

Distributed Systems and Applications

Ratan Guha

School of Computer Science
University of Central Florida
Orlando, FL 32816

guha@cs.ucf.edu

August 2, 2005

Acknowledgements

Diagrams are taken from the slides available for the book

A. S. Tanenbaum and M. van Steen,
Distributed Systems Principles and Paradigms, Prentice Hall Inc., 2002

Module Objectives

- Important principles of distributed systems
- Some of the paradigms of distributed systems
- High Level Architecture (HLA)

Topics on Principles of Distributed Systems

- Communication
- Processes
- Naming
- Synchronization
- Consistency and replication
- Fault tolerance

Text and Recommended Books

- A. S. Tanenbaum and M. van Steen, *Distributed Systems Principles and Paradigms*, Prentice Hall Inc., 2002.
- **Recommended books**
 - G. Coulouris, J. Dollimore, and T. Kindberg, *Distributed Systems Concepts and Design*, Third Edition, Addison Wesley, 2001.
 - F. Kuhl, R. Weatherly, J. Dahmann, *Creating Computer Simulation Systems*, Prentice Hall Inc., 1999.

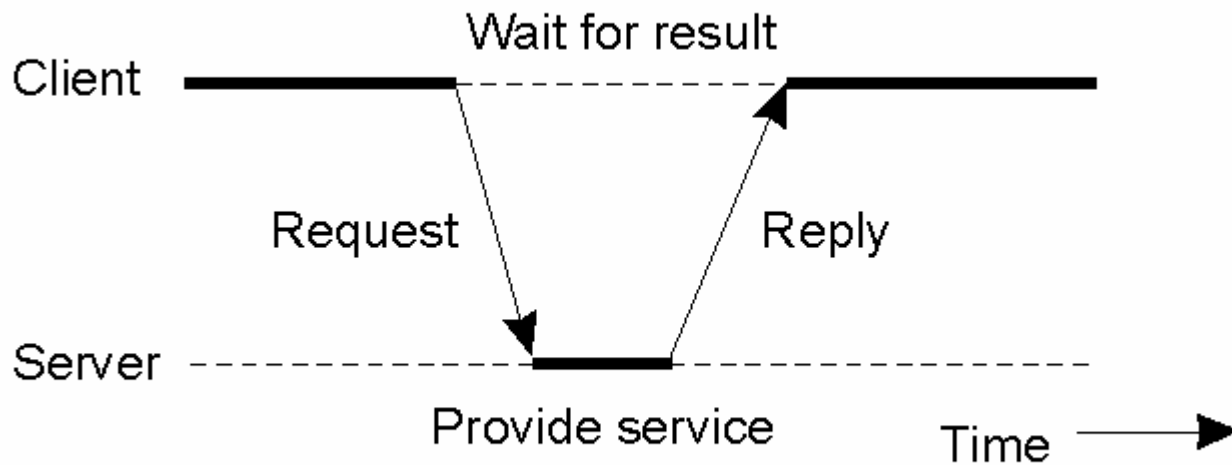
Communication

- Remote procedure call
- Synchronous communication
- Asynchronous communication

Remote Procedure Call

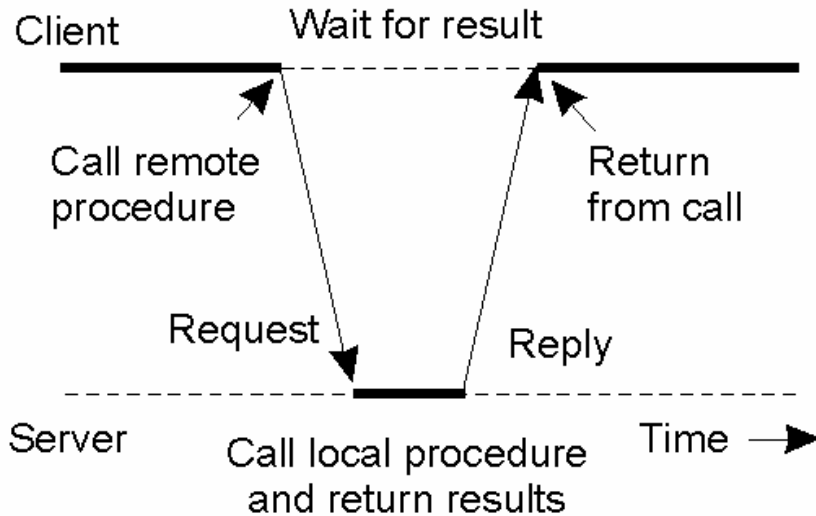
- Conventional procedure call
- Remote procedure call
 - Client and Server Stubs
 - Parameter passing
- Extended RPC models
 - Doors
 - Asynchronous RPC
 - Deferred synchronous RPC

Clients and Servers

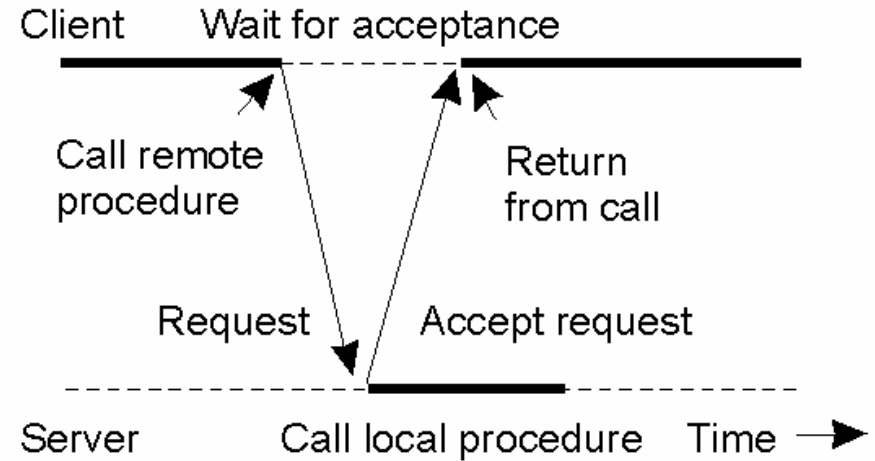


- Principle of RPC between a client and server program.

Asynchronous RPC (1)



(a)



(b)

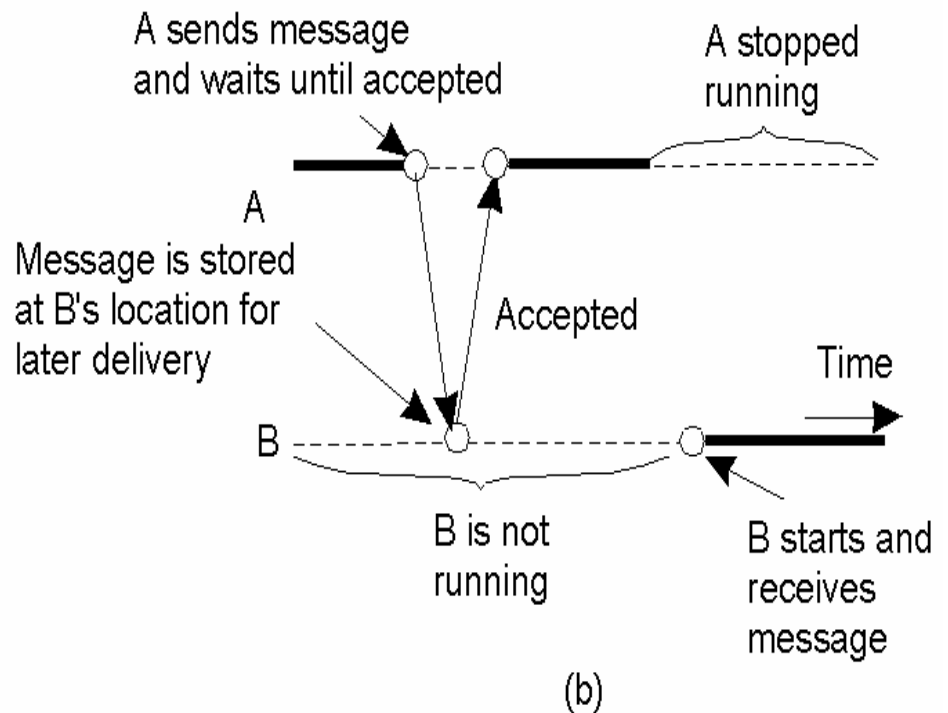
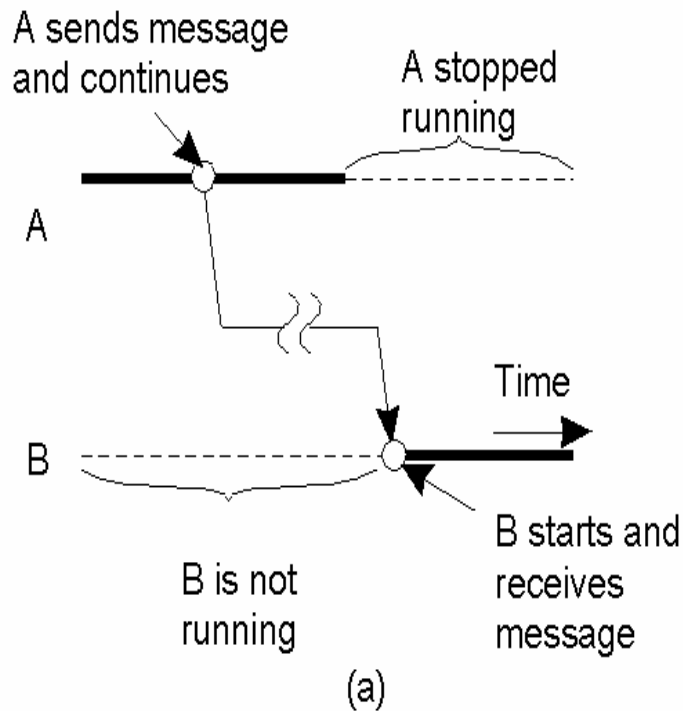
- a) The interconnection between client and server in a traditional RPC
- b) The interaction using asynchronous RPC

Persistence and Synchronicity in Message-Oriented Communication

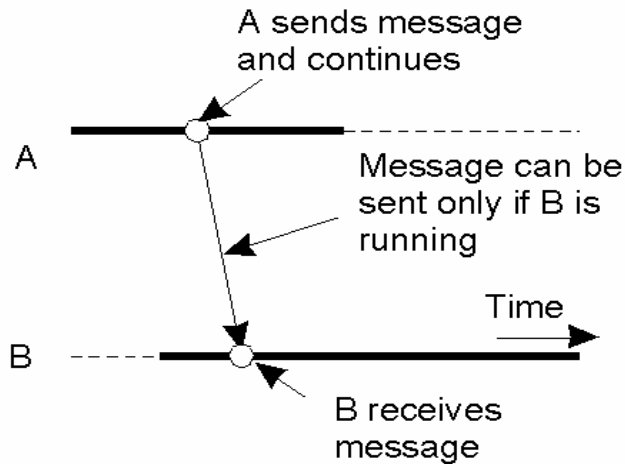
- Persistent asynchronous communication
- Persistent synchronous communication
- Transient asynchronous communication
- Receipt-based transient synchronous communication
- Delivery-based transient synchronous communication at message delivery
- Response-based transient synchronous communication

Persistence and Synchronicity in Communication (1)

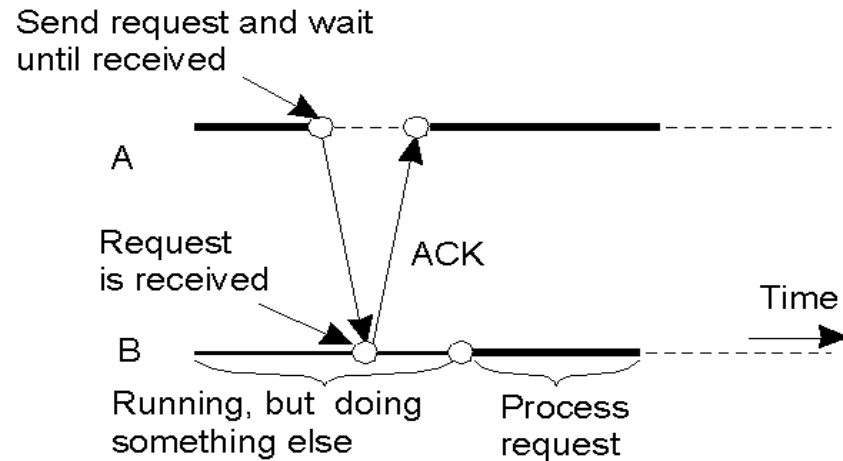
- a) Persistent asynchronous communication
- b) Persistent synchronous communication



Persistence and Synchronicity in Communication (2)



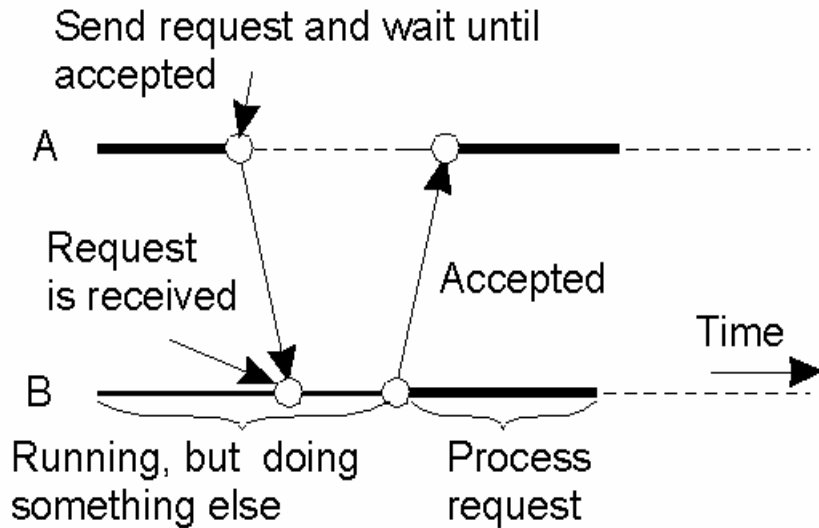
(c)



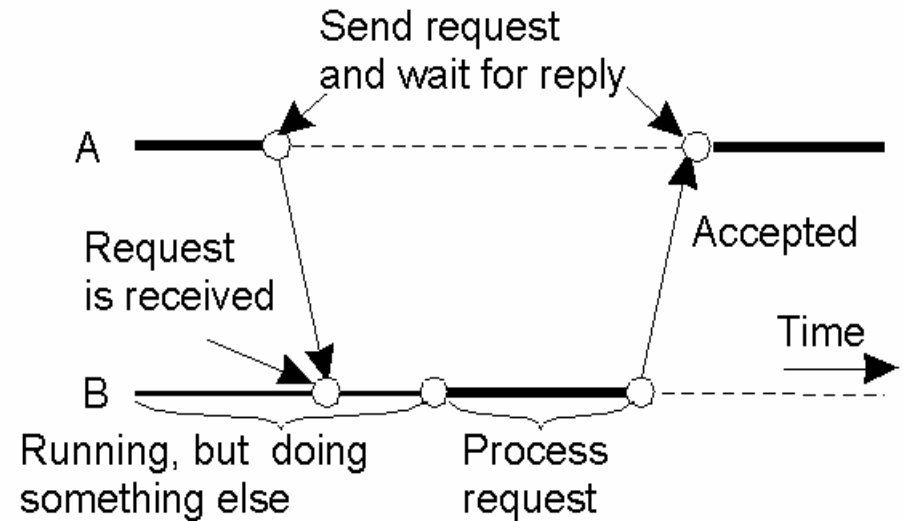
(d)

- c) Transient asynchronous communication
- d) Receipt-based transient synchronous communication

Persistence and Synchronicity in Communication (3)



(e)



(f)

- e) Delivery-based transient synchronous communication at message delivery
- f) Response-based transient synchronous communication

Synchronization

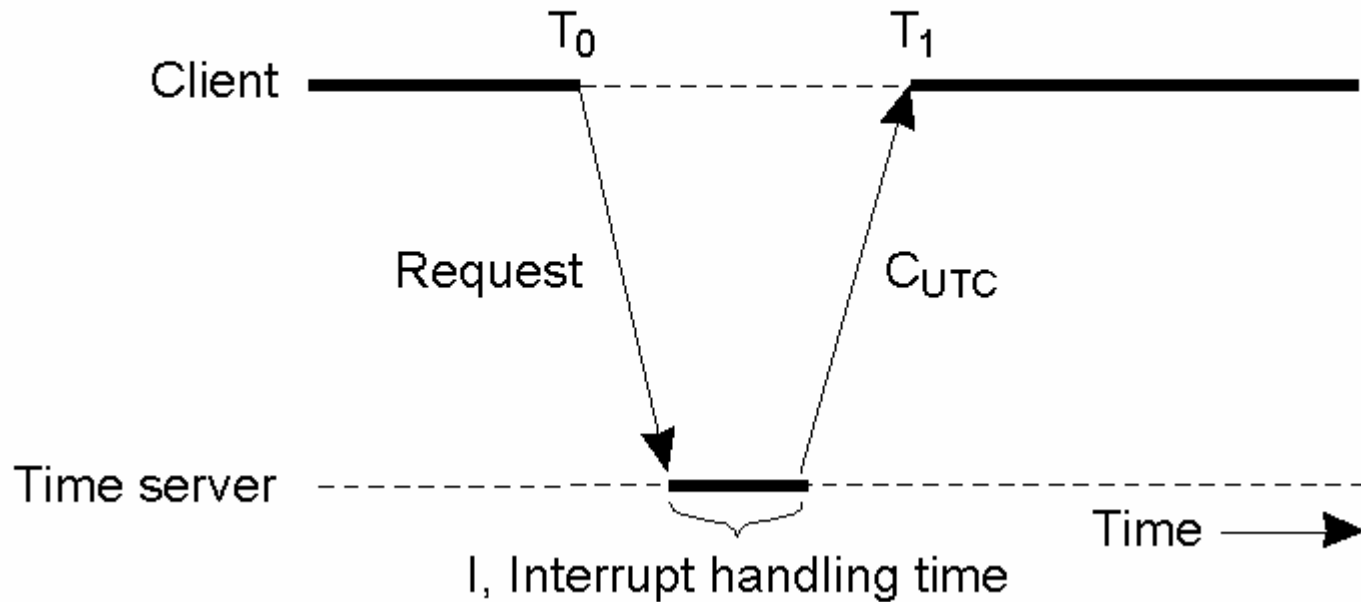
- Physical clock
- Logical clock
- Global state
- Election algorithms
- Mutual exclusion algorithms

Physical Clocks

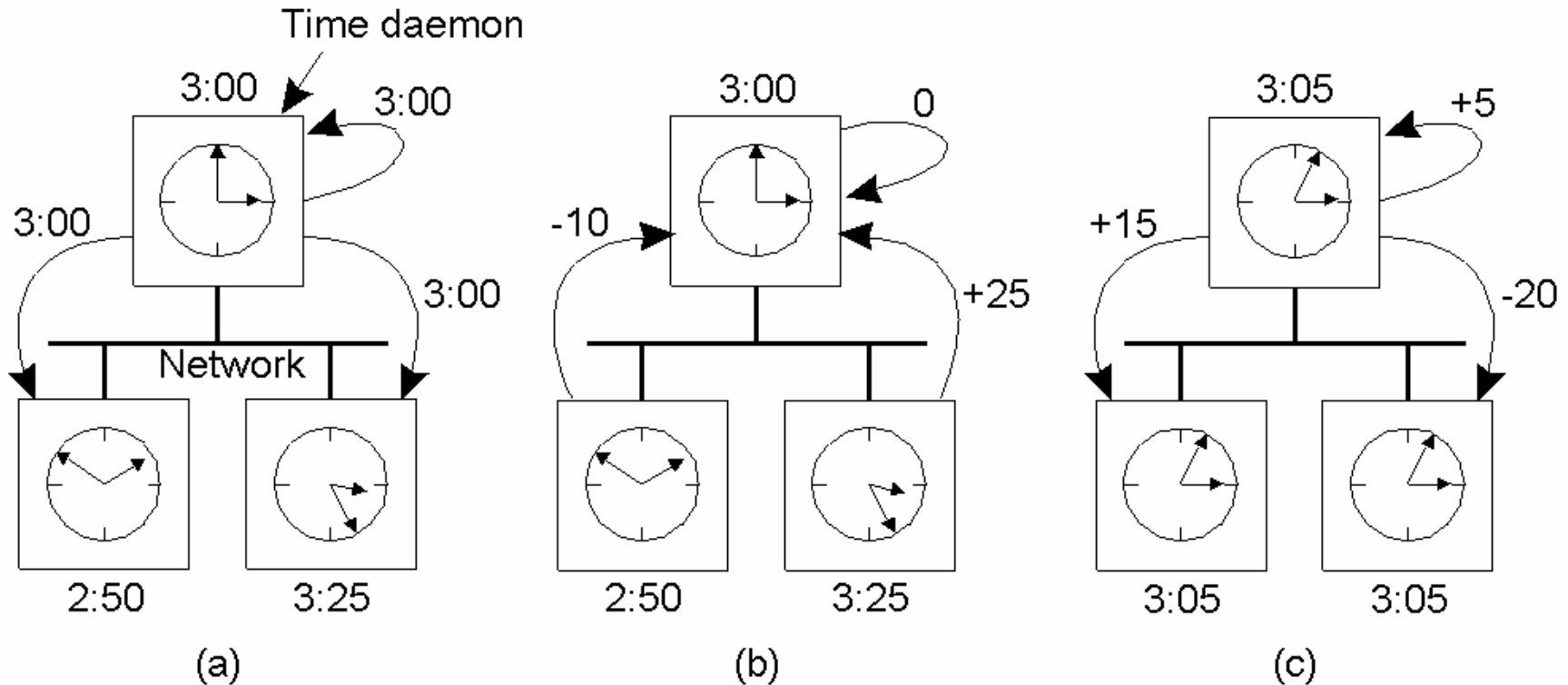
- 1 sec = Cesium 133 atom make 9,192,631,770 transitions (Solar second) – 50 labs
- International Atomic Time (TAI) [average]
- Universal Coordinated Time (UTC)
- Shortwave Radio Station – WWV broadcasts a short pulse at the start of each UTC second
- Clock synchronization
- Clock synchronization algorithms
 - Cristian's algorithm
 - Berkeley algorithm
 - Averaging algorithms

Cristian's Algorithm

Both T_0 and T_1 are measured with the same clock



The Berkeley Algorithm

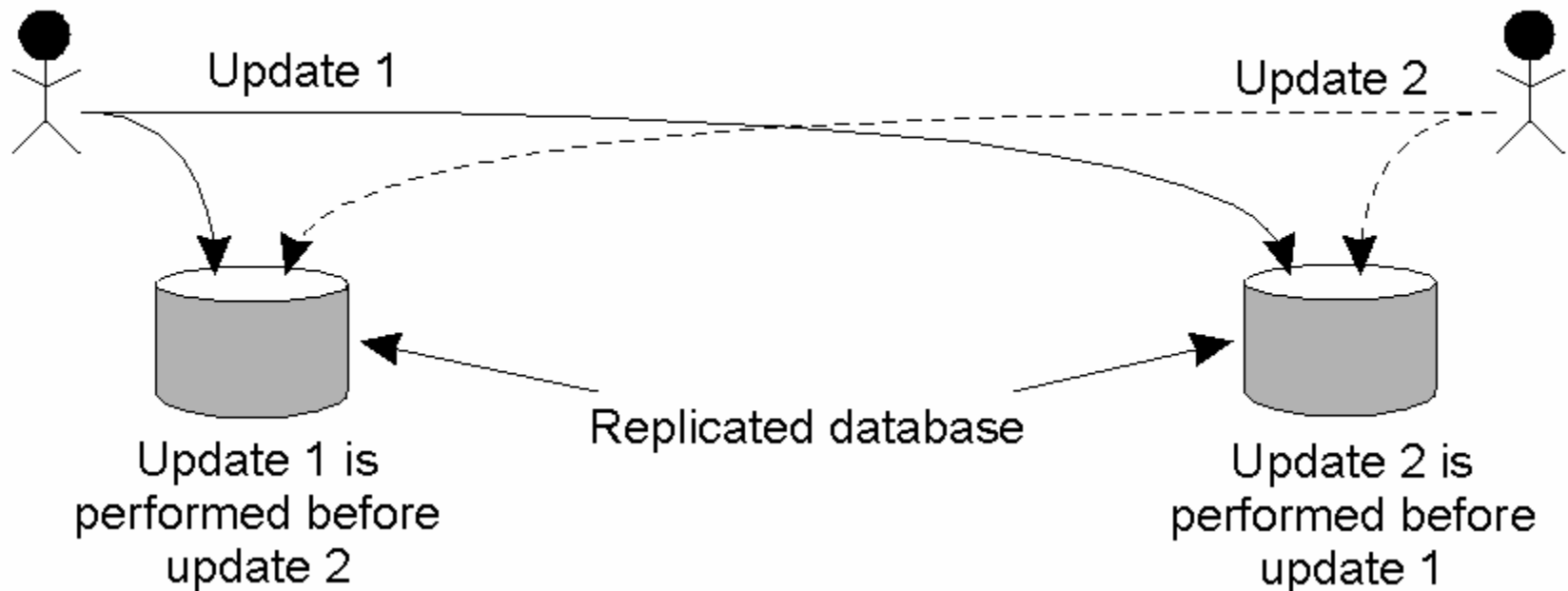


- The time daemon asks all the other machines for their clock values
- The machines answer
- The time daemon tells everyone how to adjust their clock

Logical Clocks

- Lamport timestamps
 - Happens before ($a \rightarrow b$)
- Totally-ordered multicasting
- Vector timestamps

Lamport Timestamps



Updating a replicated database and leaving it in inconsistent state
Lamport's algorithm corrects the clocks (totally ordered multicast).

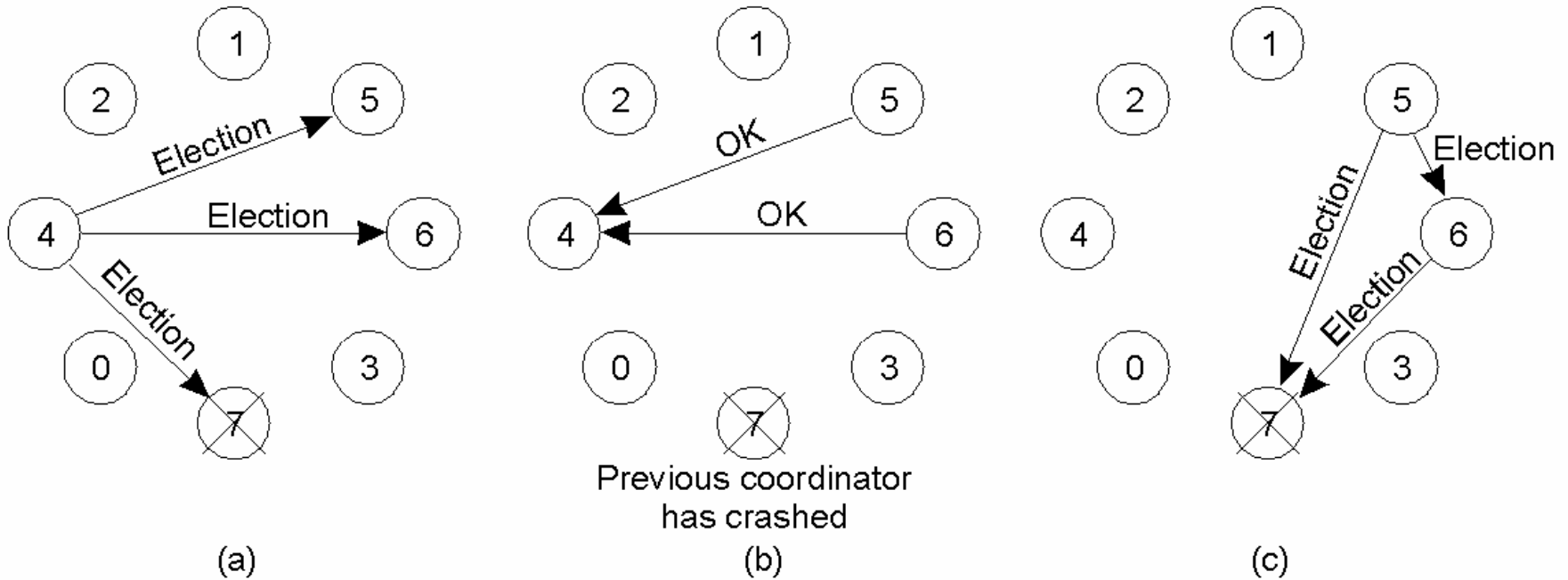
Global State

- Local state of each process, together with the messages that are currently in transit
- Chandy and Lamport idea of distributed snapshot

Election Algorithms

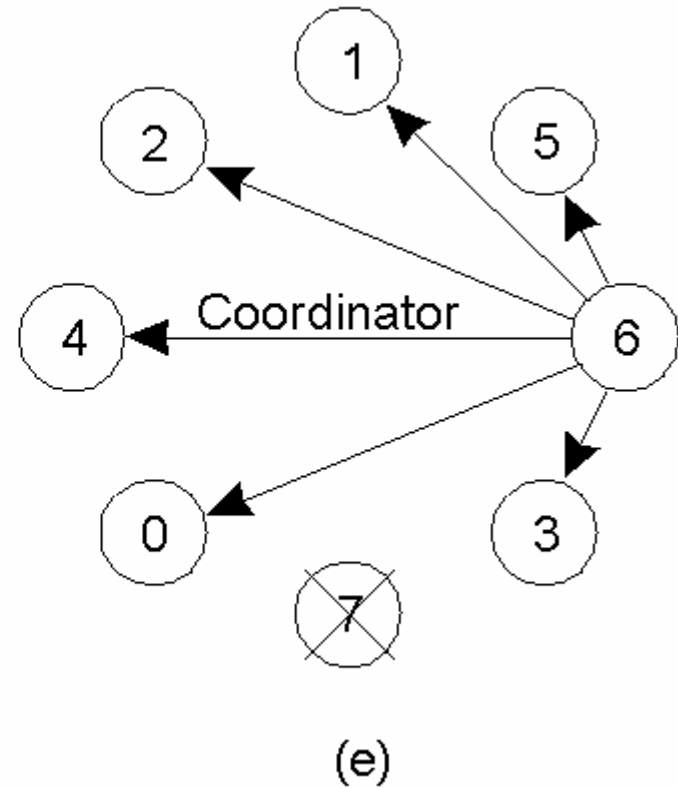
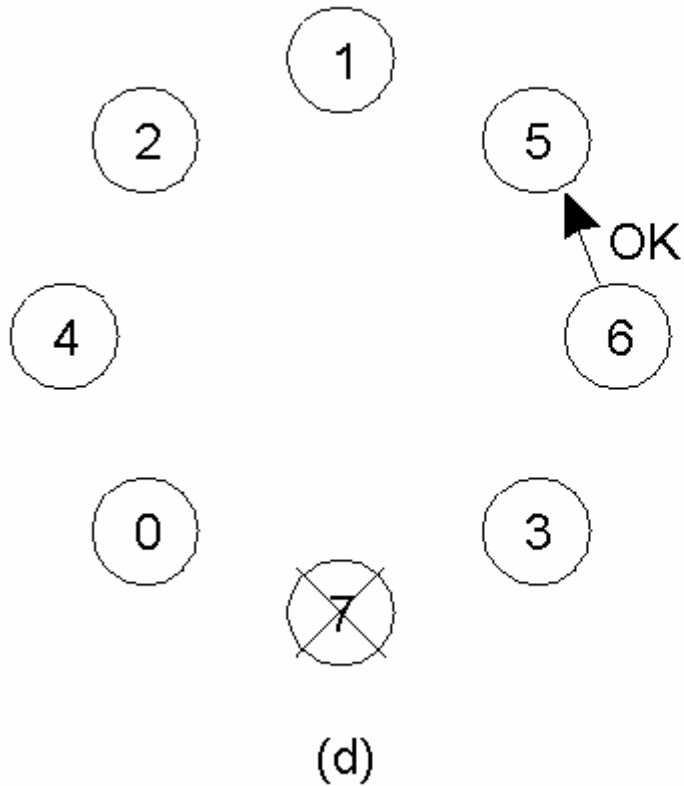
- The Bully algorithm
- A Ring algorithm

The Bully Algorithm (1)

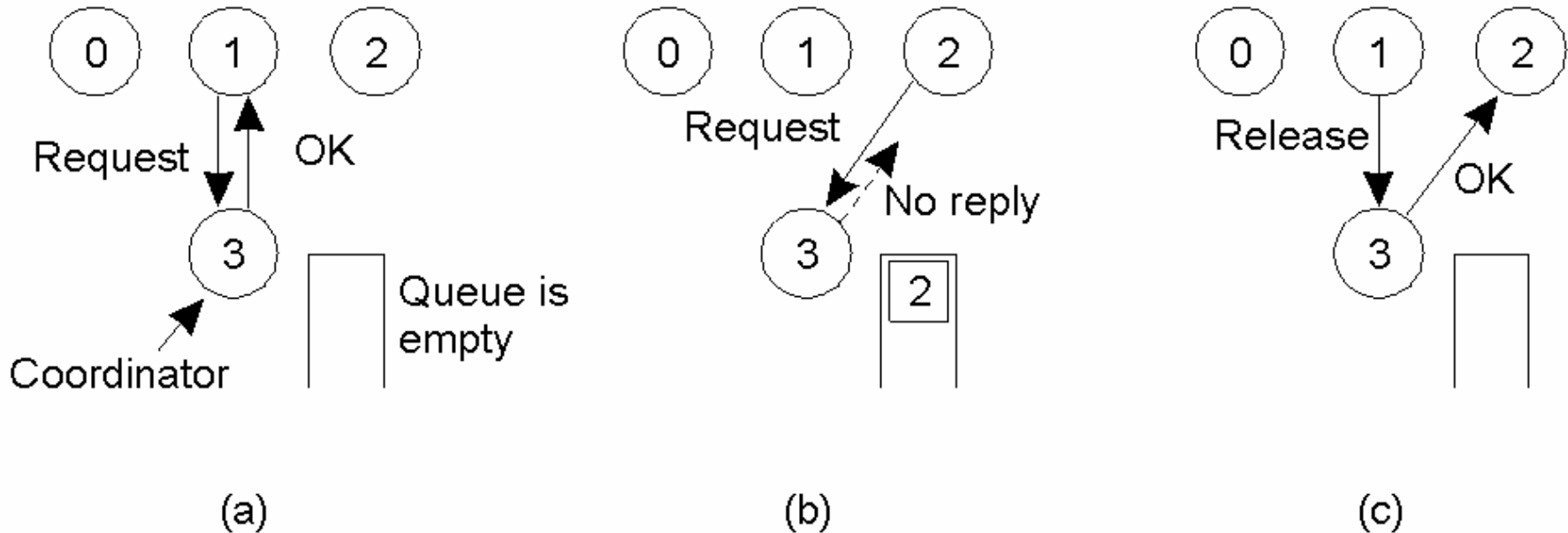


- The bully election algorithm
- Process 4 holds an election
- Process 5 and 6 respond, telling 4 to stop
- Now 5 and 6 each hold an election

The Bully Algorithm (2)

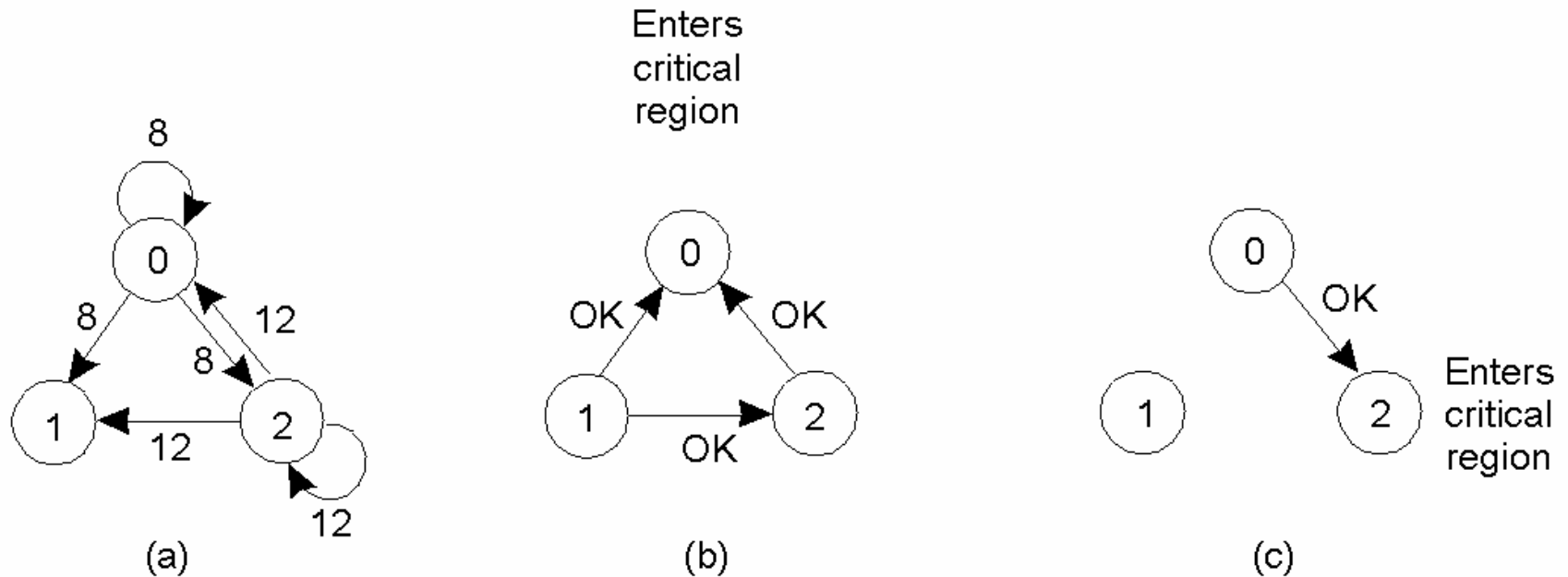


Mutual Exclusion: A Centralized Algorithm



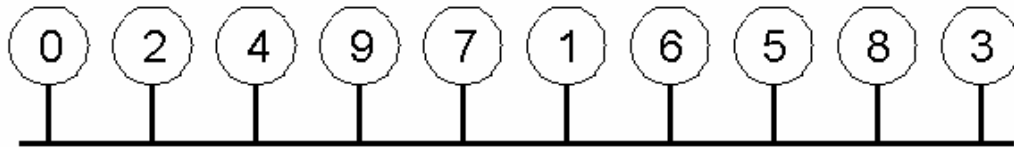
- Process 1 asks the coordinator for permission to enter a critical region. Permission is granted
- Process 2 then asks permission to enter the same critical region. The coordinator does not reply.
- When process 1 exits the critical region, it tells the coordinator, when then replies to 2

A Distributed Algorithm

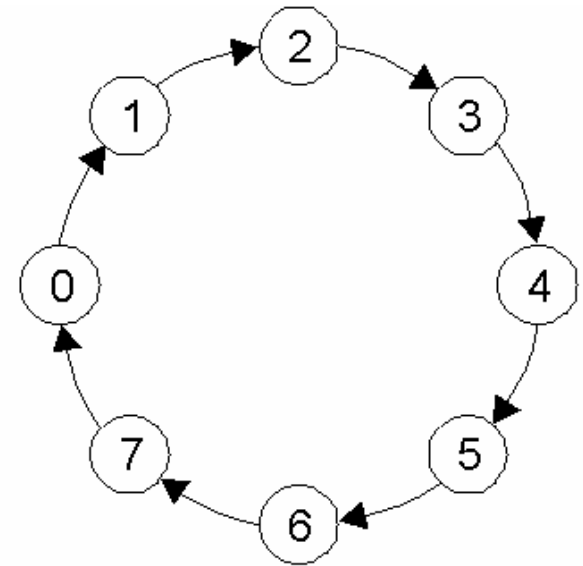


- Two processes want to enter the same critical region at the same moment.
- Process 0 has the lowest timestamp, so it wins.
- When process 0 is done, it sends an OK also, so 2 can now enter the critical region.

A Token Ring Algorithm



(a)



(b)

- a) An unordered group of processes on a network.
- b) A logical ring constructed in software.

Comparison

Algorithm	Messages per entry/exit	Delay before entry (in message times)	Problems
Centralized	3	2	Coordinator crash
Distributed	$2(n - 1)$	$2(n - 1)$	Crash of any process
Token ring	1 to ∞	0 to $n - 1$	Lost token, process crash

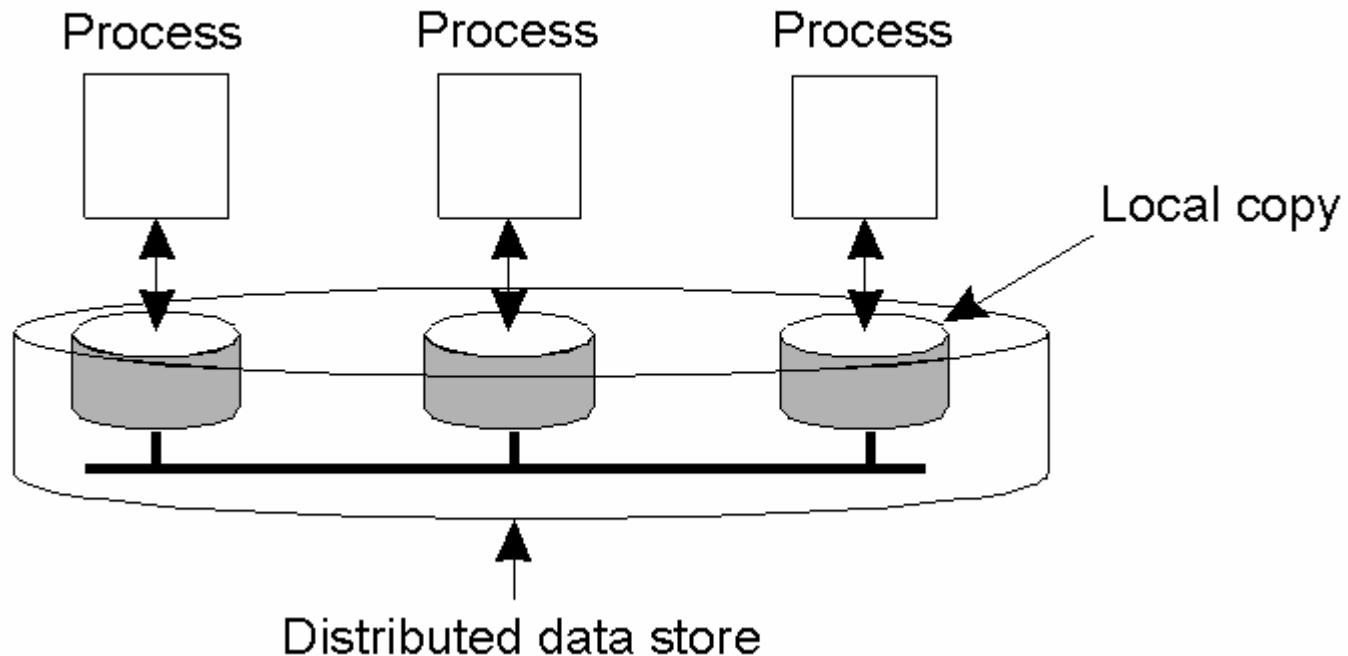
- A comparison of three mutual exclusion algorithms.

Replication and Consistency

- Object replication
- Data-centric consistency models
- Client-centric consistency models
- Consistency protocols

Data-Centric Consistency Models

- The general organization of a logical data store, physically distributed and replicated across multiple processes.



Data-Centric Consistency Models

- Strict Consistency
- Linearizability and Sequential Consistency
- Casual Consistency
- FIFO Consistency
- Weak Consistency
- Release Consistency
- Entry Consistency

Strict Consistency

P1: W(x)a

P2: R(x)a
(a)

P1: W(x)a

P2: R(x)NIL R(x)a
(b)

- Behavior of two processes, operating on the same data item.
- A strictly consistent store.
- A store that is not strictly consistent.

Linearizability and Sequential Consistency (1)

P1:	W(x)a		
<hr/>			
P2:	W(x)b		
<hr/>			
P3:		R(x)b	R(x)a
<hr/>			
P4:		R(x)b	R(x)a

(a)

P1:	W(x)a		
<hr/>			
P2:	W(x)b		
<hr/>			
P3:		R(x)b	R(x)a
<hr/>			
P4:		R(x)a	R(x)b

(b)

- a) A sequentially consistent data store.
- b) A data store that is not sequentially consistent.

Casual Consistency (1)

- Necessary condition:
Writes that are potentially casually related must be seen by all processes in the same order. Concurrent writes may be seen in a different order on different machines.

Casual Consistency (2)

P1:	W(x)a		W(x)c		
P2:		R(x)a	W(x)b		
P3:		R(x)a		R(x)c	R(x)b
P4:		R(x)a		R(x)b	R(x)c

- This sequence is allowed with a casually-consistent store, but not with sequentially or strictly consistent store.

Casual Consistency (3)

P1:	W(x)a		
P2:		R(x)a	W(x)b
P3:			R(x)b
P4:			R(x)a

(a)

P1:	W(x)a		
P2:		W(x)b	
P3:			R(x)b
P4:			R(x)a

(b)

- a) A violation of a casually-consistent store.
- b) A correct sequence of events in a casually-consistent store.

FIFO Consistency (1)

- Necessary Condition:
Writes done by a single process are seen by all other processes in the order in which they were issued, but writes from different processes may be seen in a different order by different processes.

FIFO Consistency (2)

P1: W(x)a

P2: R(x)a W(x)b W(x)c

P3: R(x)b R(x)a R(x)c

P4: R(x)a R(x)b R(x)c

- A valid sequence of events of FIFO consistency
- Writes by a process seen in the same order by others
- Writes by different processes may be seen in different order

Discussion About the Class

- Homework
- Laboratory assignments
- Reading of papers
- Mid term examination
- Final examination
- Final project
 - Programming project
 - Writing term paper

Comments about the course

- Students liked programming with HLA
- Some students enjoyed reading papers
- Some concepts were difficult
- Some students did not have enough background