Systèmes Distribués

Master MIAGE 1

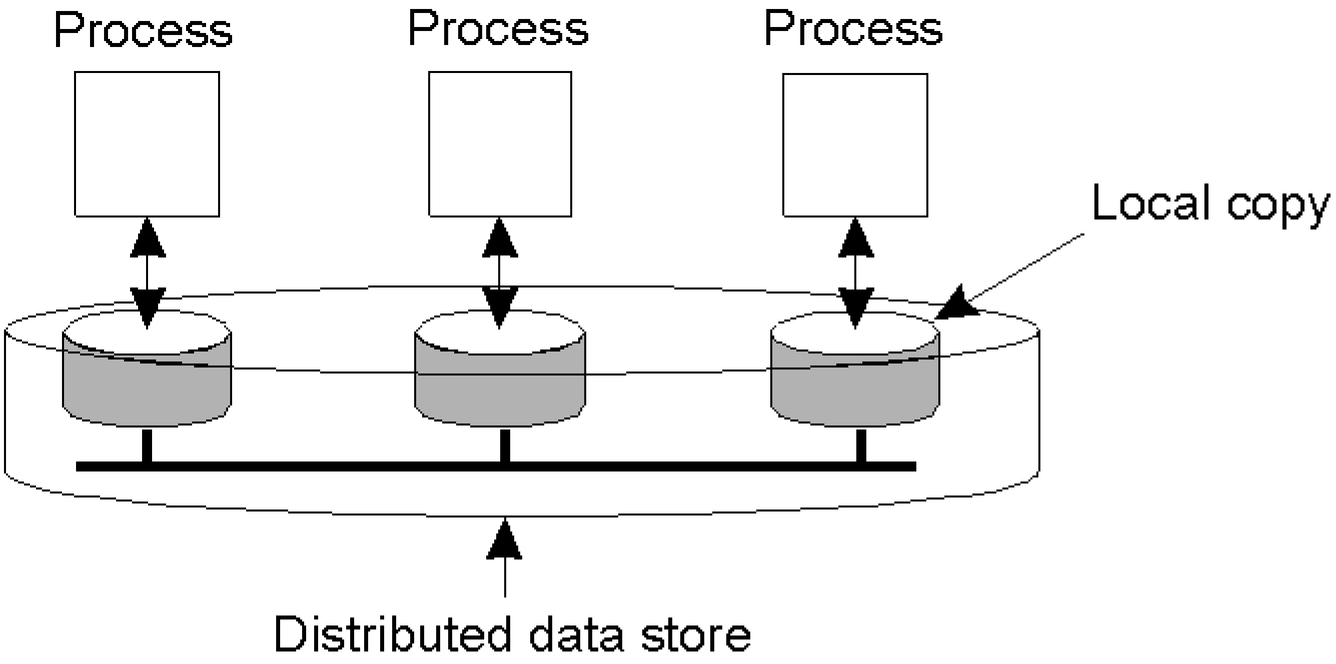
Andrea G. B. Tettamanzi
Université de Nice Sophia Antipolis
Département Informatique
andrea.tettamanzi@unice.fr

CM - Séance 6 – Partie I

Réplication et contrôle de cohérence

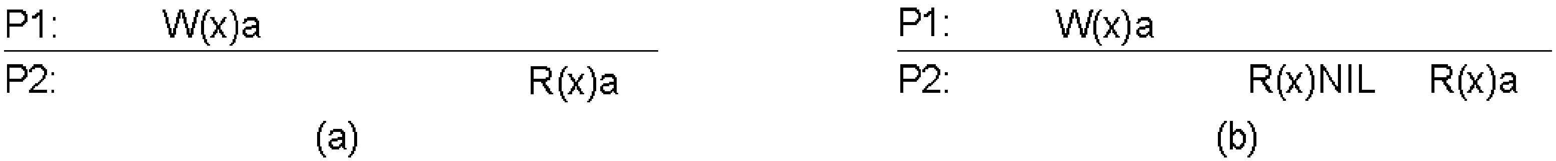
(chapitre 7)

Data-Centric Consistency Models



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

Strict Consistency



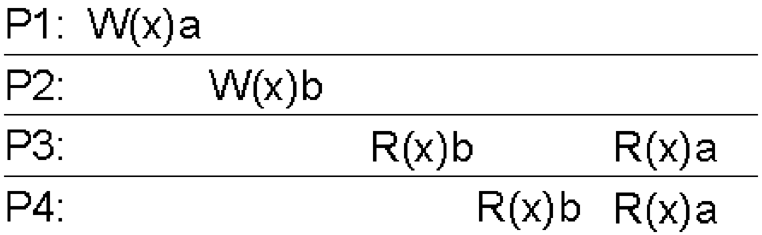
(a)

(b)

Behavior of two processes, operating on the same data item.

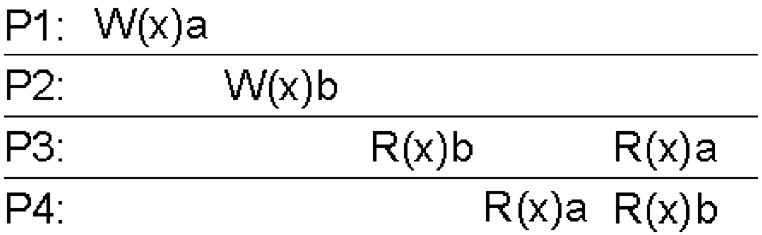
- A strictly consistent store.
- A store that is not strictly consistent.

Linearizability and Sequential Consistency (1)



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

(a)



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

(b)

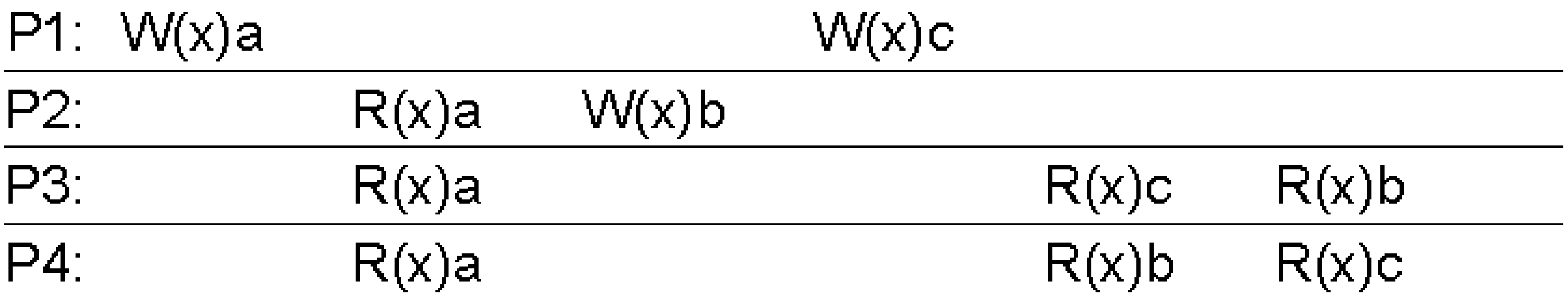
a) A sequentially consistent data store.

b) A data store that is not sequentially consistent.

Causal Consistency (1)

- Necessary 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.

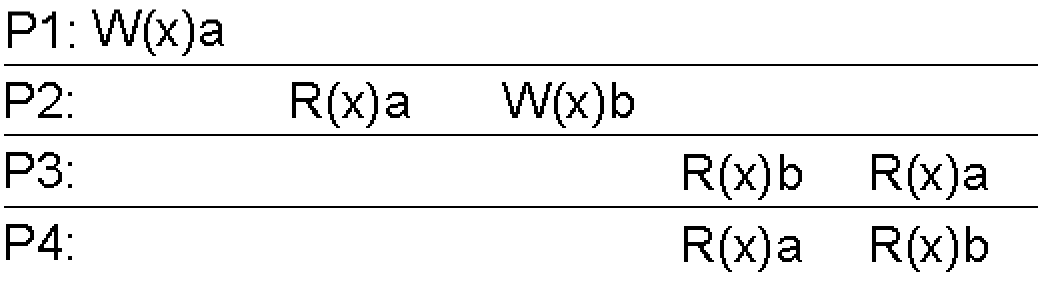
Causal Consistency (2)



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

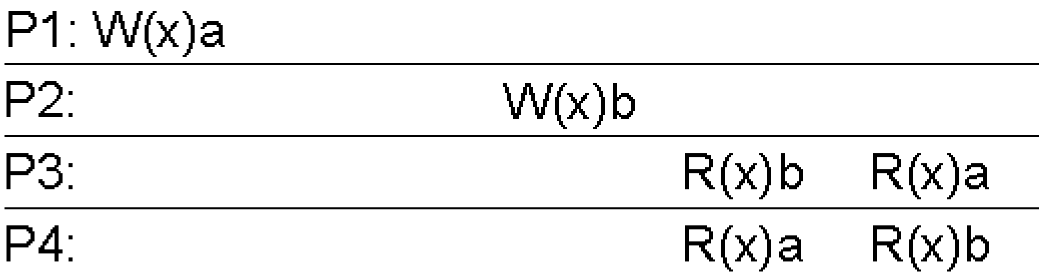
This sequence is allowed with a causally-consistent store, but not with sequentially or strictly consistent store.

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)



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

(b)

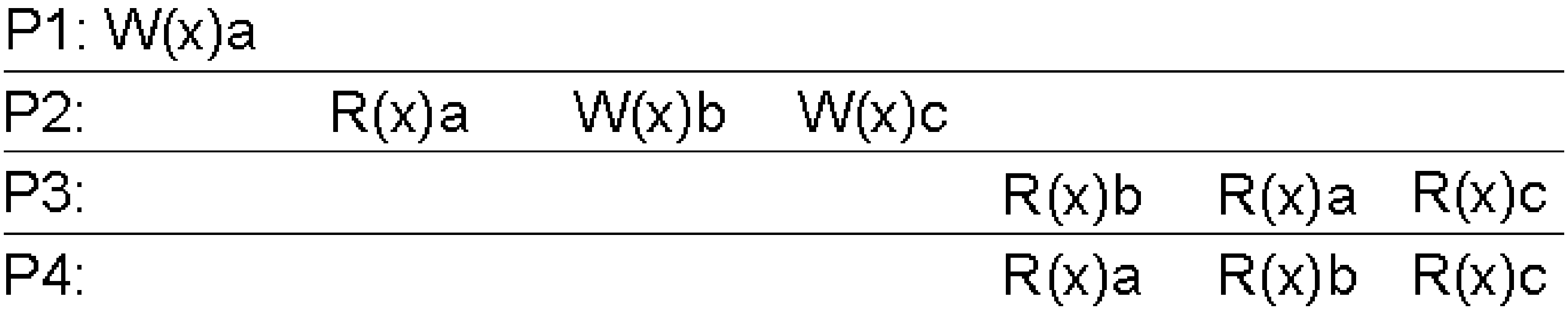
a) A violation of a causally-consistent store.

b) A correct sequence of events in a causally-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

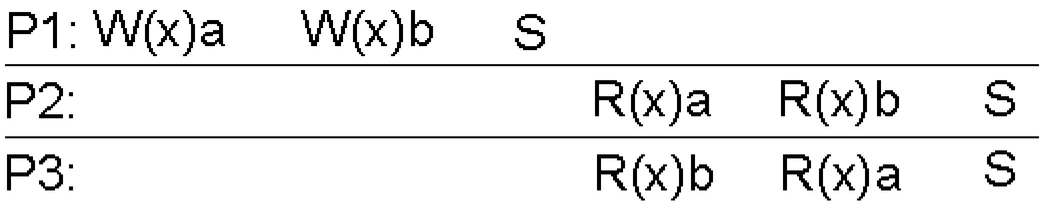
Modèles avec synchronisation

- Il n'est pas toujours nécessaire d'assurer la cohérence à tout instant.
- D'où l'idée de la forcer uniquement lorsqu'elle est nécessaire, à l'aide de variables et d'opérations de synchronisation.
- Dans ces modèles, des instructions de synchronisation permettent d'appliquer temporairement une cohérence plus forte.
- Ces instructions sont généralement des « barrières », qui assurent que les accès précédents ont été rendus visibles de tous les acteurs.
- Ceci permet de spécifier de manière fine les variables à protéger, en utilisant des informations sur les accès prévus. La motivation est de réduire le nombre d'opérations de mise en cohérence.

Weak Consistency (1)

- Properties:
  - Accesses to synchronization variables associated with a data store are sequentially consistent
  - No operation on a synchronization variable is allowed to be performed until all previous writes have been completed everywhere
  - No read or write operation on data items are allowed to be performed until all previous operations to synchronization variables have been performed.

Weak Consistency (3)

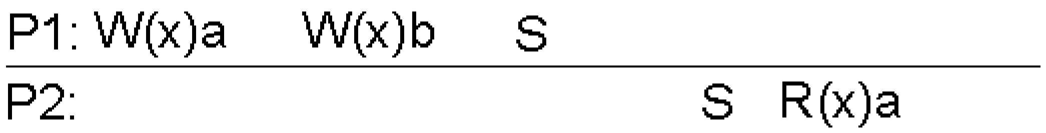


P1: W(x)a W(x)b S

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

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

(a)



P1: W(x)a W(x)b S

P2: S R(x)a

(b)

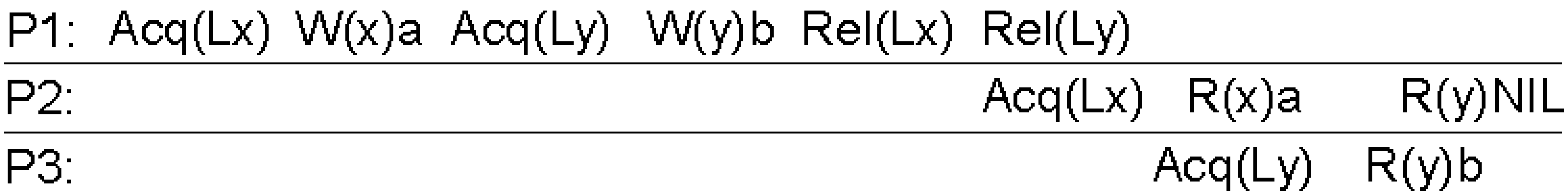
a) A valid sequence of events for weak consistency.

b) An invalid sequence for weak consistency.

Entry Consistency (1)

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

Entry Consistency (1)



```
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
```

A valid event sequence for entry consistency.

Summary of Consistency Models

Consistency	Description
Strict	Absolute time ordering of all shared accesses matters.
Linearizability	All processes must see all shared accesses in the same order. Accesses are furthermore ordered according to a (nonunique) global timestamp
Sequential	All processes see all shared accesses in the same order. Accesses are not ordered in time
Causal	All processes see causally-related shared accesses in the same order.
FIFO	All processes see writes from each other in the order they were used. Writes from different processes may not always be seen in that order

(a)

Consistency	Description
Weak	Shared data can be counted on to be consistent only after a synchronization is done
Release	Shared data are made consistent when a critical region is exited
Entry	Shared data pertaining to a critical region are made consistent when a critical region is entered.

(b)

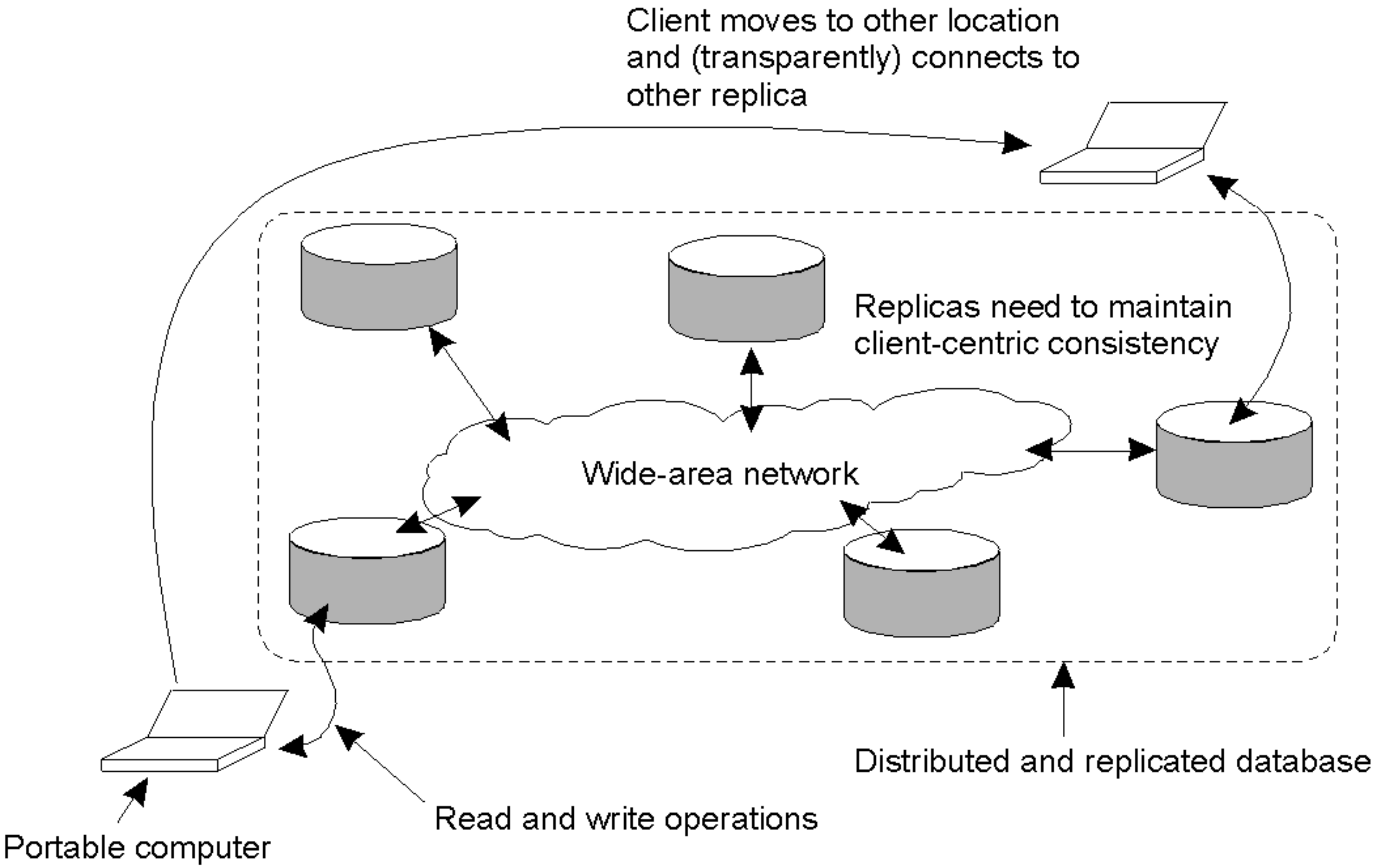
a) Consistency models not using synchronization operations.
b) Models with synchronization operations.

Client-Centric Consistency

Cohérence centrée sur les clients

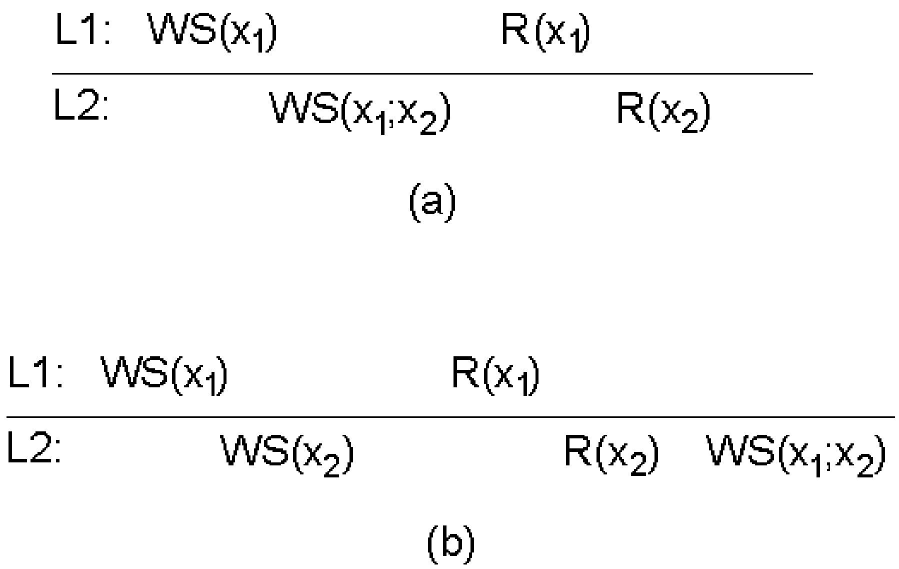
- Dans beaucoup de systèmes, la concurrence est présente en mesure limitée
- Par exemple,
  - Un seul (ou un nombre très petit de) processus fait des mises à jour des données
  - Les autres processus se limitent à accéder en lecture
  - Pas de conflits Write-Write
  - Uniquement de conflits Read-Write
- Un retard de la propagation des mises à jours peut être acceptable.
- => Eventual consistency (= cohérence à terme)

Eventual Consistency w/ Mobile Clients


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

The principle of a mobile user accessing different replicas of a distributed database.

Monotonic Reads

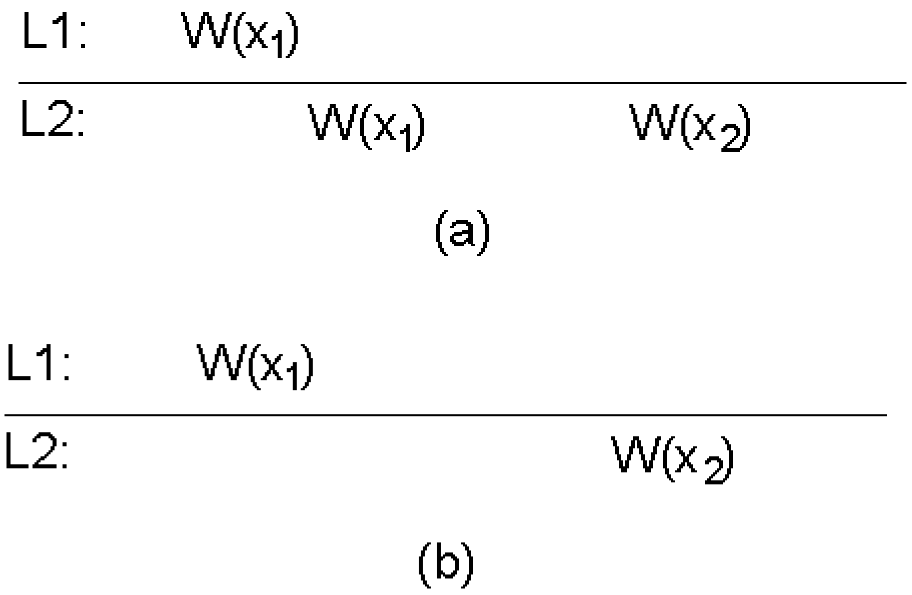

L1: WS(x_1) R(x_1)
L2: WS(x_1;x_2) R(x_2)
(a)
L1: WS(x_1) R(x_1)
L2: WS(x_2) R(x_2) WS(x_1;x_2)
(b)

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

b) A data store that does not provide monotonic reads.

Monotonic Writes



(a)

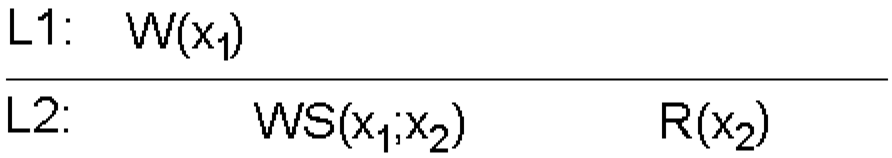
(b)

The write operations performed by a single process *P* at two different local copies of the same data store

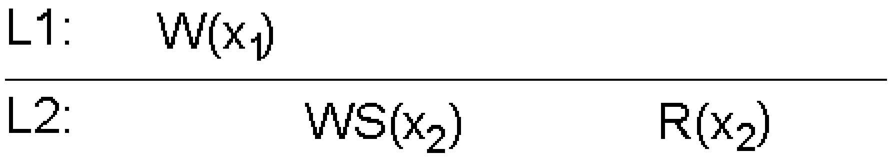
a) A monotonic-write consistent data store.

b) A data store that does not provide monotonic-write consistency.

Read Your Writes



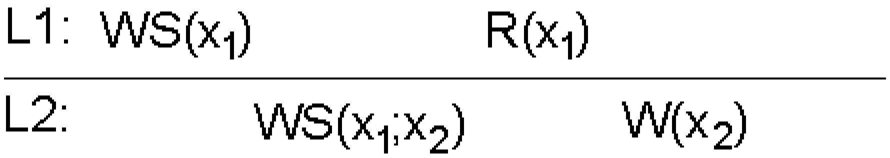
(a)



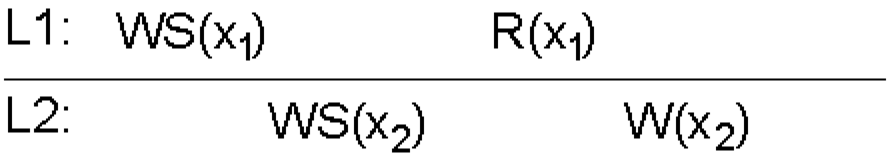
(b)

a) A data store that provides read-your-writes consistency.
b) A data store that does not.

Writes Follow Reads



(a)

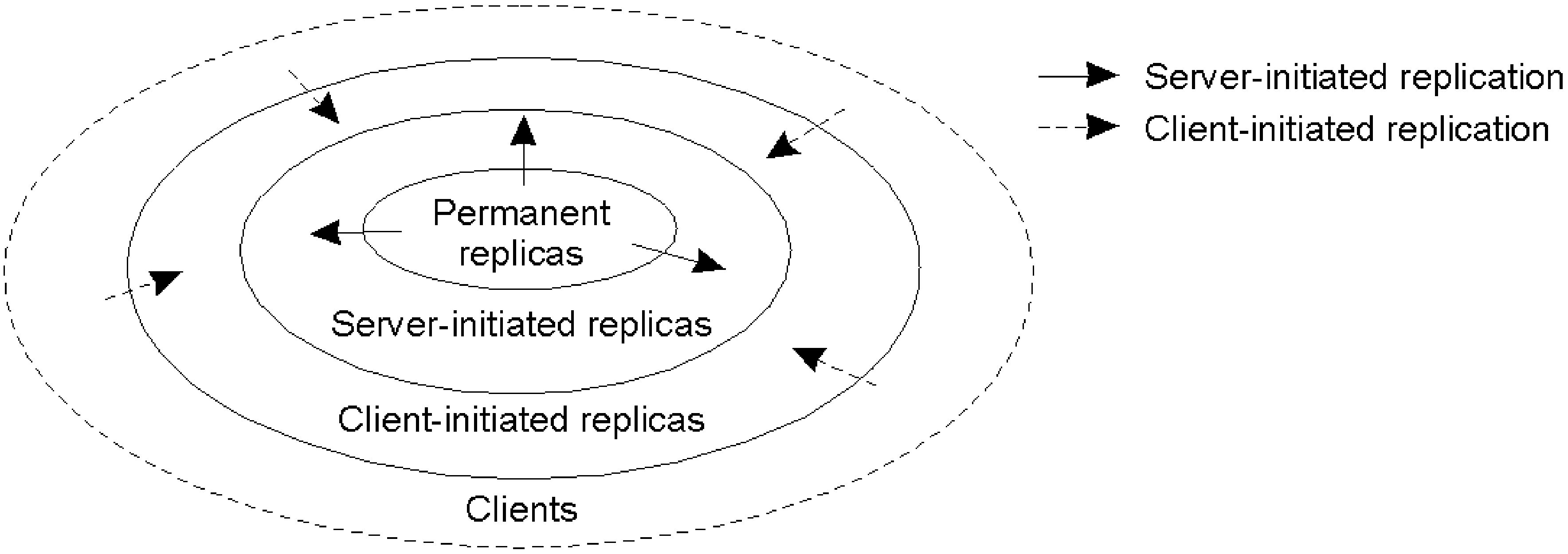


(b)

a) A writes-follow-reads consistent data store

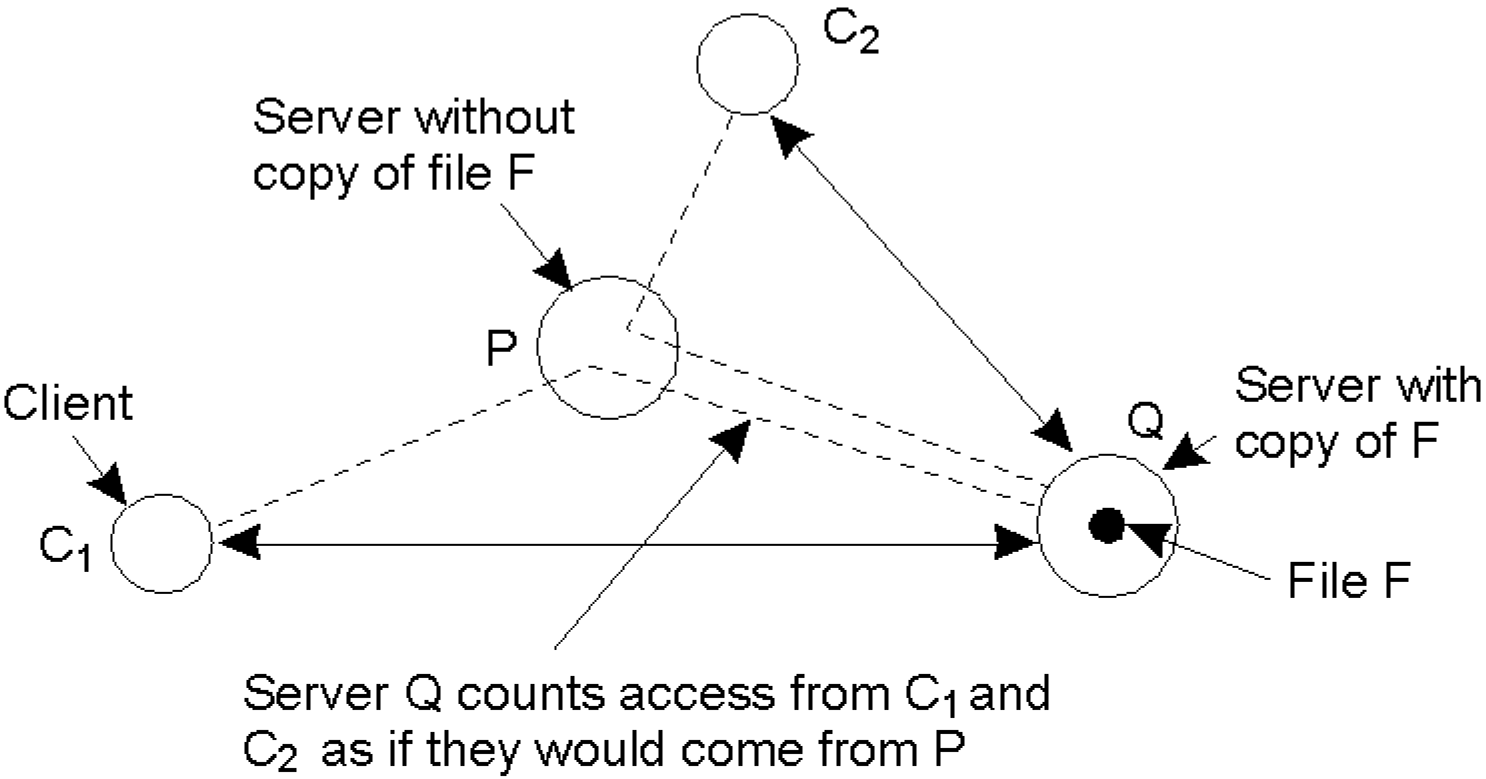
b) A data store that does not provide writes-follow-reads consistency

Replica Placement


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

The logical organization of different kinds of copies of a data store into three concentric rings.

Server-Initiated Replicas



Counting access requests from different clients.

Client-Initiated Replicas

- Client cache
- Cache hit
- Client cache placement

Content Distribution

- Three possibilities:
  - Propagate notification only (invalidation)
  - Transfer data
  - Propagate update operations (active replication)

Pull versus Push Protocols

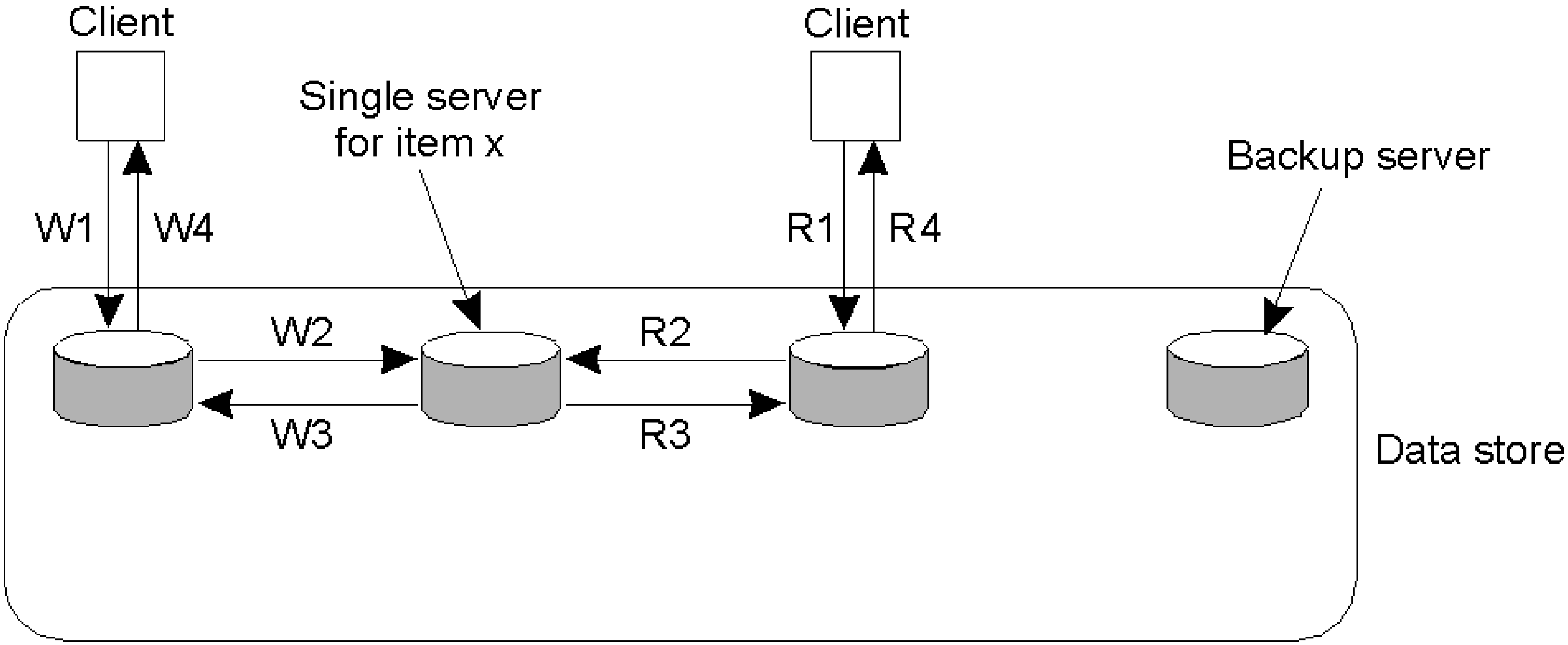
Issue	Push-based	Pull-based
State of server	List of client replicas and caches	None
Messages sent	Update (and possibly fetch update later)	Poll and update
Response time at client	Immediate (or fetch-update time)	Fetch-update time

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

Consistency Protocols

- Implementation of a model
- Complicated protocols hardly adopted
- Locking and transactions
- Primary-based protocols (primary copy)

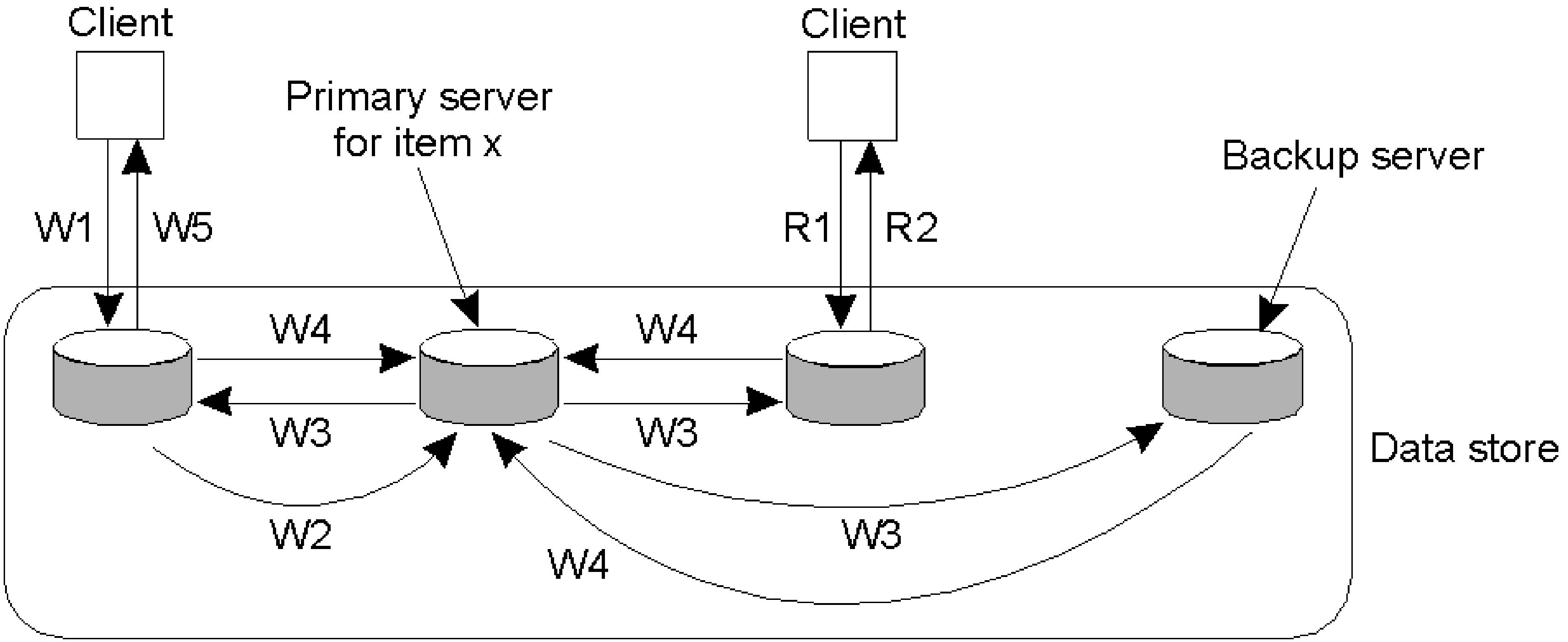
Remote-Write Protocols (1)



W1. Write request
W2. Forward request to server for x
W3. Acknowledge write completed
W4. Acknowledge write completed

R1. Read request
R2. Forward request to server for x
R3. Return response
R4. Return response

Remote-Write Protocols (2)

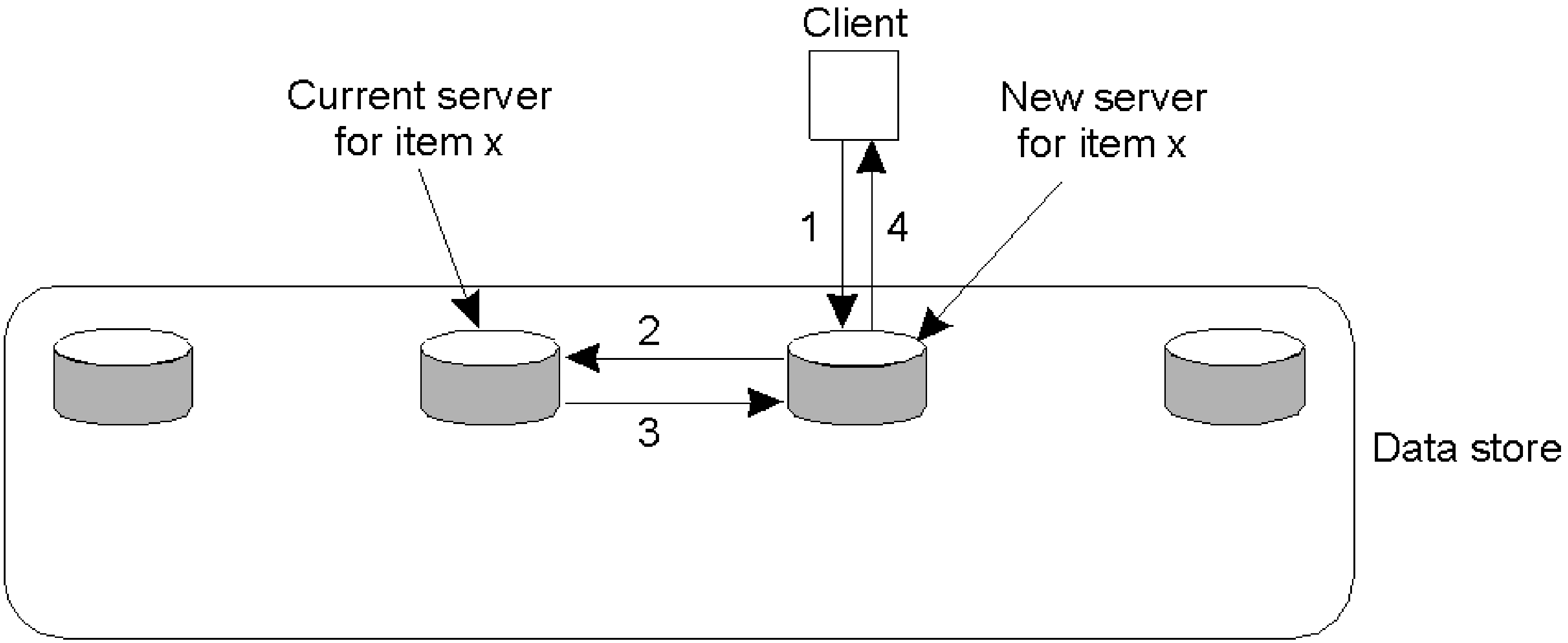


W1. Write request
W2. Forward request to primary
W3. Tell backups to update
W4. Acknowledge update
W5. Acknowledge write completed

R1. Read request
R2. Response to read

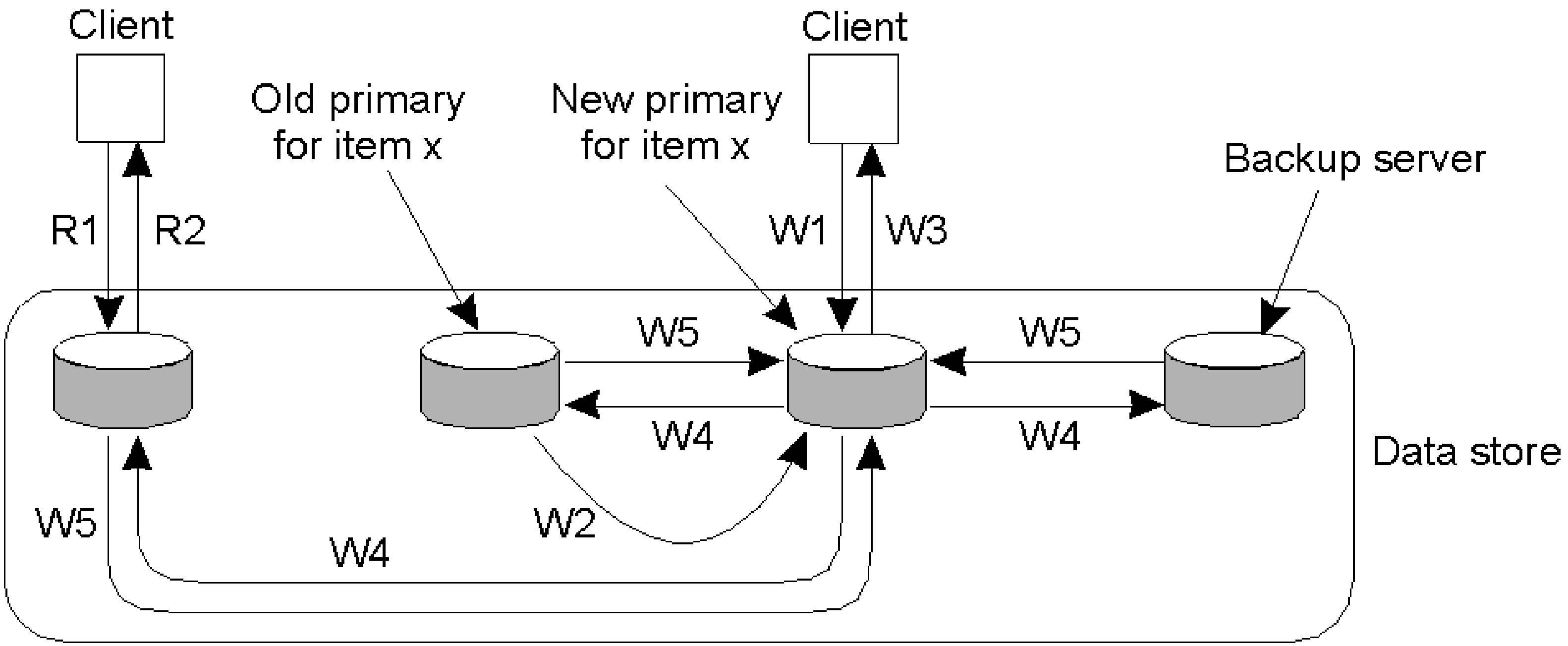
The principle of primary-backup protocol.

Local-Write Protocols (1)



1. Read or write request
2. Forward request to current server for x
3. Move item x to client's server
4. Return result of operation on client's server

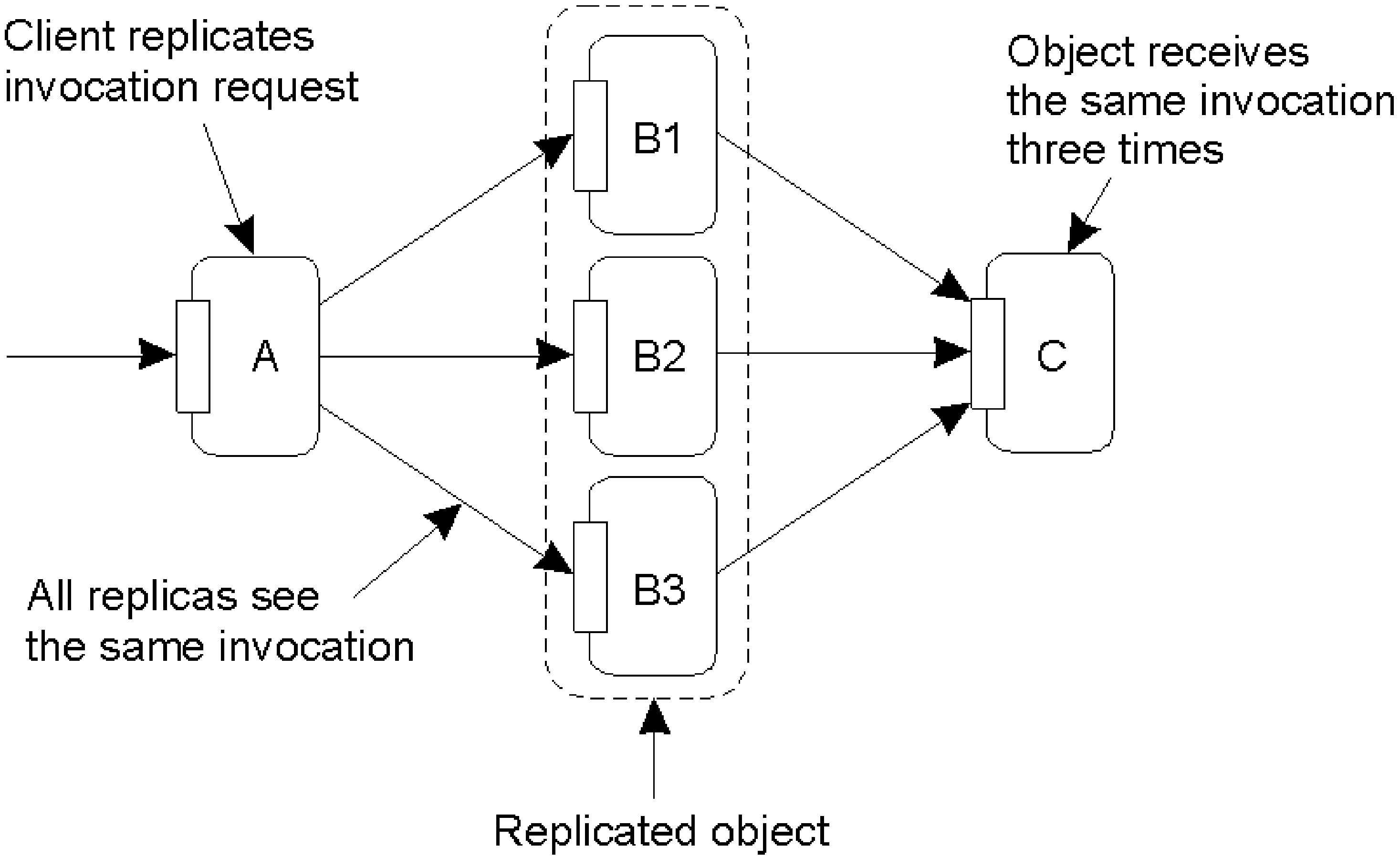
Local-Write Protocols (2)



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

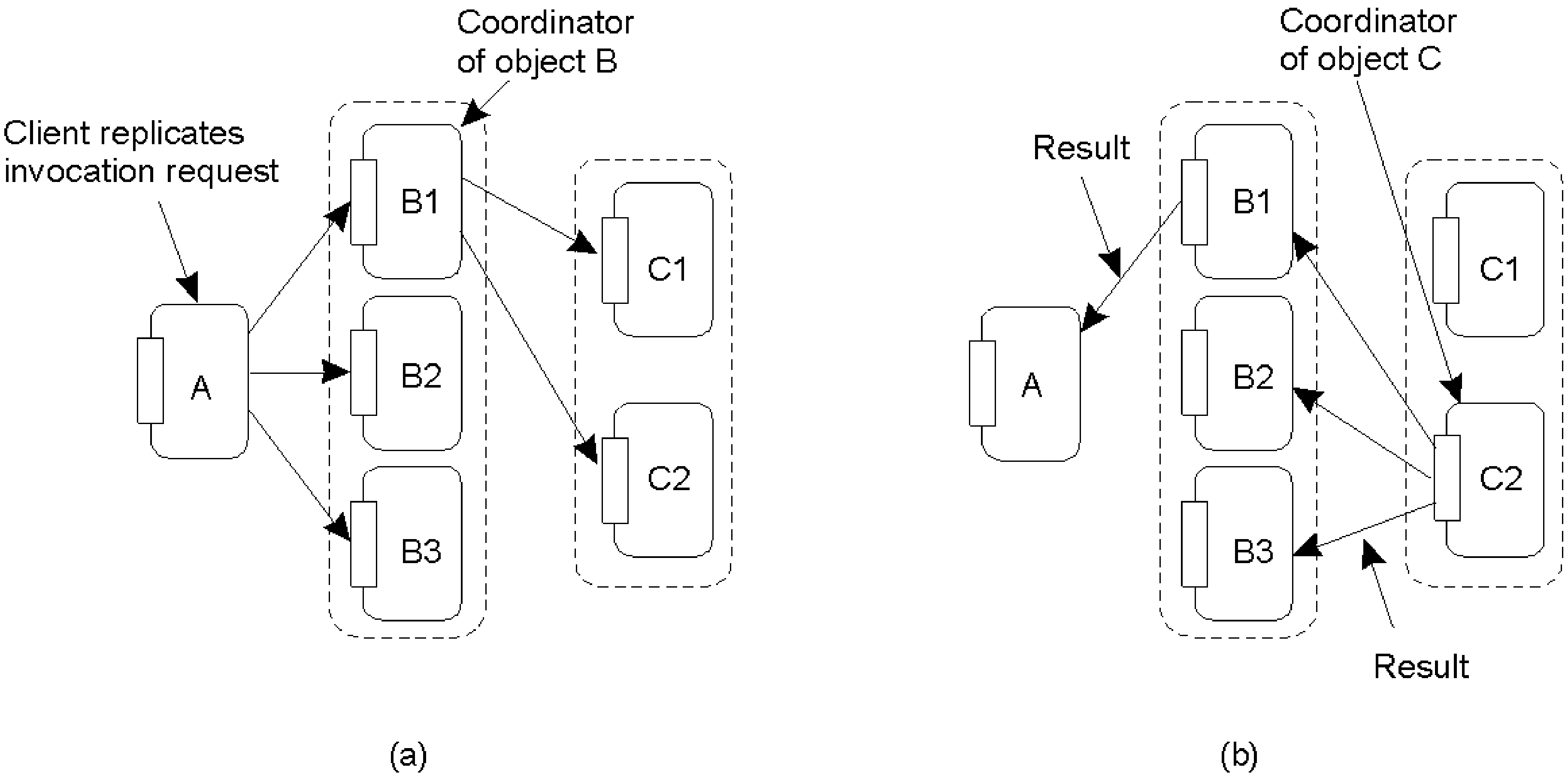
R1. Read request
R2. Response to read

Active Replication (1)


Client replicates invocation request
Object receives the same invocation three times
B1
A
B2
C
B3
All replicas see the same invocation
Replicated object

The problem of replicated invocations.

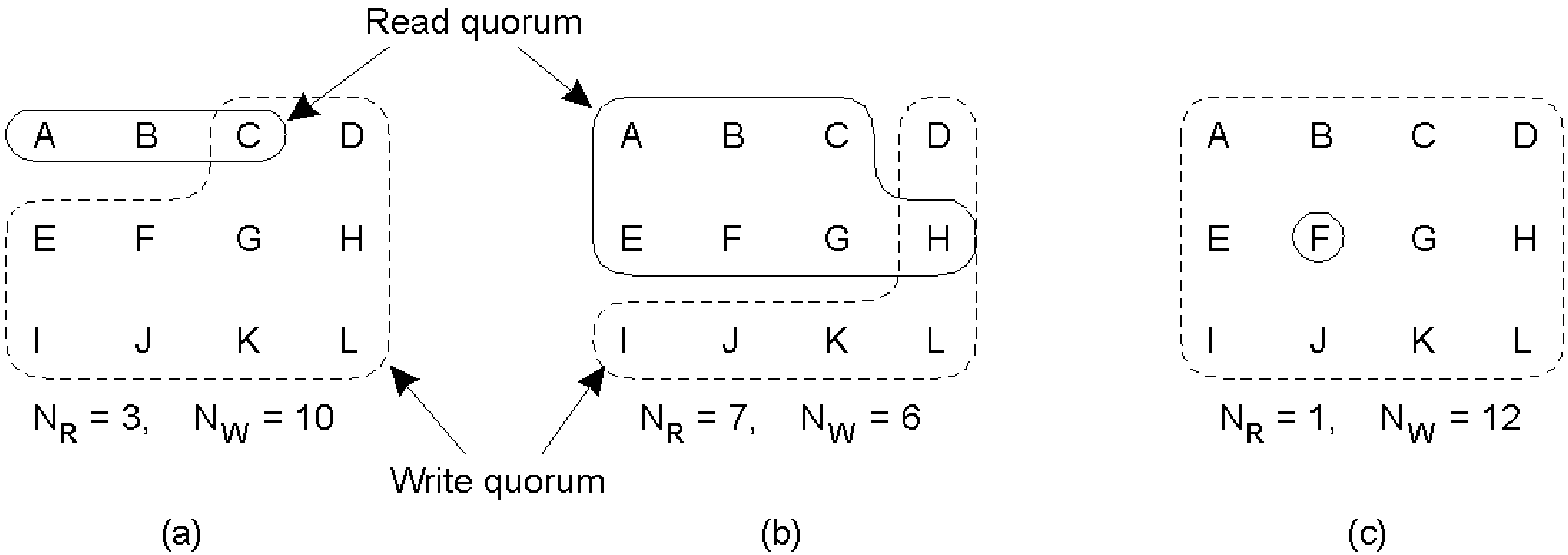
Active Replication (2)



a) Forwarding an invocation request from a replicated object.

b) Returning a reply to a replicated object.

Quorum-Based Protocols



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)

Merci de votre attention