CSE 421/521 - Operating Systems
Fall 2013

Lecture - XXIV

Distributed Systems - I

Tevfik Koşar

University at Buffalo

November 26th, 2013

Motivation

- **Distributed system** is collection of loosely coupled processors that
  - do not share memory
  - interconnected by a communications network
- Reasons for distributed systems
  - Resource sharing
    - sharing and printing files at remote sites
    - processing information in a distributed database
    - accessing remote files
    - using remote specialized hardware devices
  - Computation speedup – **load sharing**
  - Reliability – detect and recover from site failure, function transfer, reintegrate failed site

Distributed-Operating Systems

- Users not aware of multiplicity of machines
  - Access to remote resources similar to access to local resources
- Data Migration – transfer data by transferring entire file, or transferring only those portions of the file necessary for the immediate task
- Computation Migration – transfer the computation, rather than the data, across the system

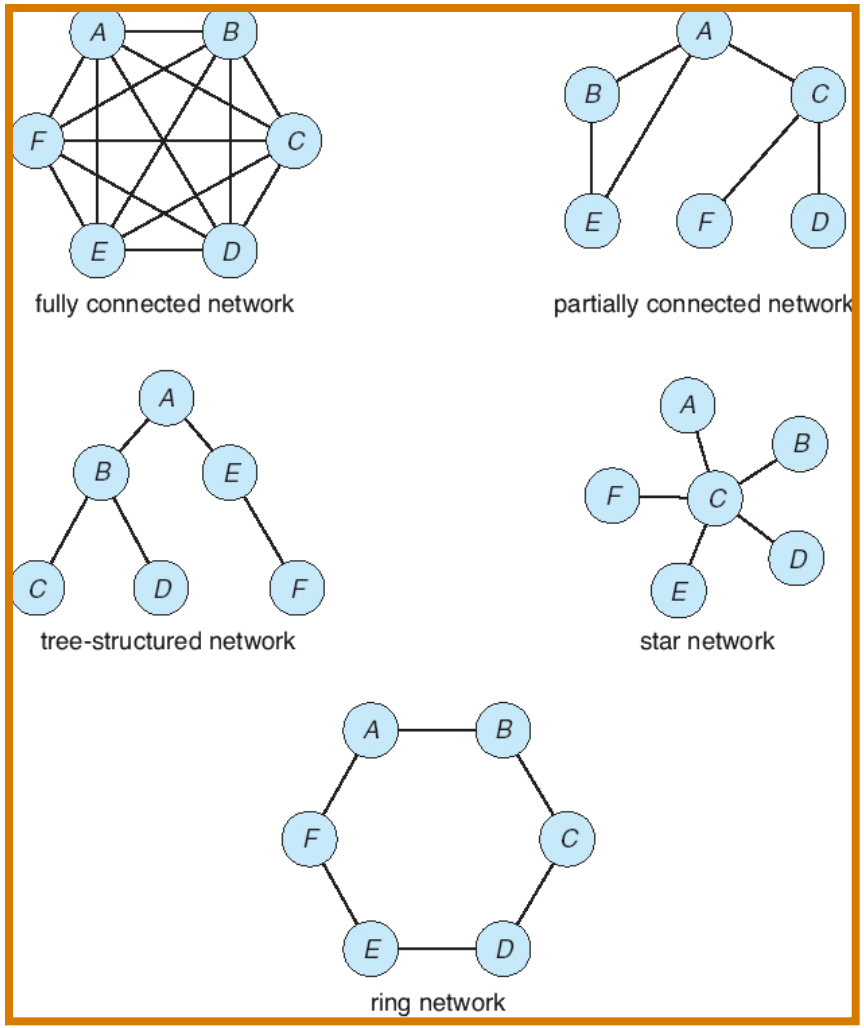
Distributed-Operating Systems (Cont.)

- Process Migration – execute an entire process, or parts of it, at different sites
  - Load balancing – distribute processes across network to even the workload
  - Computation speedup – subprocesses can run concurrently on different sites
  - Hardware preference – process execution may require specialized processor
  - Software preference – required software may be available at only a particular site
  - Data access – run process remotely, rather than transfer all data locally

Distributed File Systems

- Distributed file system (**DFS**) – a distributed implementation of the classical time-sharing model of a file system, where multiple users share files and storage resources over a network
- A DFS manages set of dispersed storage devices
- Overall storage space managed by a DFS is composed of different, remotely located, smaller storage spaces
- There is usually a correspondence between constituent storage spaces and sets of files

Distributed Network Topology



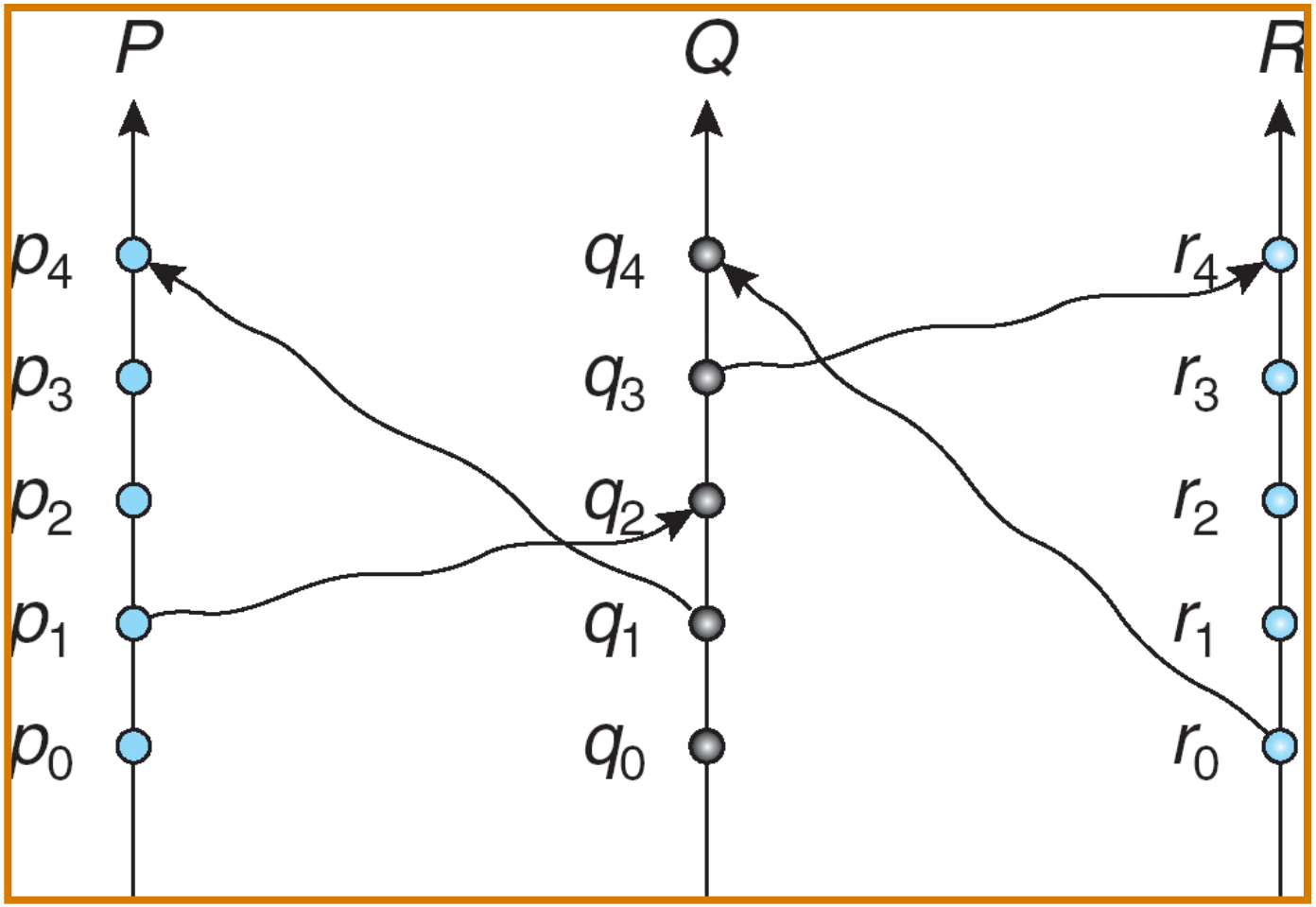
Distributed Coordination

- Ordering events and achieving synchronization in centralized systems is easier.
  - We can use common clock and memory
- What about distributed systems?
  - No common clock or memory
  - *happened-before* relationship provides partial ordering
  - How to provide total ordering?

Event Ordering

- Happened-before relation (denoted by $\rightarrow$)
  - If *A* and *B* are events in the same process (assuming sequential processes), and *A* was executed before *B*, then $A \rightarrow B$
  - If *A* is the event of sending a message by one process and *B* is the event of receiving that message by another process, then $A \rightarrow B$
  - If $A \rightarrow B$ and B $\rightarrow C$ then $A \rightarrow C$
  - If two events A and B are not related by the $\rightarrow$ relation, then these events are executed concurrently.

Relative Time for Three Concurrent Processes

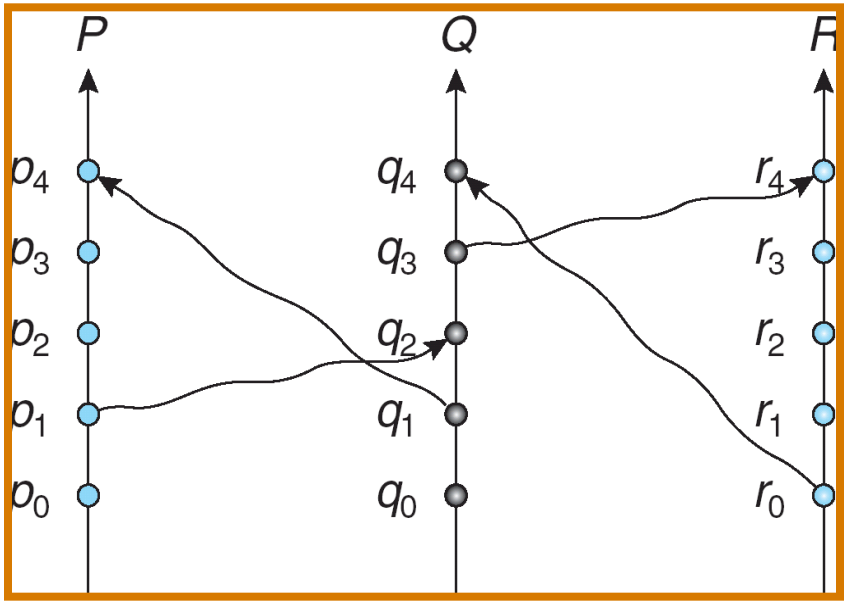


Which events are concurrent and which ones are ordered?

Exercise

Which of the following event orderings are true?

(a) p0 --> p3 :
(b) p1 --> q3 :
(c) q0 --> p3 :
(d) r0 --> p4 :
(e) p0 --> r4 :



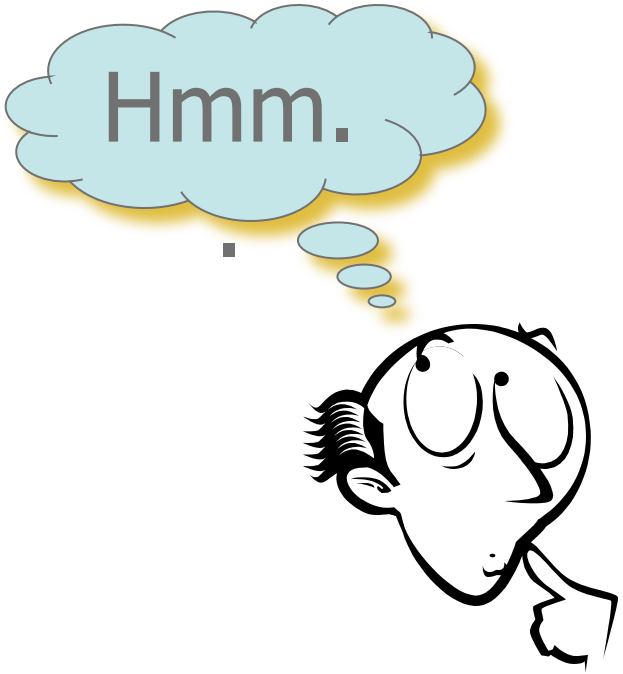
Which of the following statements are true?

(a) p2 and q2 are concurrent processes.
(b) q1 and r1 are concurrent processes.
(c) p0 and q3 are concurrent processes.
(d) r0 and p0 are concurrent processes.
(e) r0 and p4 are concurrent processes.

Implementation of $\rightarrow$

- Associate a timestamp with each system event
  - Require that for every pair of events A and B, if A $\rightarrow$ B, then the timestamp of A is less than the timestamp of B
- Within each process Pi, define a **logical clock**
  - The logical clock can be implemented as a simple counter that is incremented between any two successive events executed within a process
    - Logical clock is **monotonically increasing**
- A process advances its logical clock when it receives a message whose timestamp is greater than the current value of its logical clock
  - Assume A sends a message to B, $LC_1(A)=200$, $LC_2(B)=195$ --> $LC_2(B)=201$
- If the timestamps of two events A and B are the same, then the events are concurrent
  - We may use the process identity numbers to break ties and to create a total ordering

Any Questions?



Acknowledgements

- “Operating Systems Concepts” book and supplementary material by A. Silberschatz, P. Galvin and G. Gagne
- “Operating Systems: Internals and Design Principles” book and supplementary material by W. Stallings
- “Modern Operating Systems” book and supplementary material by A. Tanenbaum
- R. Doursat and M. Yuksel from UNR