DISTRIBUTED SYSTEMS
Principles and Paradigms
Second Edition
ANDREW S. TANENBAUM
MAARTEN VAN STEEN

Chapter 7
Consistency And Replication

A. A. Pourhaji Kazem, Spring 2009

Reasons for Replication

- Data are replicated to increase the reliability of a system.
- Replication for performance
  - Scaling in numbers
  - Scaling in geographical area
- Caveat
  - Gain in performance
  - Cost of increased bandwidth for maintaining replication

Data-centric Consistency Models

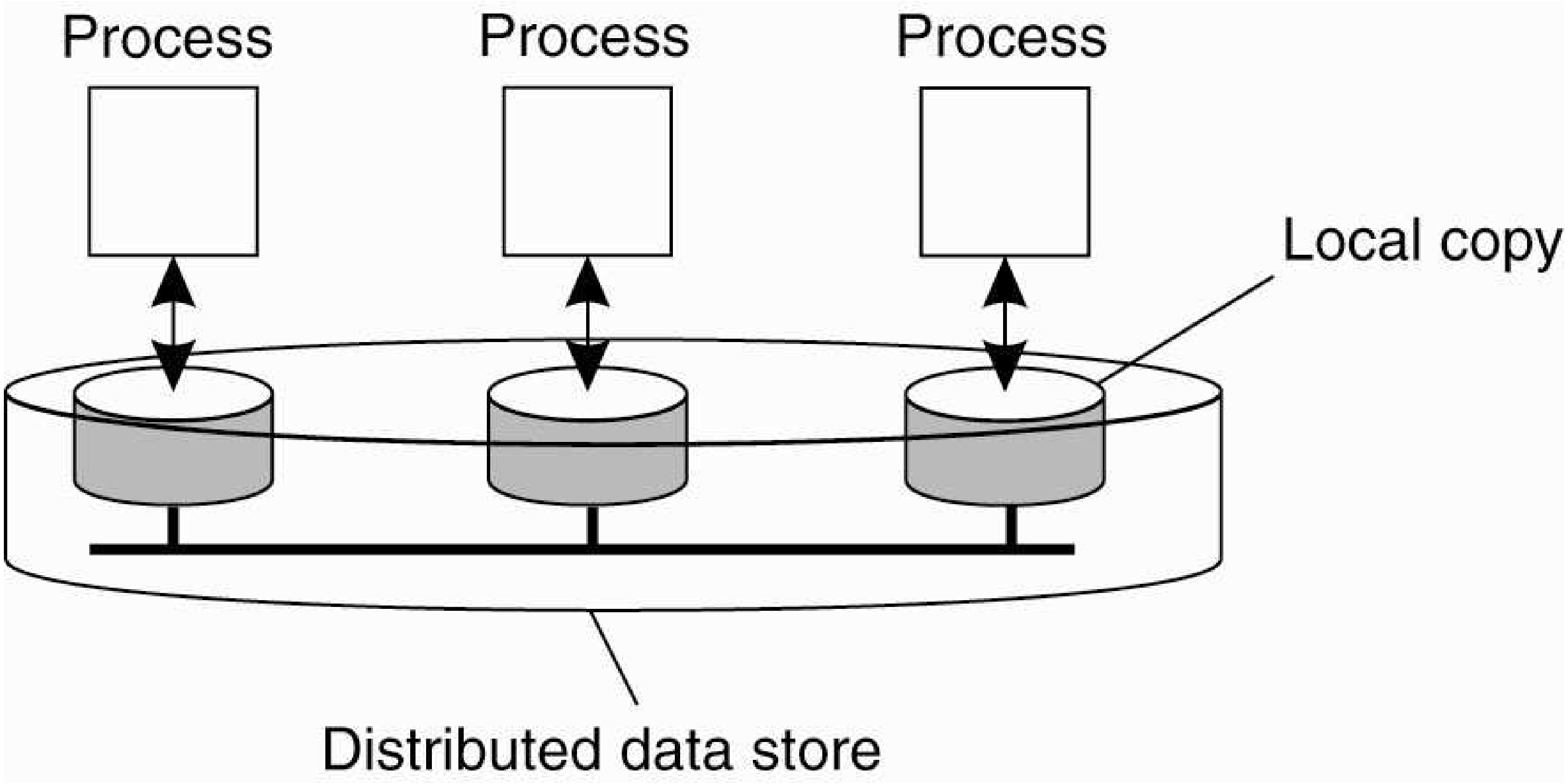
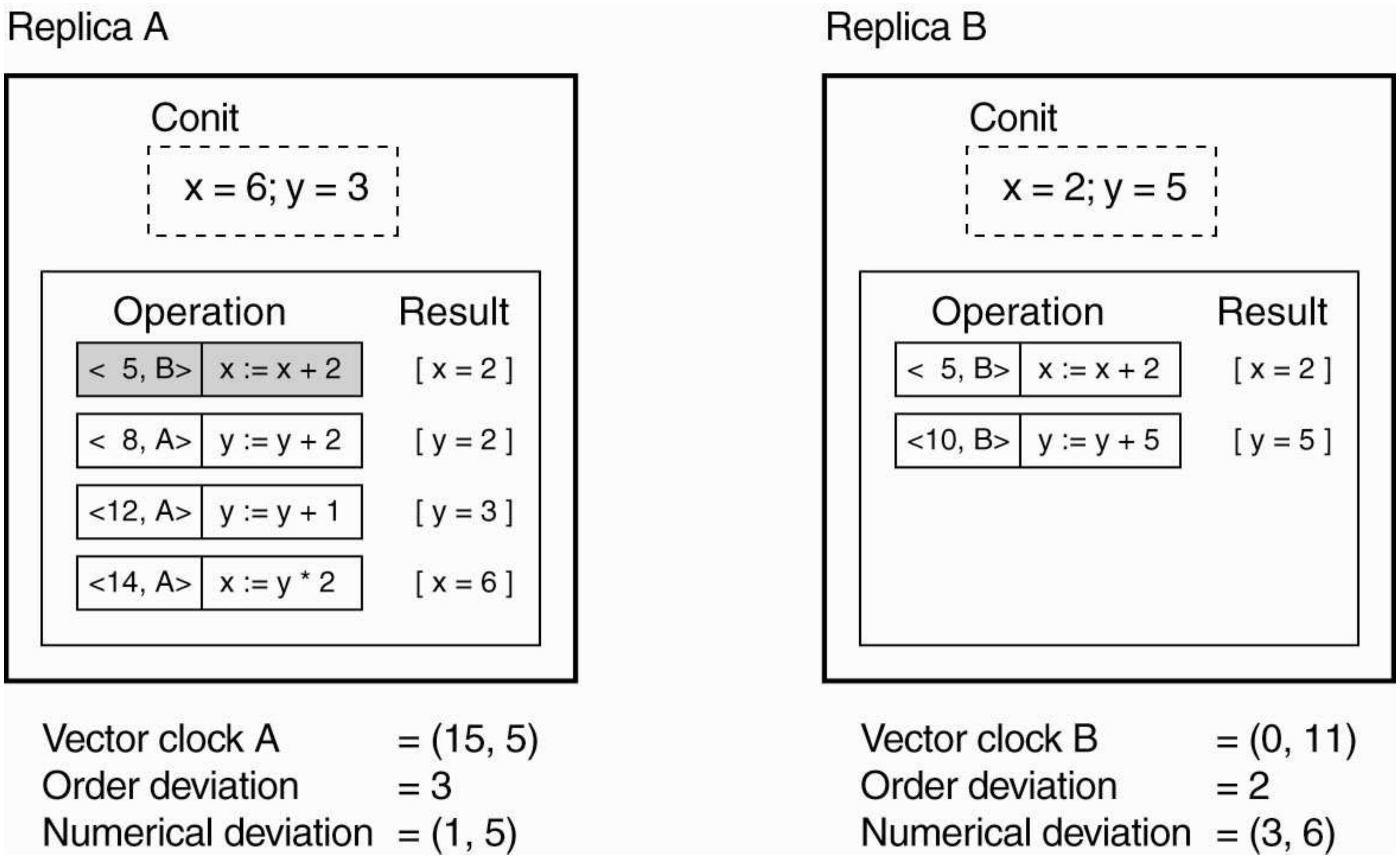


Figure 7-1. The general organization of a logical data store, physically distributed and replicated across multiple processes.

Continuous Consistency (1)



Replica A

Conit

x = 6; y = 3

Operation		Result
< 5, B>	x := x + 2	[x = 2]
< 8, A>	y := y + 2	[y = 2]
<12, A>	y := y + 1	[y = 3]
<14, A>	x := y * 2	[x = 6]

Vector clock A = (15, 5)
Order deviation = 3
Numerical deviation = (1, 5)

Replica B

Conit

x = 2; y = 5

Operation		Result
< 5, B>	x := x + 2	[x = 2]
<10, B>	y := y + 5	[y = 5]

Vector clock B = (0, 11)
Order deviation = 2
Numerical deviation = (3, 6)

Figure 7-2. An example of keeping track of consistency deviations [adapted from (Yu and Vahdat, 2002)].

Continuous Consistency (2)

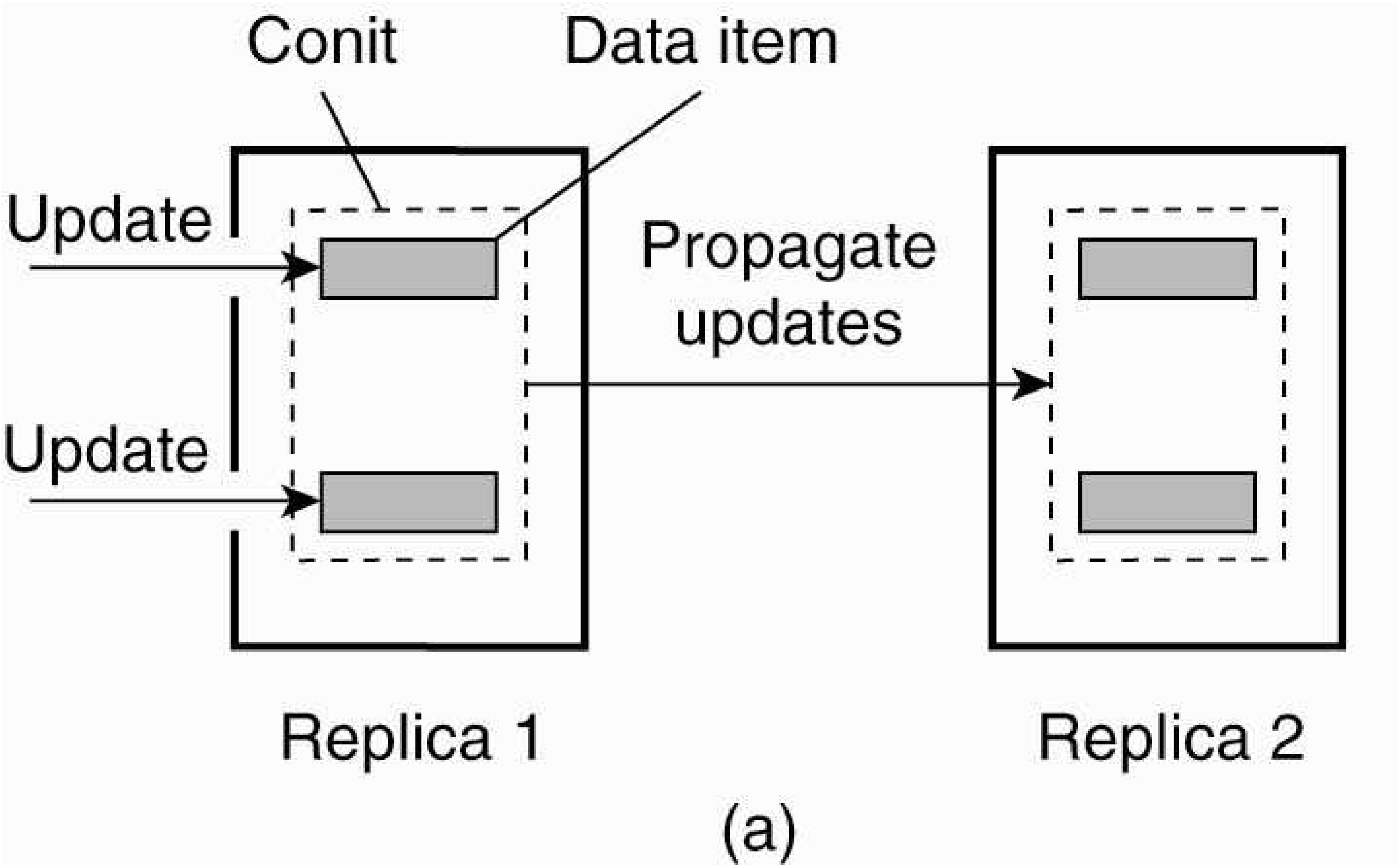


Figure 7-3. Choosing the appropriate granularity for a conit. (a) Two updates lead to update propagation.

Continuous Consistency (3)

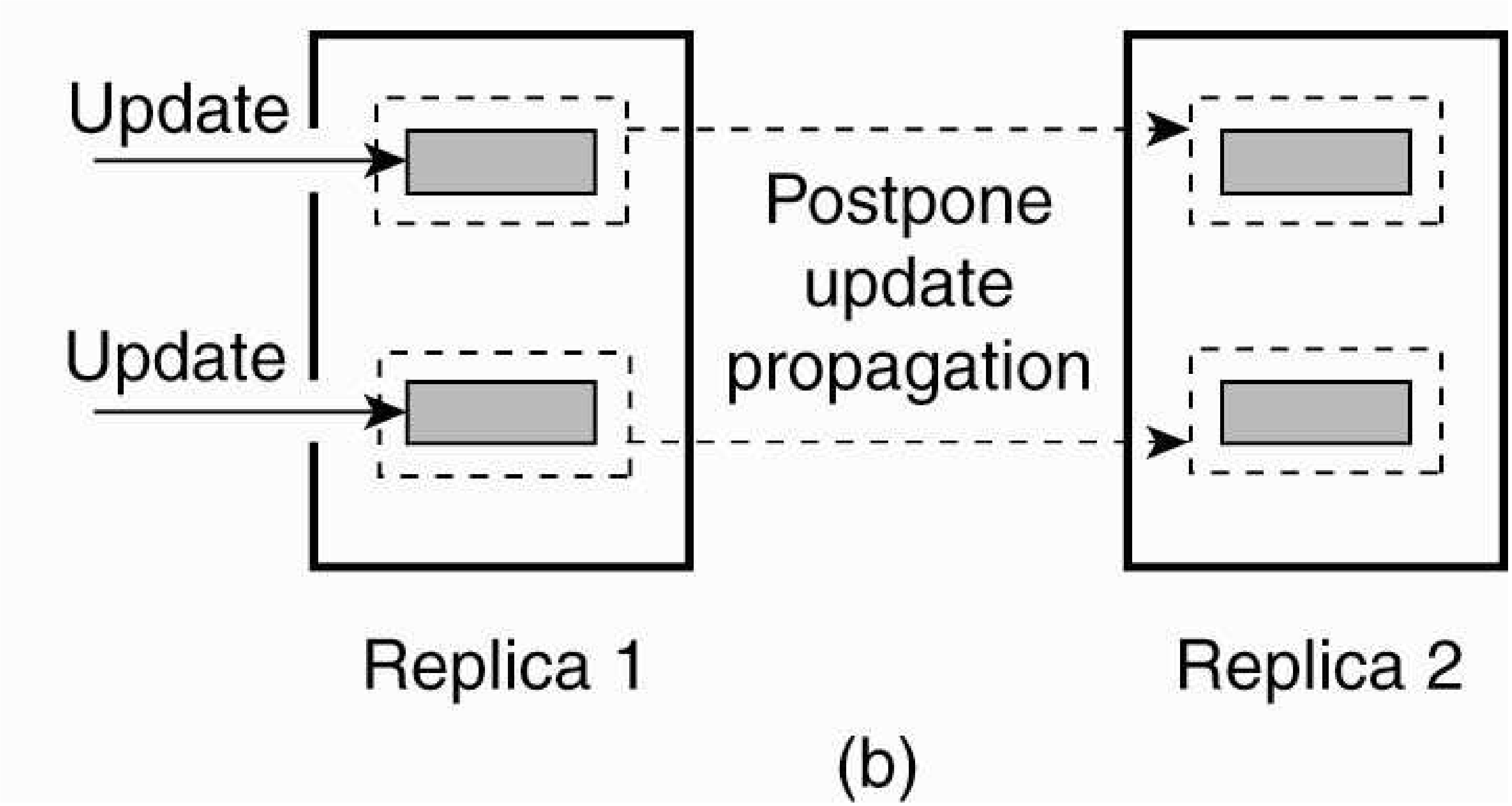
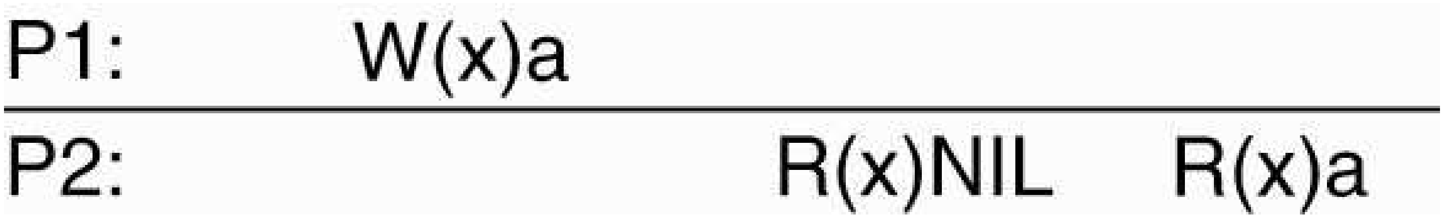


Figure 7-3. Choosing the appropriate granularity for a conit. (b) No update propagation is needed (yet).

Sequential Consistency (1)



P1: W(x)a

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

Figure 7-4. Behavior of two processes operating on the same data item. The horizontal axis is time.

Sequential Consistency (2)

A data store is sequentially consistent when:

The result of any execution is the same as if the (read and write) operations by all processes on the data store …

- were executed in some sequential order and …
- the operations of each individual process appear …
  - in this sequence
  - in the order specified by its program.

Sequential Consistency (3)

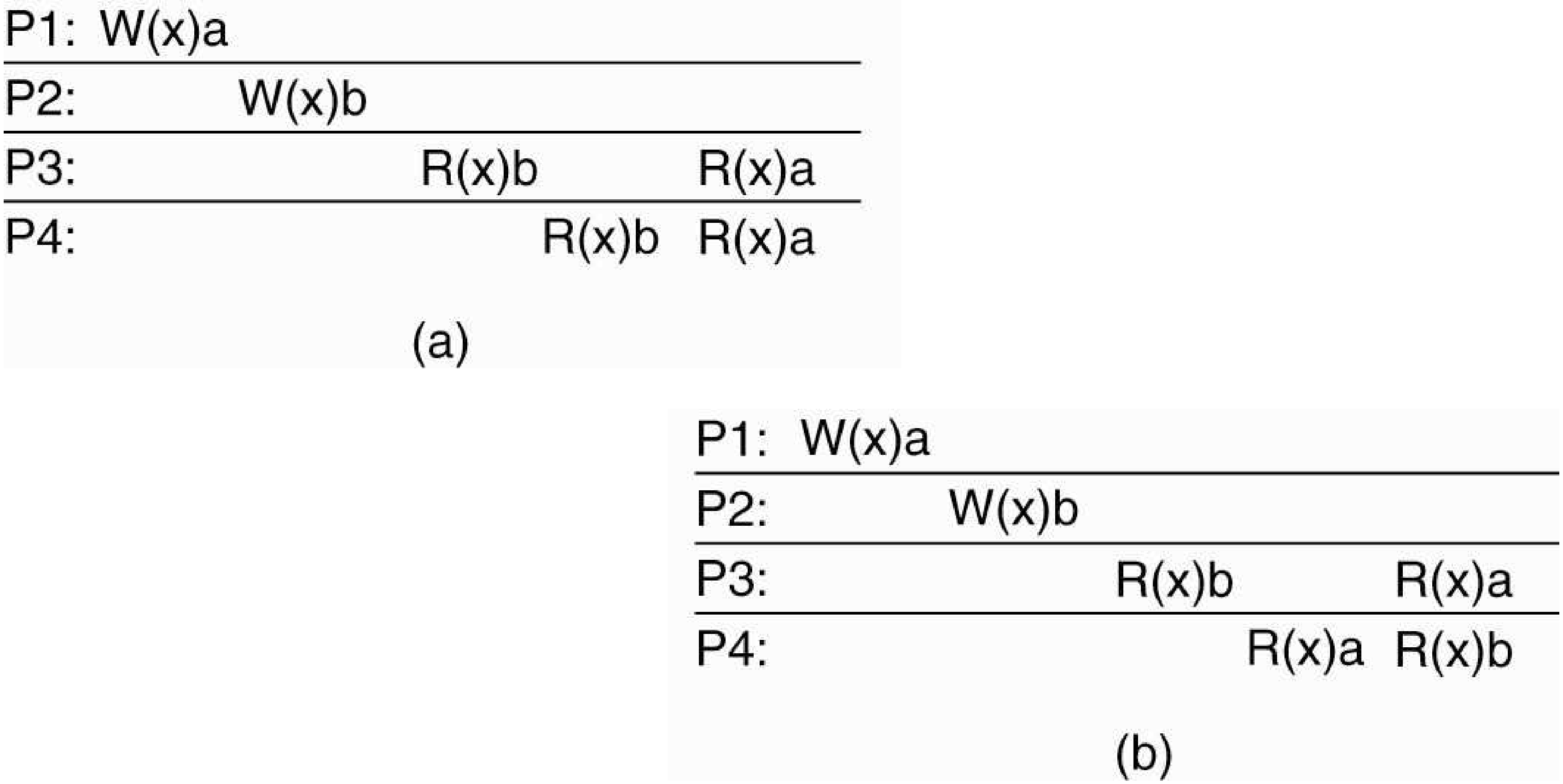
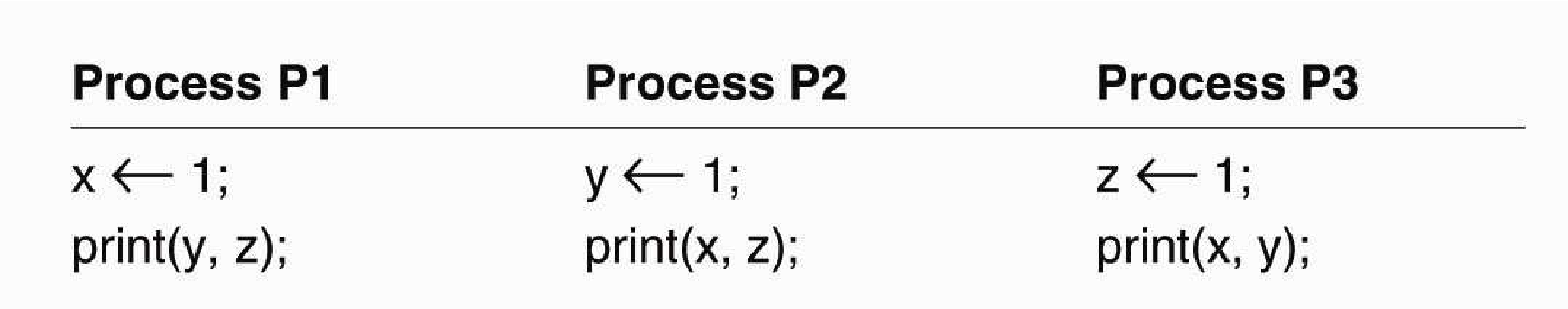


Figure 7-5. (a) A sequentially consistent data store.
(b) A data store that is not sequentially consistent.

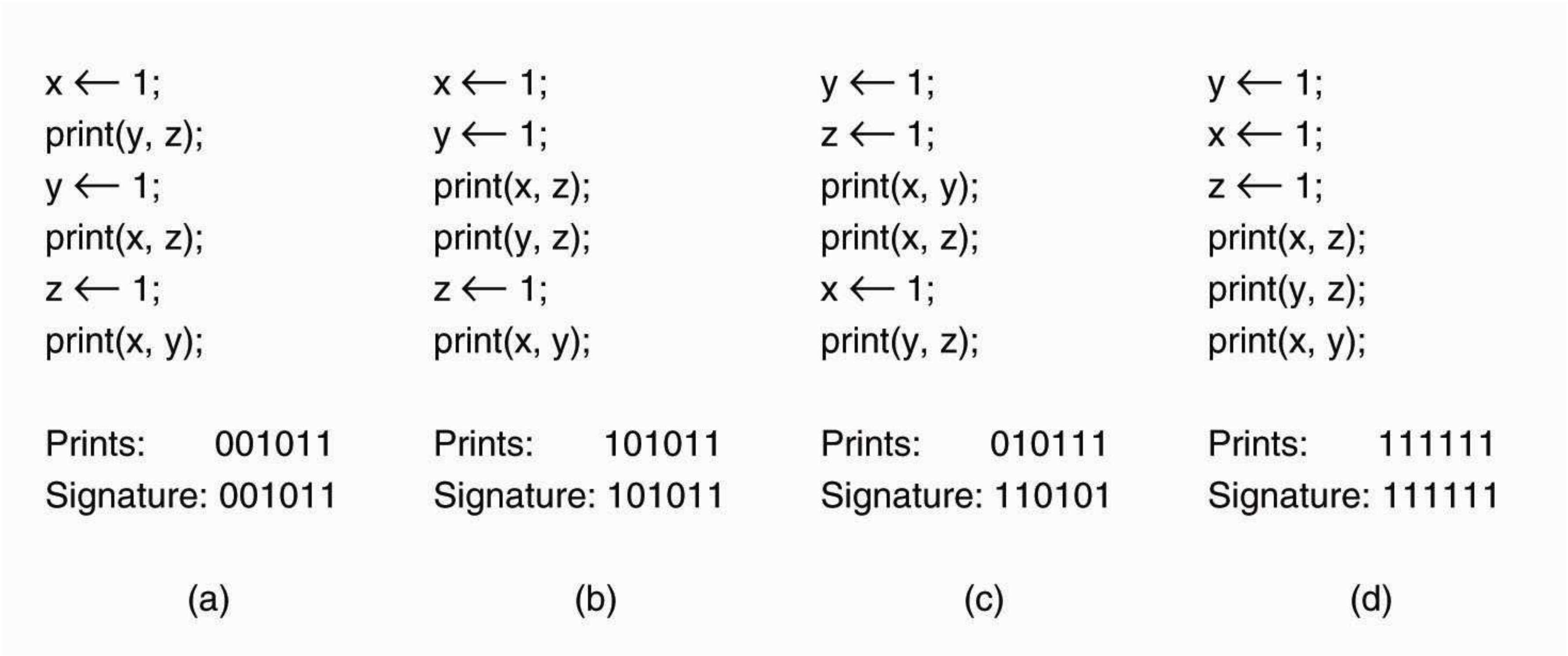
Sequential Consistency (4)



Process P1	Process P2	Process P3
x ← 1;	y ← 1;	z ← 1;
print(y, z);	print(x, z);	print(x, y);

Figure 7-6. Three concurrently-executing processes.

Sequential Consistency (5)



x ← 1;	x ← 1;	y ← 1;	y ← 1;
print(y, z);	y ← 1;	z ← 1;	x ← 1;
y ← 1;	print(x, z);	print(x, y);	z ← 1;
print(x, z);	print(y, z);	print(x, z);	print(x, z);
z ← 1;	z ← 1;	x ← 1;	print(y, z);
print(x, y);	print(x, y);	print(y, z);	print(x, y);
Prints: 001011	Prints: 101011	Prints: 010111	Prints: 111111
Signature: 001011	Signature: 101011	Signature: 110101	Signature: 111111
(a)	(b)	(c)	(d)

Figure 7-7. Four valid execution sequences for the processes of Fig. 7-6. The vertical axis is time.

Causal Consistency (1)

For a data store to be considered causally consistent, it is necessary that the store obeys the following condition:

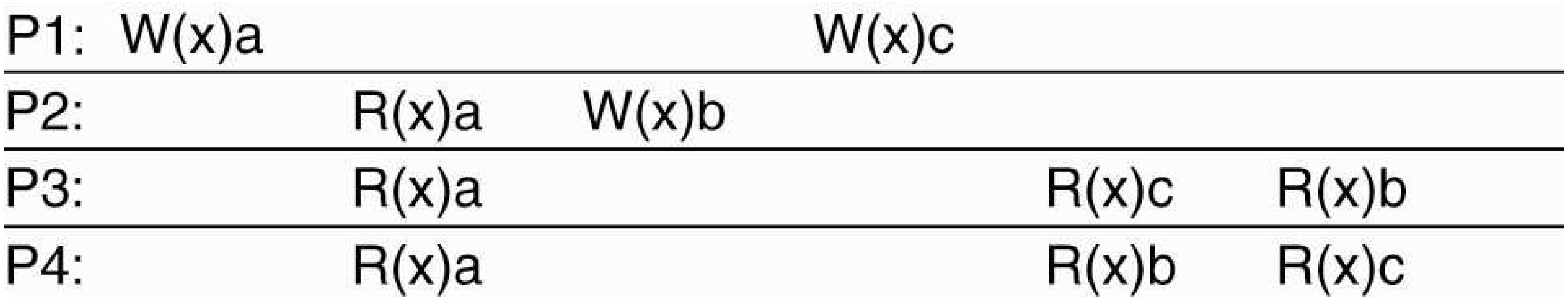
Writes that are potentially causally related …

- must be seen by all processes
- in the same order.

Concurrent writes …

- may be seen in a different order
- on different machines.

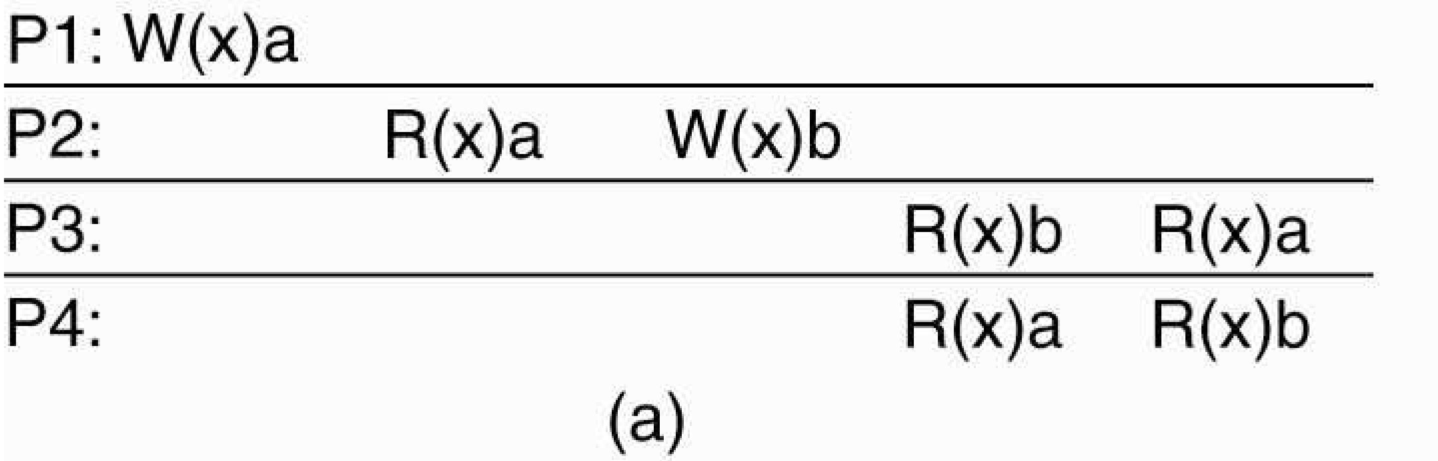
Causal Consistency (2)



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

Figure 7-8. This sequence is allowed with a causally-consistent store, but not with a sequentially consistent store.

Causal Consistency (3)



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

Figure 7-9. (a) A violation of a causally-consistent store.

Causal Consistency (4)

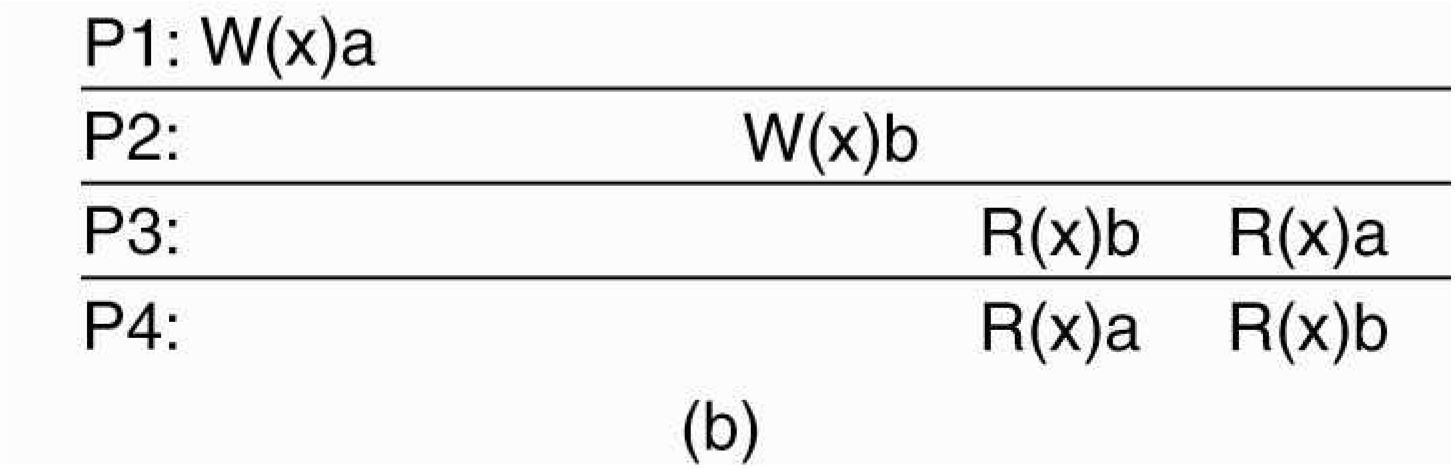


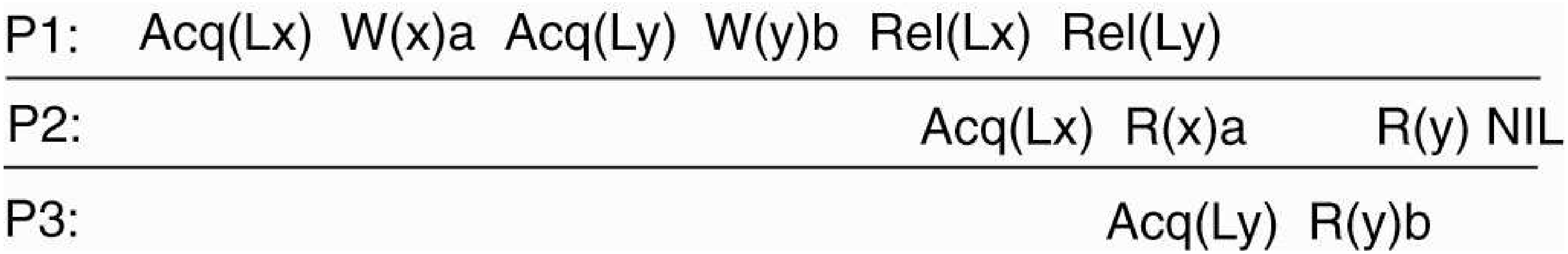
Figure 7-9. (b) A correct sequence of events in a causally-consistent store.

Grouping Operations (1)

Necessary criteria for correct synchronization:

- An acquire access of a synchronization variable, not allowed to perform until all updates to guarded shared data have been performed with respect to that process.
- Before exclusive mode access to synchronization variable by process is allowed to perform with respect to that process, no other process may hold synchronization variable, not even in nonexclusive mode.
- After exclusive mode access to synchronization variable has been performed, any other process' next nonexclusive mode access to that synchronization variable may not be performed until it has performed with respect to that variable's owner.

Grouping Operations (2)



P1: Acq(Lx) W(x)a Acq(Ly) W(y)b Rel(Lx) Rel(Ly)

P2: Acq(Lx) R(x)a R(y) NIL

P3: Acq(Ly) R(y)b

Figure 7-10. A valid event sequence for entry consistency.

Eventual Consistency

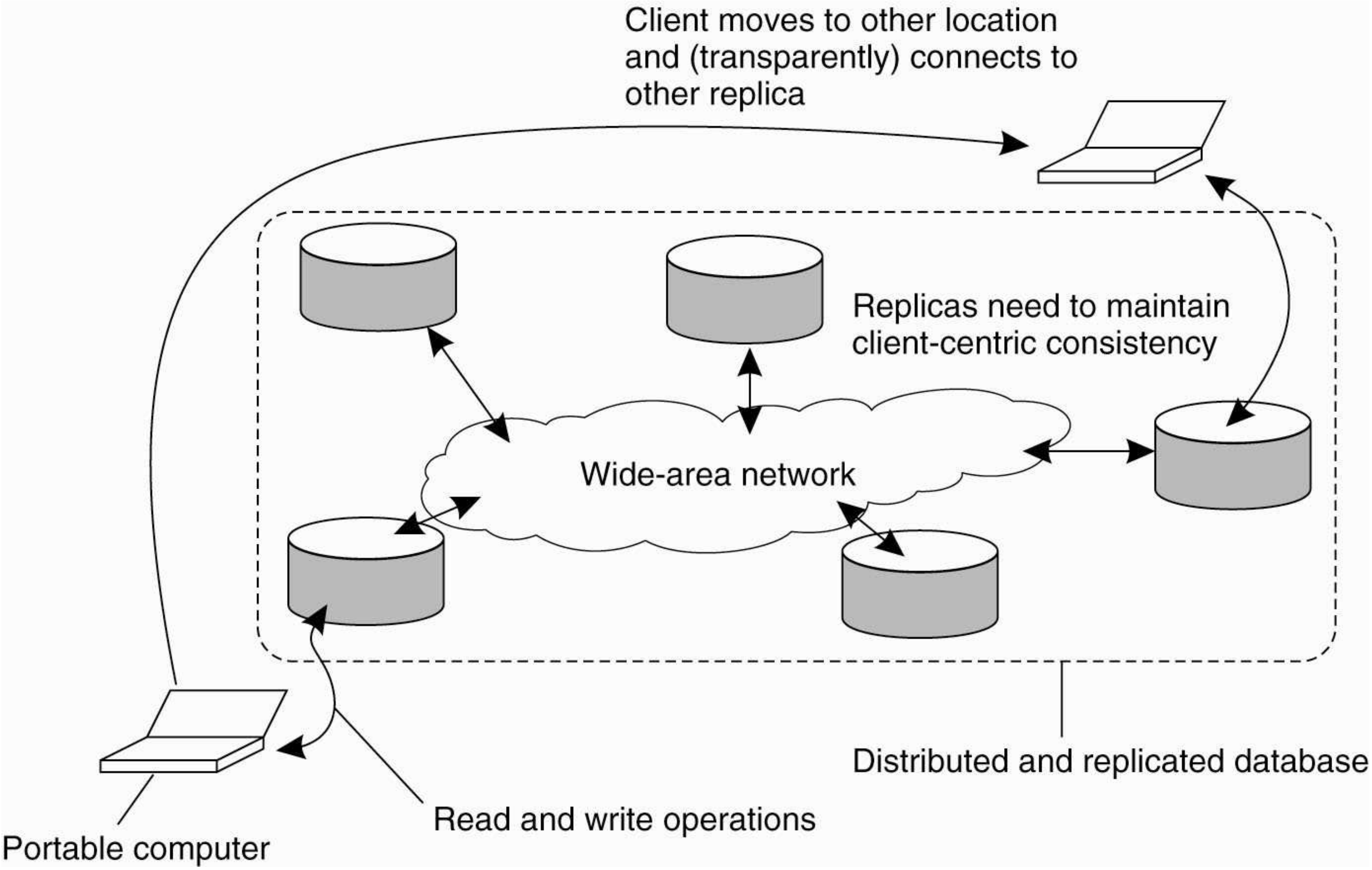

Client moves to other location and (transparently) connects to other replica
Replicas need to maintain client-centric consistency
Wide-area network
Distributed and replicated database
Read and write operations
Portable computer

Figure 7-11. The principle of a mobile user accessing different replicas of a distributed database.

Monotonic Reads (1)

A data store is said to provide monotonic-read consistency if the following condition holds:

If a process reads the value of a data item x …

- any successive read operation on x by that process
- will always return that same value
- or a more recent value.

Monotonic Reads (2)

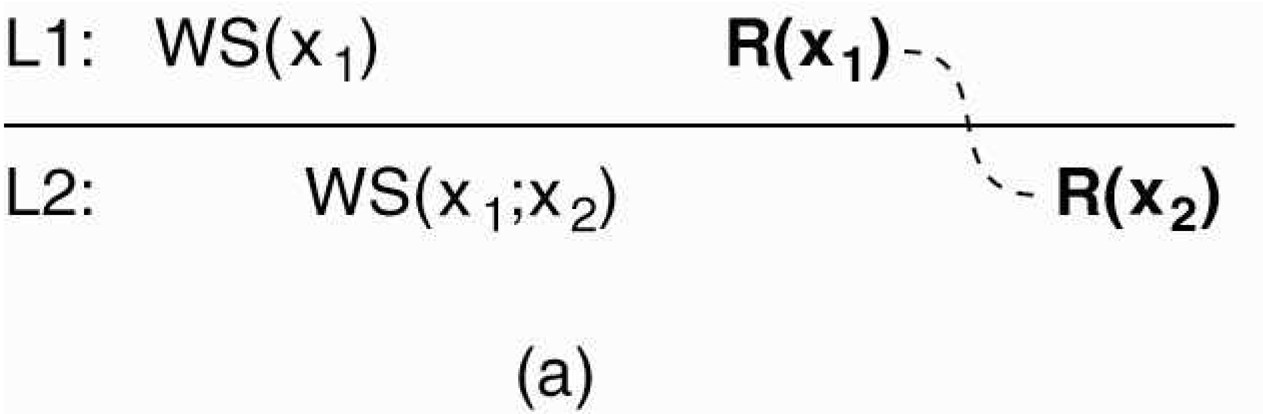


Figure 7-12. The read operations performed by a single process P at two different local copies of the same data store.
(a) A monotonic-read consistent data store.

Monotonic Reads (3)

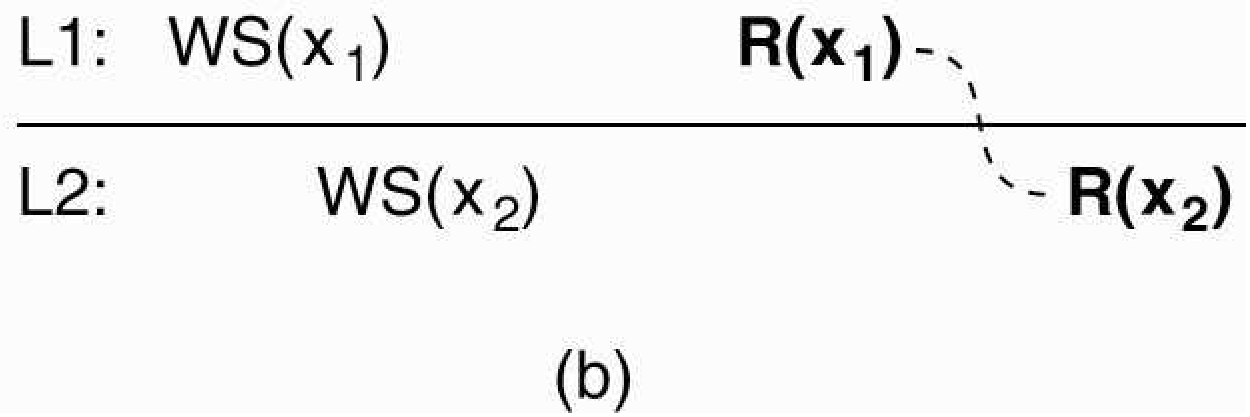


Figure 7-12. The read operations performed by a single process P at two different local copies of the same data store.
(b) A data store that does not provide monotonic reads.

Monotonic Writes (1)

In a monotonic-write consistent store, the following condition holds:

A write operation by a process on a data item x …

- is completed before any successive write operation on x
- by the same process.

Monotonic Writes (2)

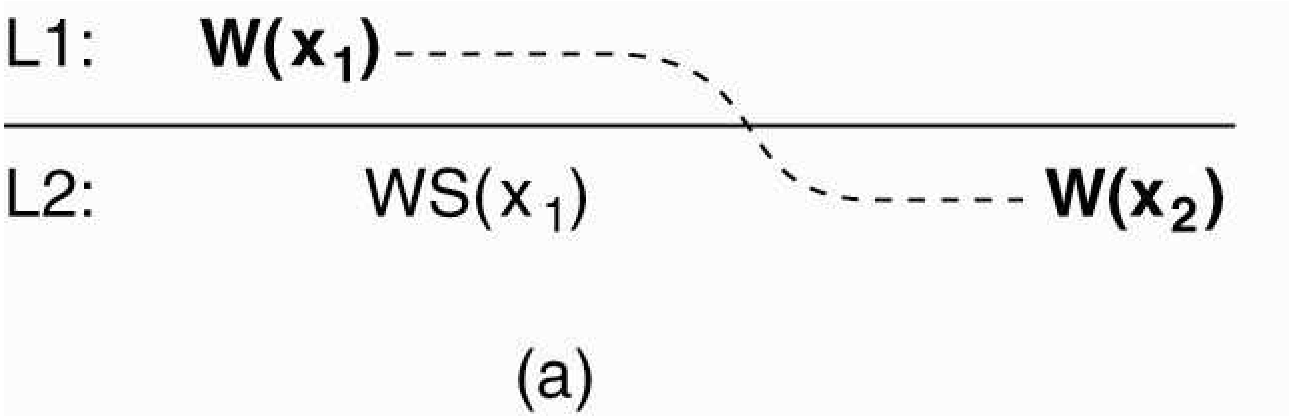


Figure 7-13. The write operations performed by a single process P at two different local copies of the same data store. (a) A monotonic-write consistent data store.

Monotonic Writes (3)

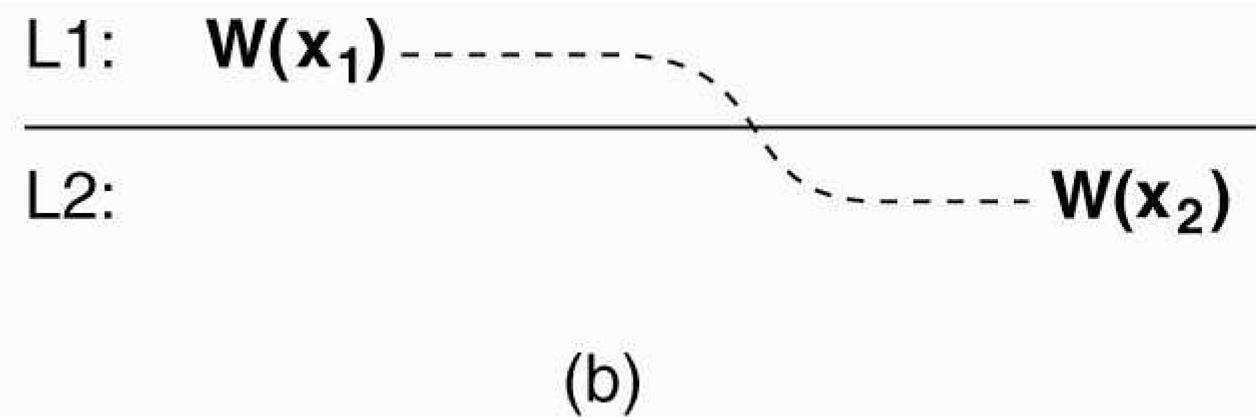


Figure 7-13. The write operations performed by a single process P at two different local copies of the same data store. (b) A data store that does not provide monotonic-write consistency.

Read Your Writes (1)

A data store is said to provide read-your-writes consistency, if the following condition holds:

The effect of a write operation by a process on data item x …

- will always be seen by a successive read operation on x
- by the same process.

Read Your Writes (2)

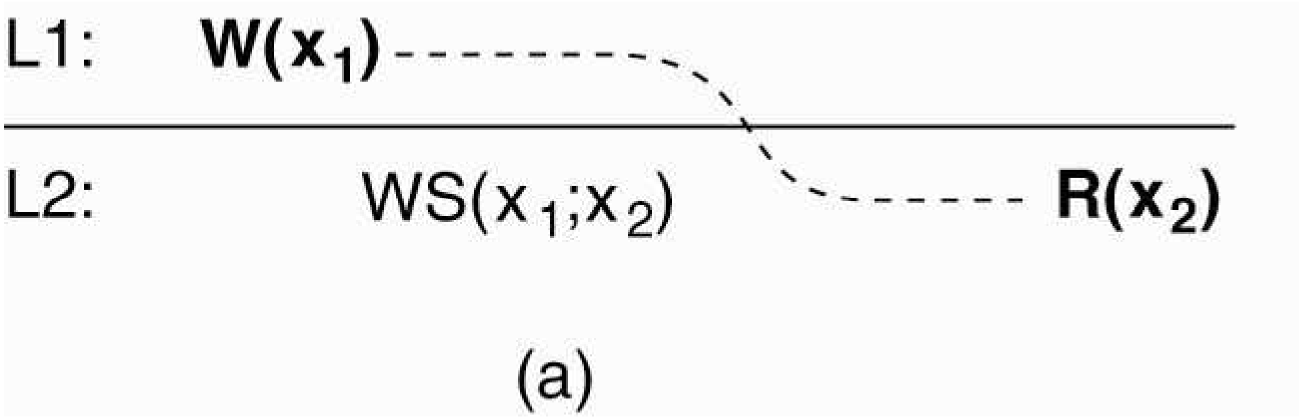


Figure 7-14. (a) A data store that provides read-your-writes consistency.

Read Your Writes (3)

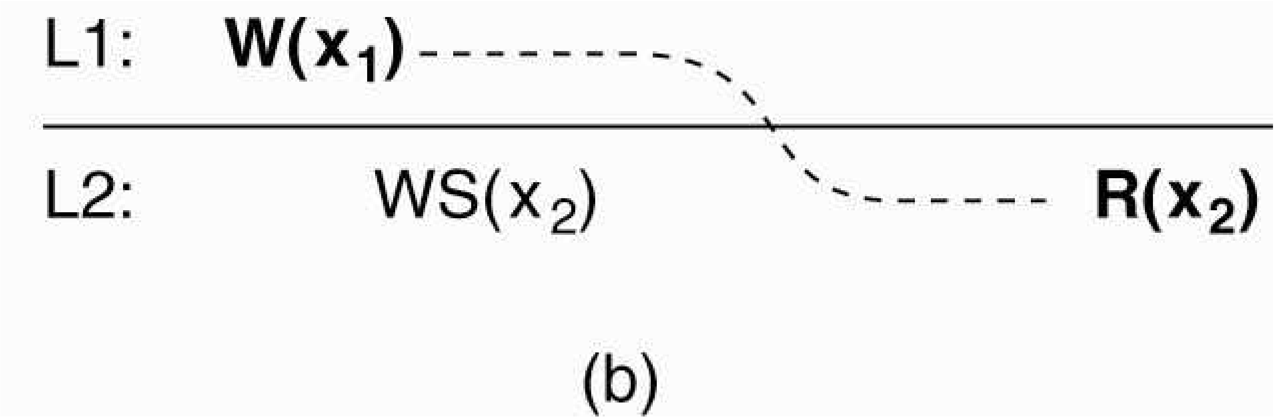


Figure 7-14. (b) A data store that does not.

Writes Follow Reads (1)

A data store is said to provide writes-follow-reads consistency, if the following holds:

A write operation by a process …

- on a data item x following a previous read operation on x by the same process …
- is guaranteed to take place on the same or a more recent value of x that was read.

Writes Follow Reads (2)

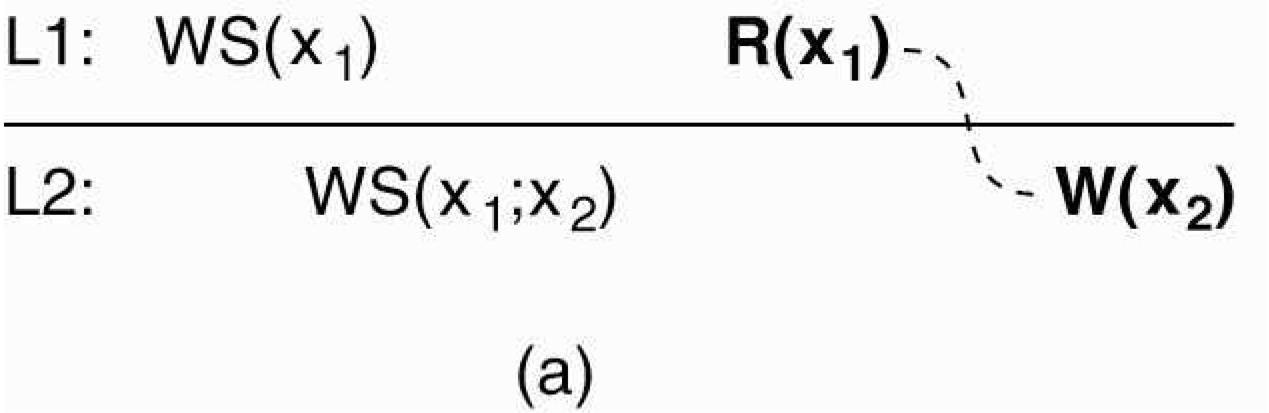


Figure 7-15. (a) A writes-follow-reads consistent data store.

A. A. Pourhaji Kazem, Spring 2009

Writes Follow Reads (3)

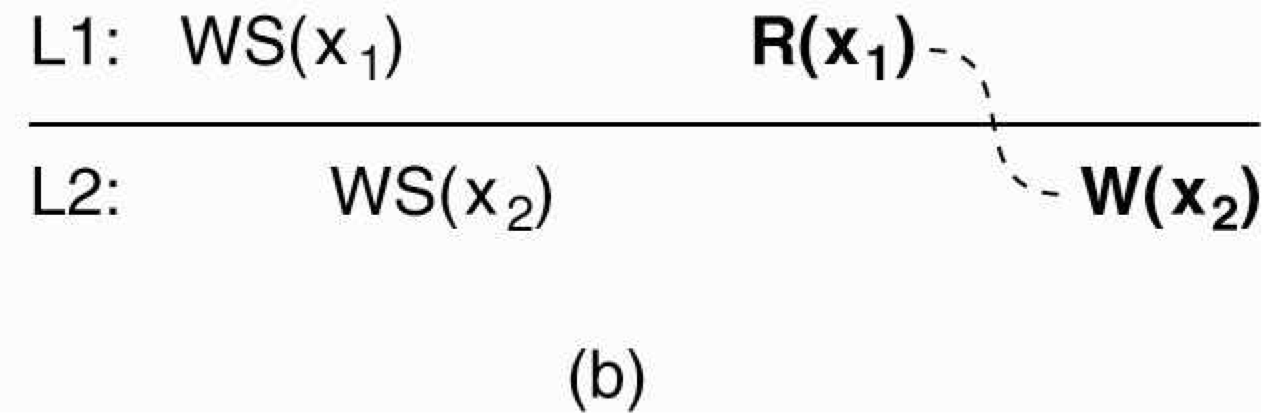


Figure 7-15. (b) A data store that does not provide writes-follow-reads consistency.

Replica-Server Placement

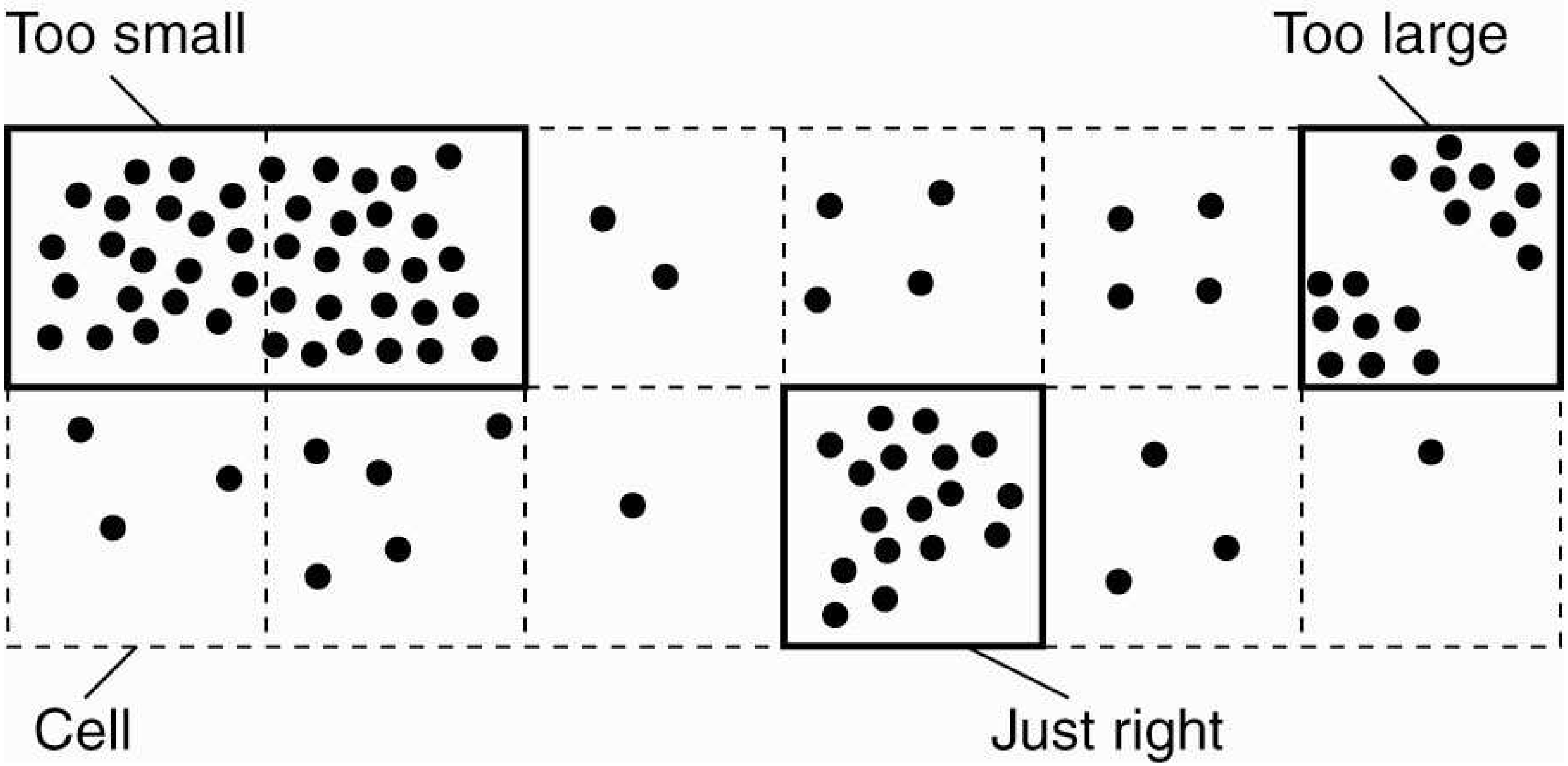


Figure 7-16. Choosing a proper cell size for server placement.

Content Replication and Placement

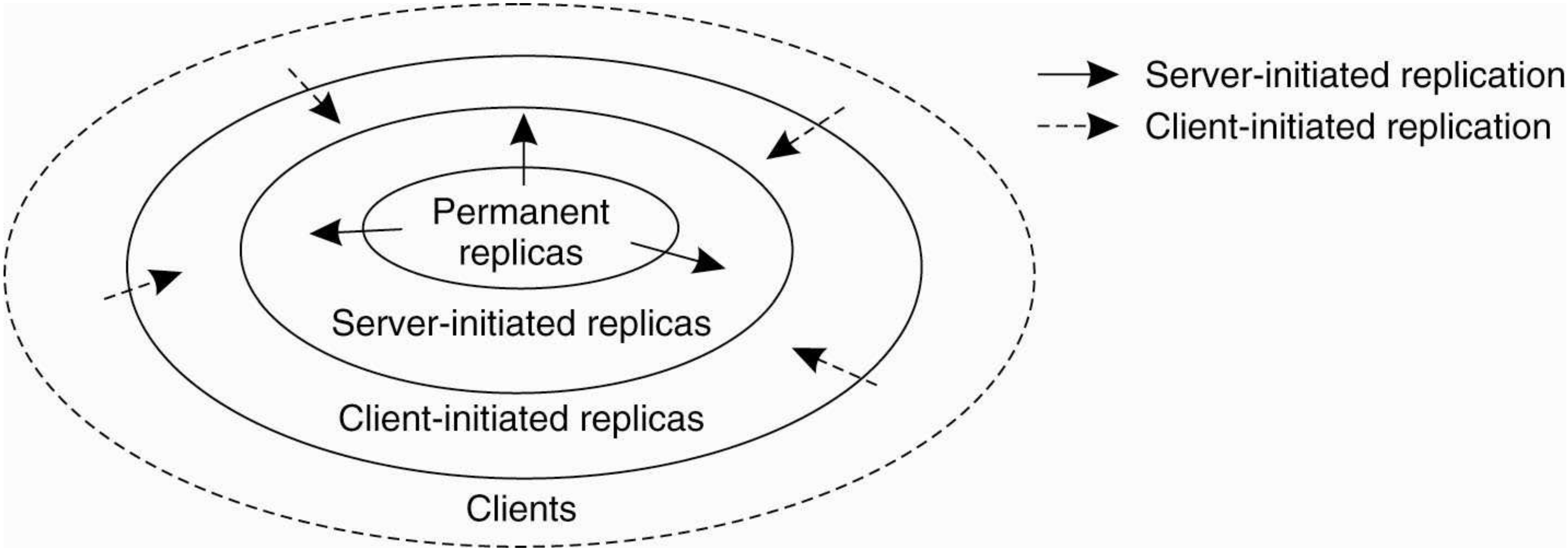

Server-initiated replication
Client-initiated replication
Permanent replicas
Server-initiated replicas
Client-initiated replicas
Clients

Figure 7-17. The logical organization of different kinds of copies of a data store into three concentric rings.

Server-Initiated Replicas

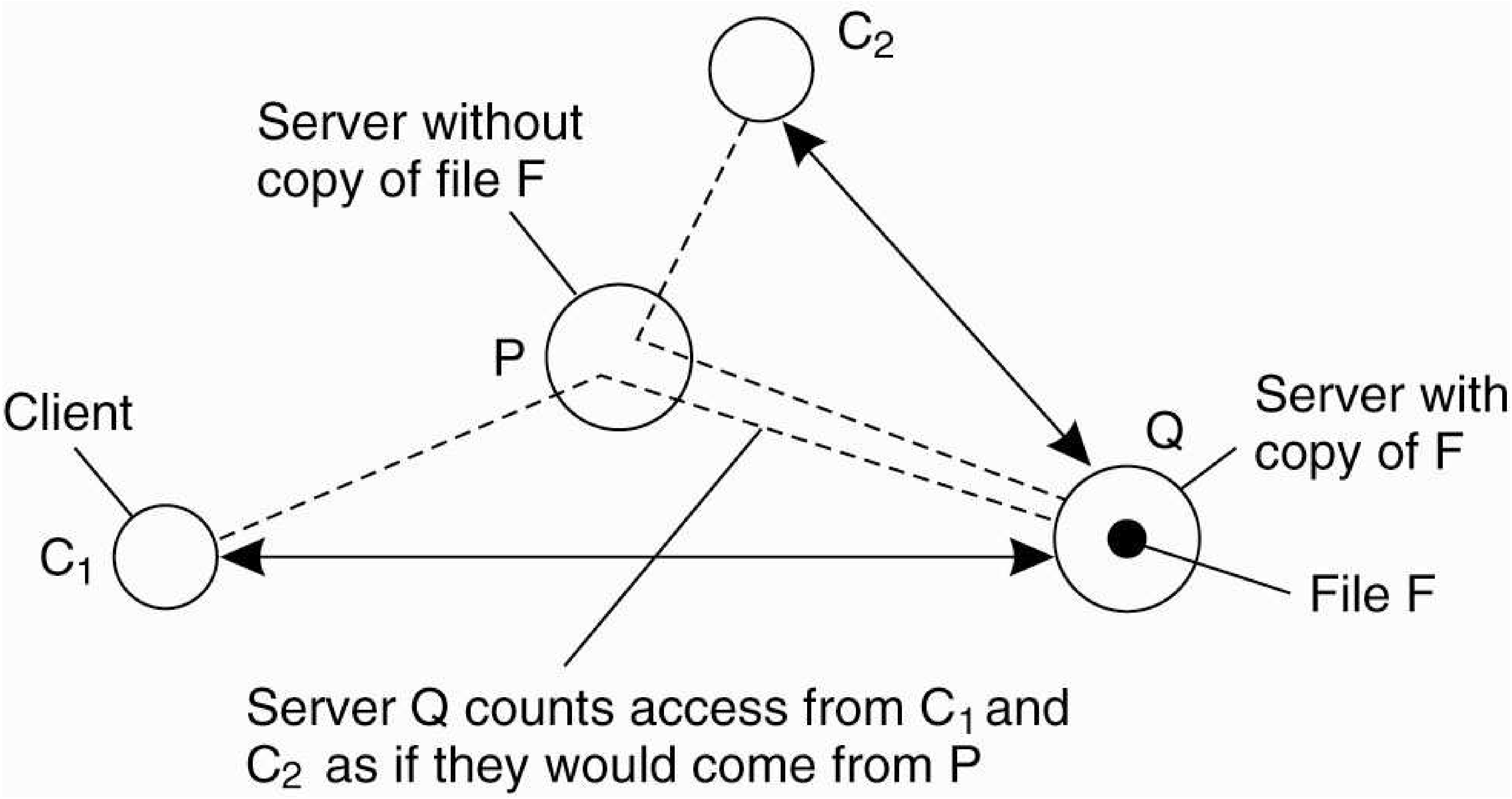

C$_2$
Server without copy of file F
P
Client
C$_1$
Q
Server with copy of F
File F
Server Q counts access from C$_1$ and C$_2$ as if they would come from P

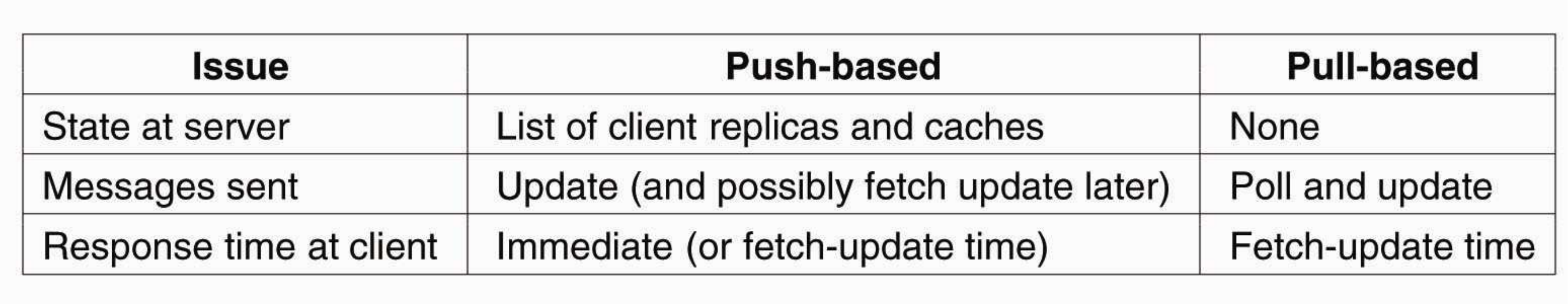
Figure 7-18. Counting access requests from different clients.

State versus Operations

Possibilities for what is to be propagated:

1. Propagate only a notification of an update.
2. Transfer data from one copy to another.
3. Propagate the update operation to other copies.

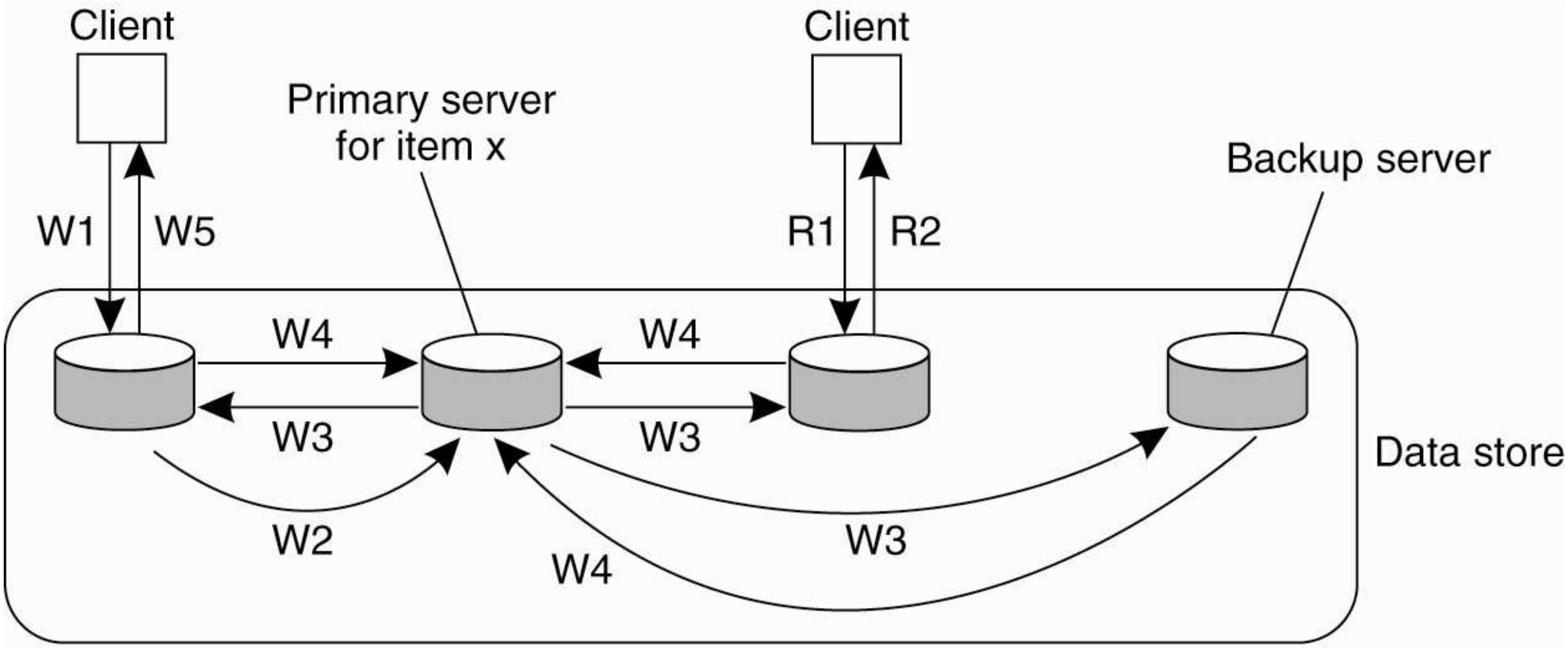
Pull versus Push Protocols



Issue	Push-based	Pull-based
State at server	List of client replicas and caches	None
Messages sent	Update (and possibly fetch update later)	Poll and update
Response time at client	Immediate (or fetch-update time)	Fetch-update time

Figure 7-19. A comparison between push-based and pull-based protocols in the case of multiple-client, single-server systems.

Remote-Write Protocols

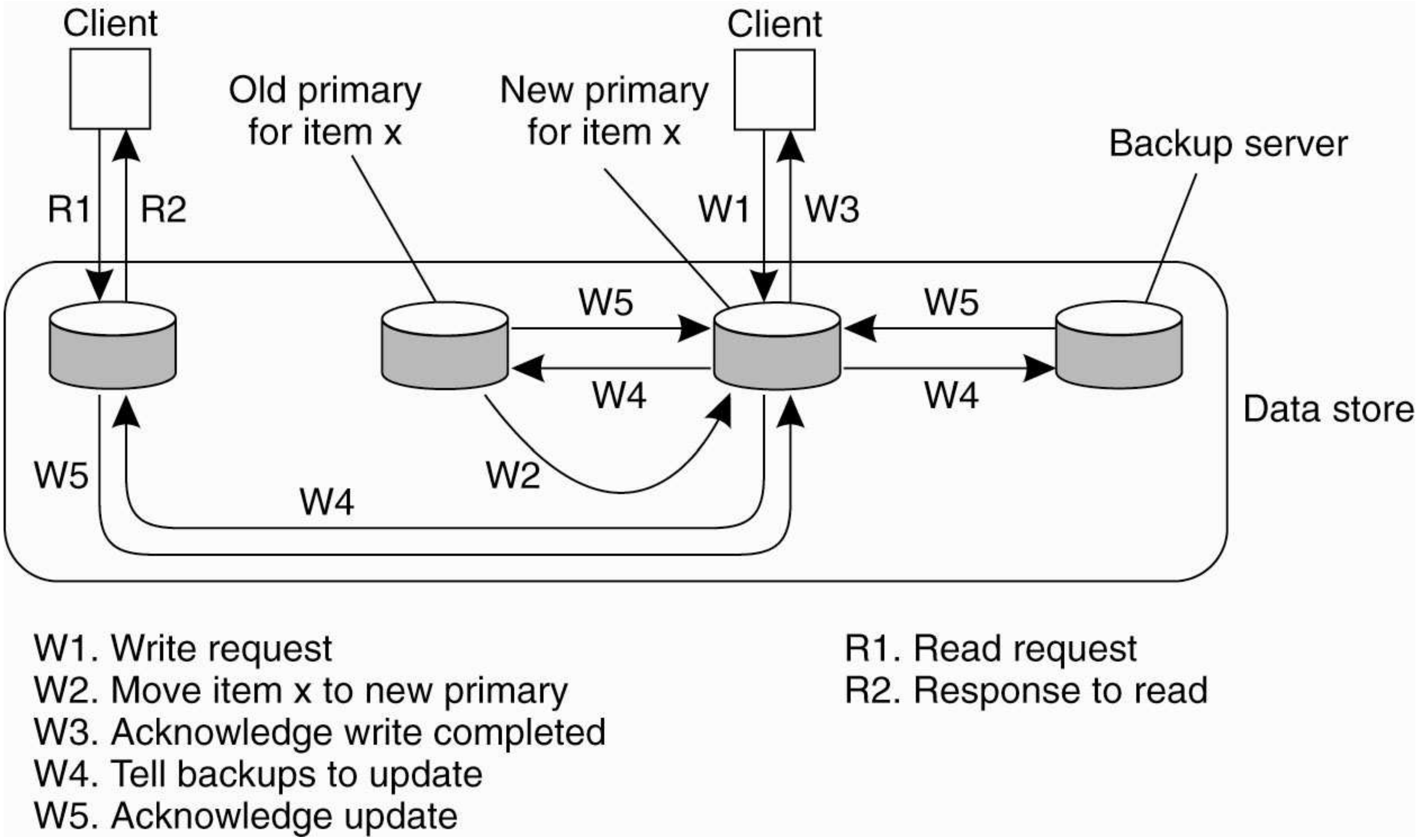


W1. Write request
W2. Forward request to primary
W3. Tell backups to update
W4. Acknowledge update
W5. Acknowledge write completed

R1. Read request
R2. Response to read

Figure 7-20. The principle of a primary-backup protocol.

Local-Write Protocols



W1. Write request
W2. Move item x to new primary
W3. Acknowledge write completed
W4. Tell backups to update
W5. Acknowledge update

R1. Read request
R2. Response to read

Figure 7-21. Primary-backup protocol in which the primary migrates to the process wanting to perform an update.

Quorum-Based Protocols

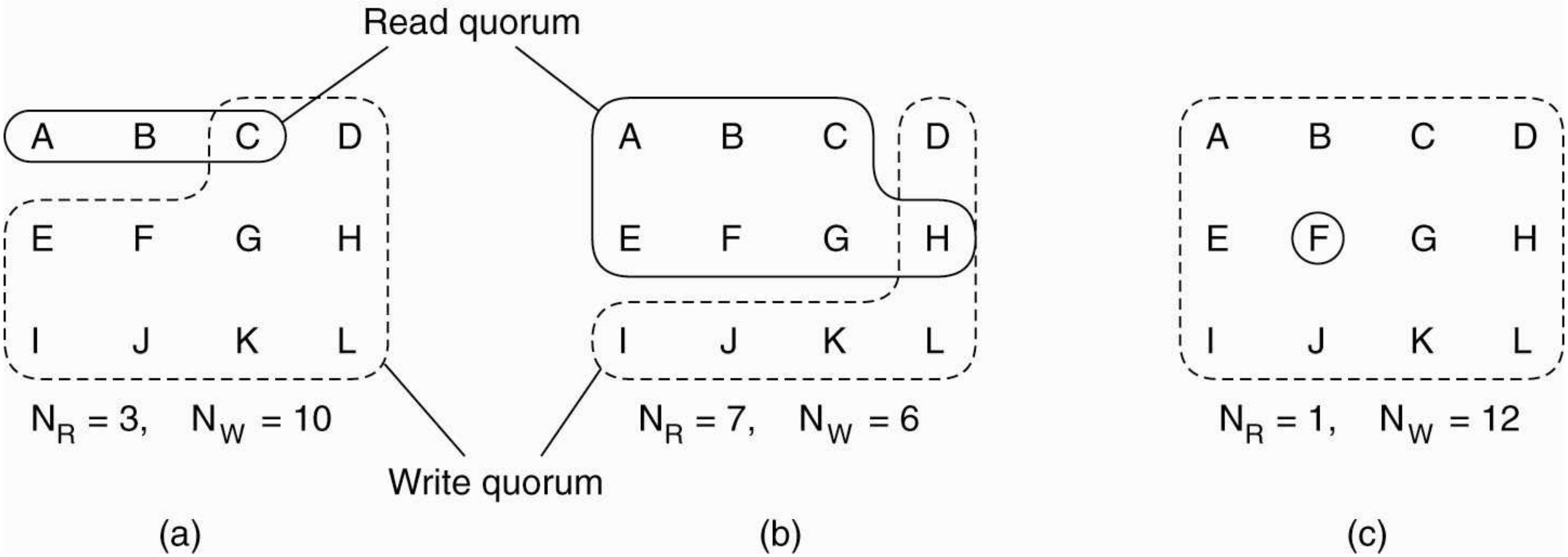


Figure 7-22. Three examples of the voting algorithm. (a) A correct choice of read and write set. (b) A choice that may lead to write-write conflicts. (c) A correct choice, known as ROWA (read one, write all).