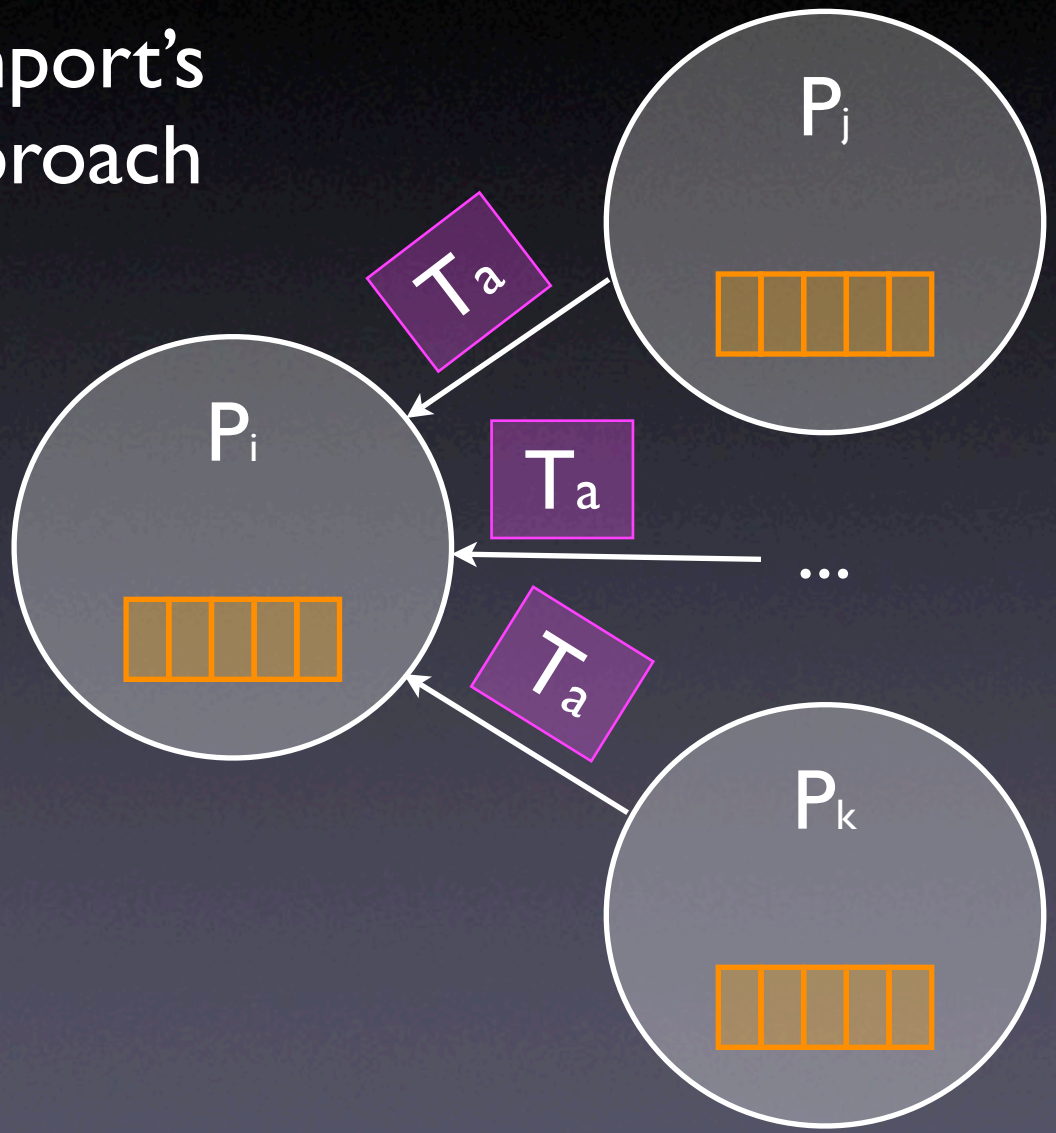
T_m

time stamped request message

request queue

Mutual Exclusion

Lamport's Approach



T_a

time stamped acknowledgment message

request queue

Mutual Exclusion

- Lamport's Approach (Obtaining)
  - Pi broadcast request message to all processes with time stamp
  - each process adds the request to its request queue and sends a time stamped acknowledgment
  - obtains access if request is first on Pi's request queue and Pi receives a message from all processes with a time stamp no later than original broadcasted time stamp

Mutual Exclusion

- Lamport's Approach (Release)
  - broadcast a message to all other processes releasing the resource
  - process removes resource request from queue

Physical Clocks

- Similar notion, except that difference between time must be minimal
- Update the clock
  - t = max(t', Tm + Ts)
  - t' - current time of the clock
  - Tm - time stamp of the message
  - Ts - minimum delay for the message to travel

References

- Lamport, L., Time, Clocks, and Ordering of Events in a Distributed System, Communications of the ACM, July 1978
- Bic, L., Shaw, A., Operating Systems Principles Prentice Hall, 2003