CprE 450/550X
Distributed Systems and Middleware

Consistency & Replication

Yong Guan
3216 Coover
Tel: (515) 294-8378
Email: guan@ee.iastate.edu

April 10, 2003

Readings for Today's Lecture

- References
  - Chapter 6 of "Distributed Systems: Principles and Paradigms"

3

Client-Centric Consistency Models

- Data-centric consistency models
  - **Multiple concurrent processes may simultaneously update the data store**
- Today, we are focusing on a special class of distributed data stores.
  - **There are no or very few simultaneous updates on the data store.**
  - **When such concurrent updates happen, they can be easily resoved.**
  - **Most operations are reading.**
  - **We will introduce a very weak consistency model – eventual consistency.**

4

Client-Centric Consistency Models

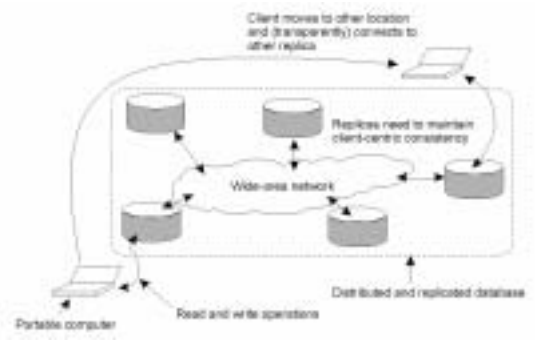
- Examples in which concurrency happens in a restricted manner:
  - **Database systems: read-only**
  - **DNS**
  - **WWW**
  - **They are in common that they can tolerate a relatively high degree of inconsistency.**
- Eventual consistency: If no updates take place for a long time, all replicas will gradually and eventually become consistent.

5

Eventual Consistency: Issue

Will work fine if client always access the same replica.

What about when different replicas are accessed?



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

6

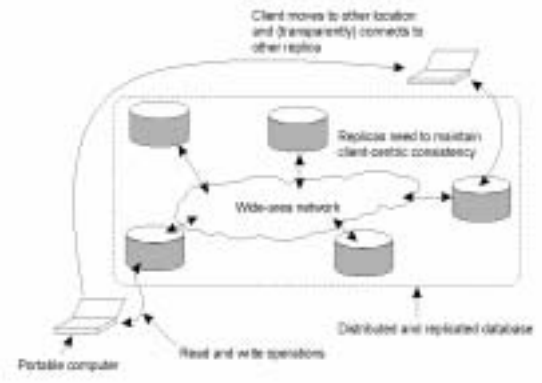
The previous problem can be alleviated

- By Using Client-centric consistency:
  - **Client-centric consistency provides guarantees for a single client concerning the consistency of accesses to a data store by that client**
  - **No guarantees are given concerning concurrent accesses by different clients.**
  - **Originated from the work Bayou.**
  - **In this model, we assume there is only one process that is permitted to update the data store.**

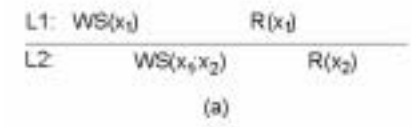
Monotonic Reads

- Condition:

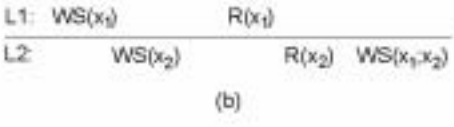
  If a process reads the value of a data item x, any successive read operations on x by that process will always return that same value or a more recent value.



Monotonic Reads



(a)



(b)

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

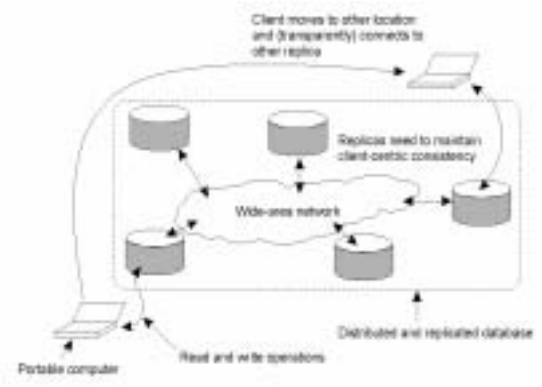
a) A monotonic-read consistent data store

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

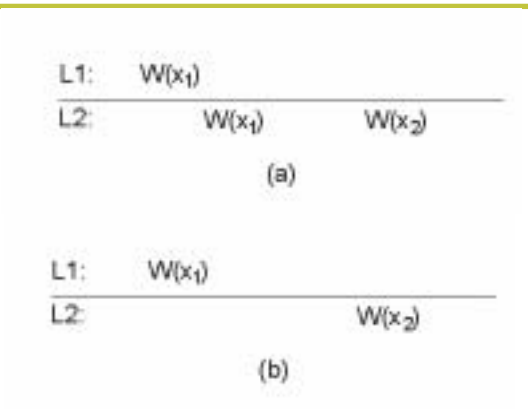
Monotonic Writes

- Condition:

  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



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

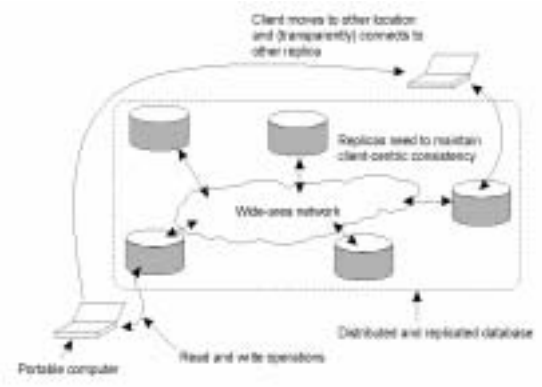
a) A monotonic-write consistent data store.

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

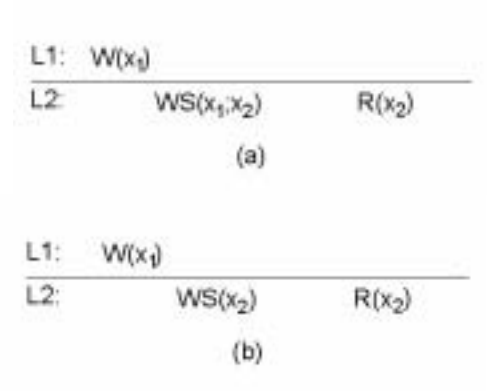
Read your Writes

- Condition:

  The effect of a write operation by a process on a data item x will always be seen by a successive read operation on x by the same process.



Read Your Writes

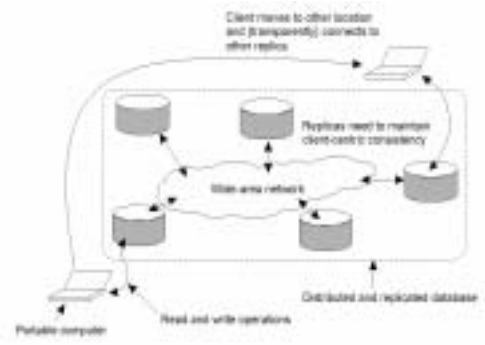


a) A data store that provides read-your-writes consistency.
b) A data store that does not.

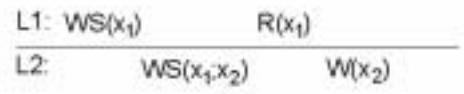
Writes Follow Reads

- Condition:

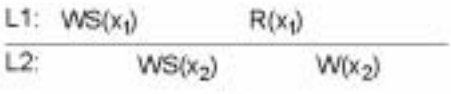
  A write operation by a process on a data item x following a previous read operation on x by the same process, is guaranteed to take place on the same or a more recent value of x that was read.



Writes Follow Reads



(a)



(b)

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

15

Implementations Issues

- Relatively straightforward without considering performance issues
- Each write operation is assigned a globally unique identifier.

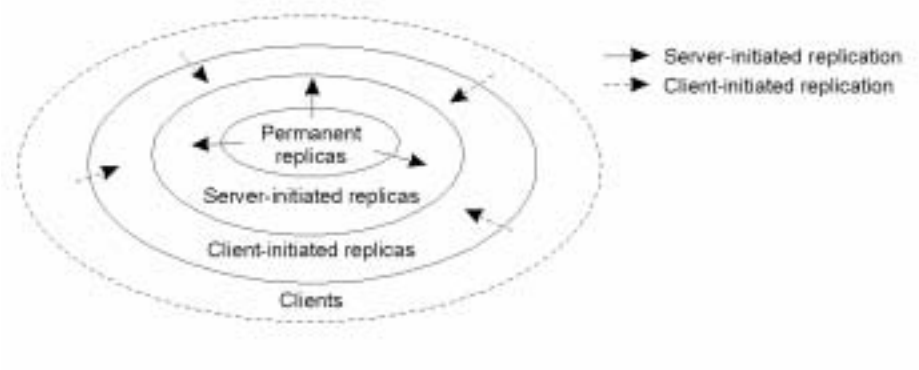
16

Distributed Protocols

- Replica Placement
- Update Propagation
- Epidemic Protocols

17

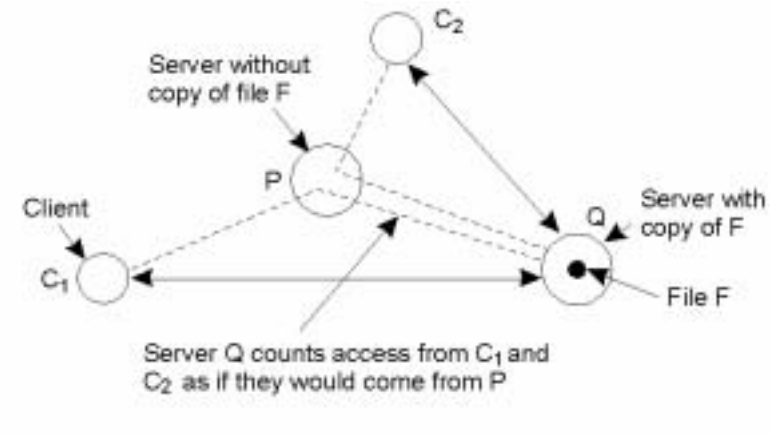
Replica Placement



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

18

Server-Initiated Replicas



Counting access requests from different clients.

19

Client-Initiated Replicas

- Client cache
- Placement of client cache

20

Update Propagaation

- State versus operations
- Pull versus pull protocols
- Unicast versus multicast

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.

Epidemic Protocols

- EP does not solve update conflicts. Propagate updates to all replicas in as few messages as possible.
- Update Propagation Models
  - **Infective if it holds an update that it is willing to spread to other servers**
  - **Susceptible if a server has not been updated yet.**
  - **Removed if an updated server that is not willing to or able to spread its update**
- Anti-entropy model:
  - **Server P chooses Q randomly and then exchanges updates with Q:**
    - » **P pushes its own update to Q**
    - » **P pulls in new updates from Q**
    - » **P and Q send updates to each other.**

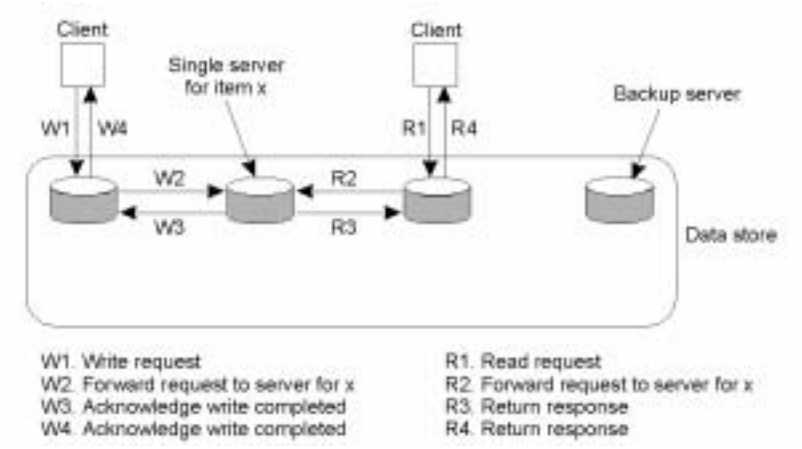
23

Epidemic Protocols

- Variant: Rumor Spreading/gossiping

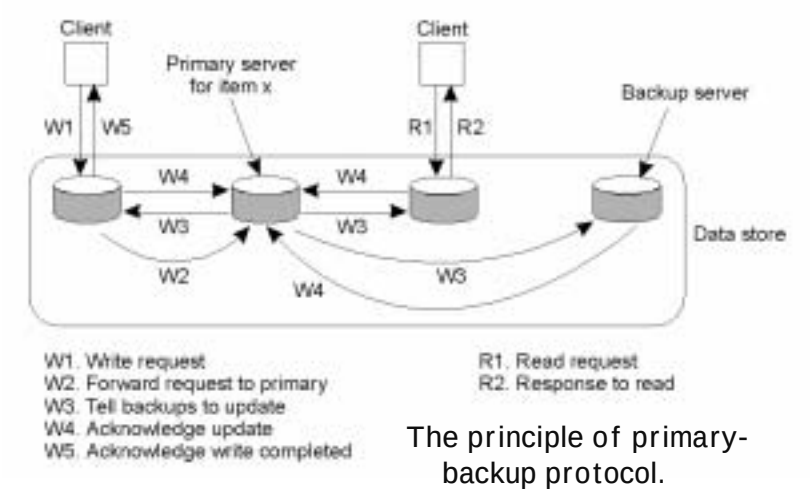
24

Remote-Write Protocols (1)



Primary-based remote-write protocol with a fixed server to which all read and write operations are forwarded.

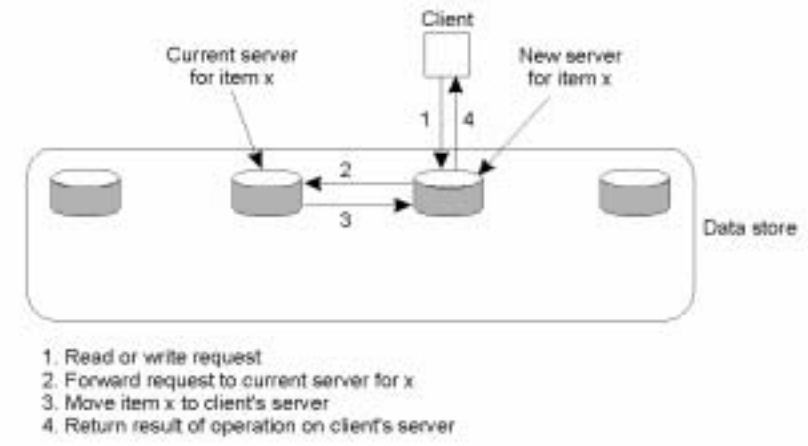
Remote-Write Protocols (2)



The principle of primary-backup protocol.

27

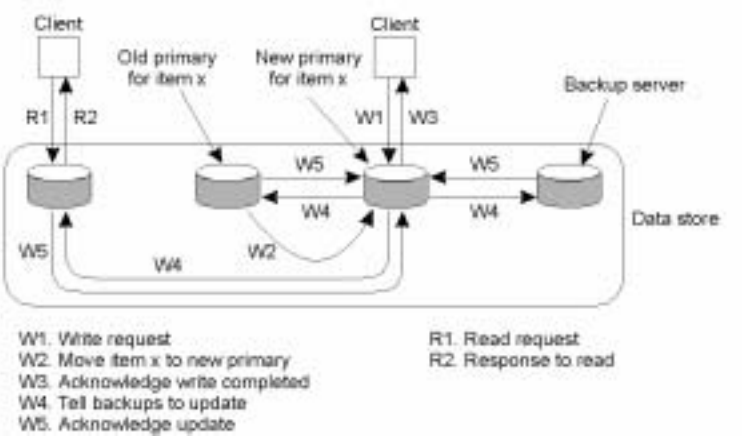
Local-Write Protocols (1)



Primary-based local-write protocol in which a single copy is migrated between processes.

28

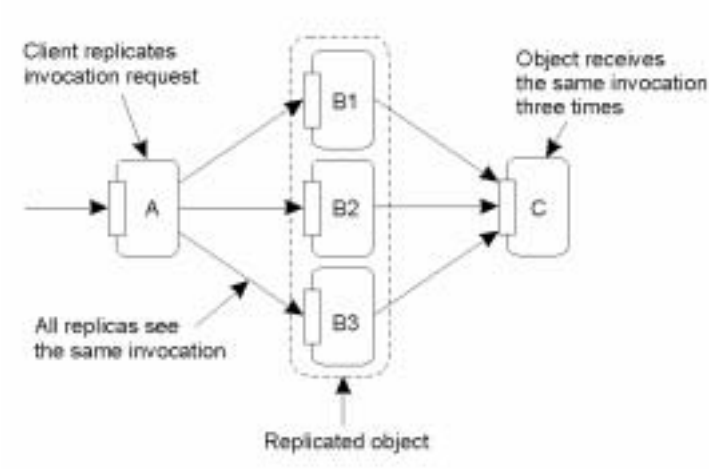
Local-Write Protocols (2)



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

29

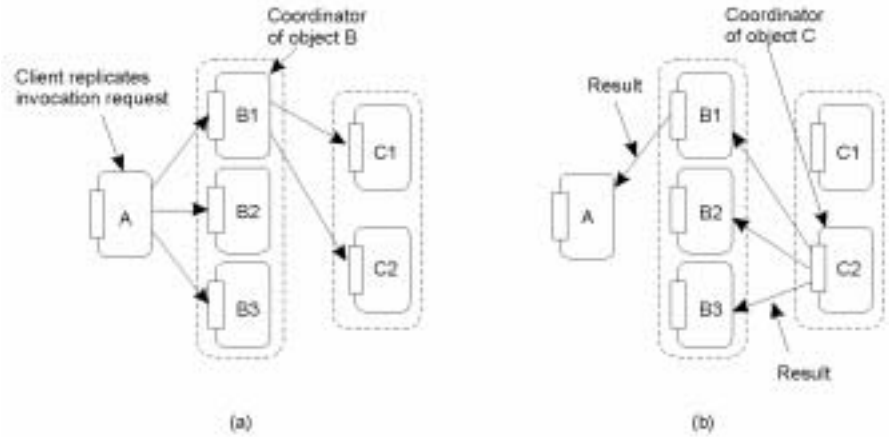
Active Replication (1)



The problem of replicated invocations.

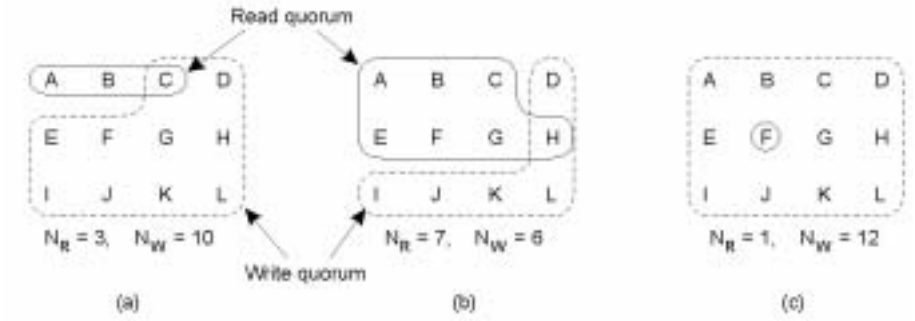
30

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)

Orca

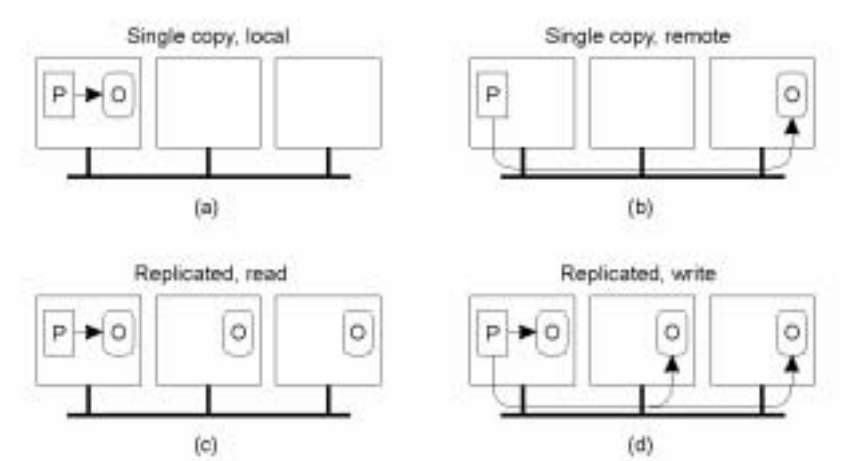
```
OBJECT IMPLEMENTATION stack;
  top: integer;                          # variable indicating the top
  stack: ARRAY[integer 0..N-1] OF integer  # storage for the stack

  OPERATION push (item: integer)         # function returning nothing
  BEGIN
    GUARD top < N DO
        stack [top] := item;             # push item onto the stack
        top := top + 1;                  # increment the stack pointer
    OD;
  END;

  OPERATION  pop():integer;              # function returning an integer
  BEGIN
    GUARD top > 0 DO                     # suspend if the stack is empty
         top := top – 1;                 # decrement the stack pointer
        RETURN stack [top];              # return the top item
    OD;
  END;
BEGIN
  top := 0;                              # initialization
END;
```

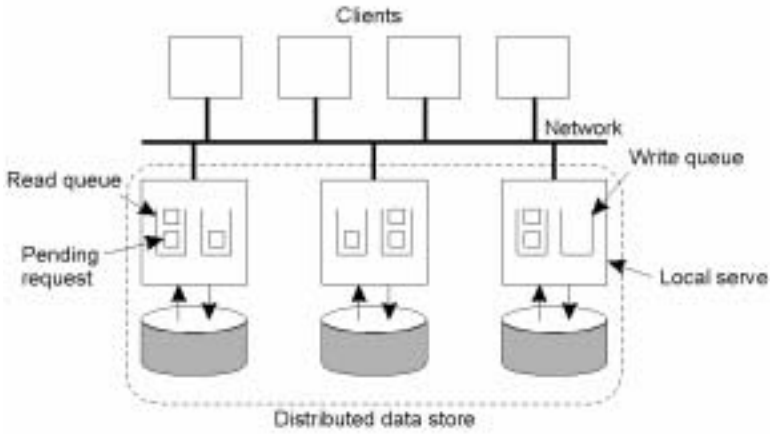
A simplified stack object in Orca, with internal data and two operations.

Management of Shared Objects in Orca



Four cases of a process *P* performing an operation on an object *O* in Orca.

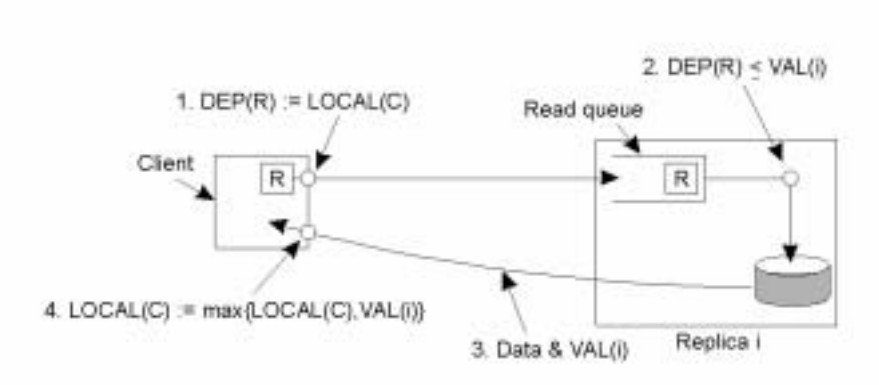
Casually-Consistent Lazy Replication



The general organization of a distributed data store. Clients are assumed to also handle consistency-related communication.

35

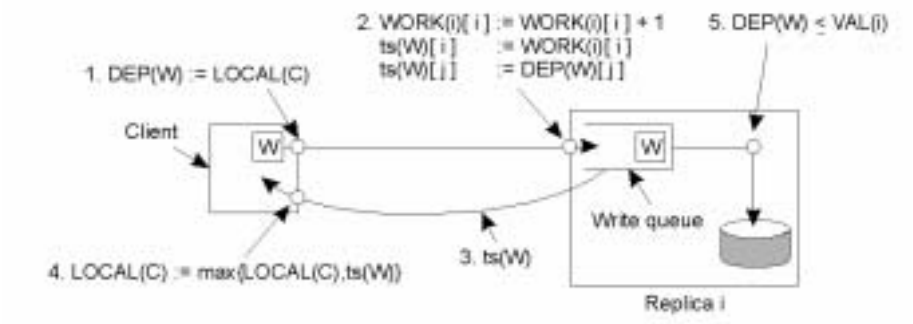
Processing Read Operations



Performing a read operation at a local copy.

36

Processing Write Operations



Performing a write operation at a local copy.


37