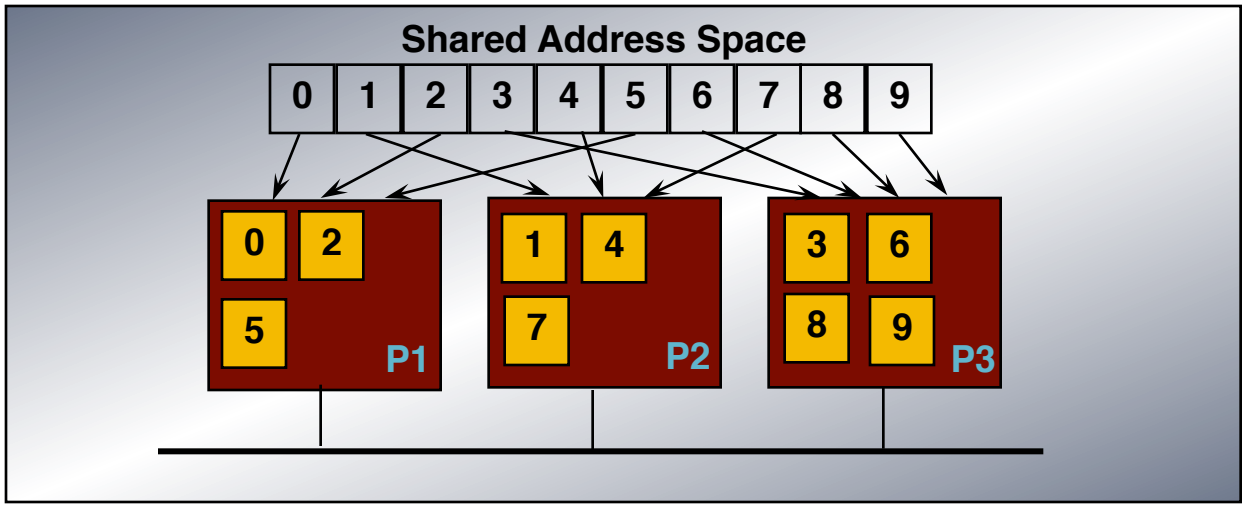
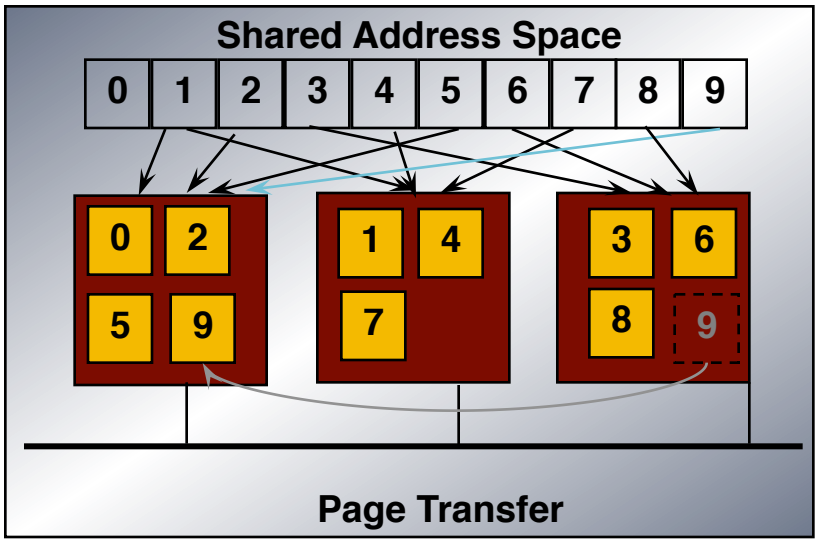
CS425/CSE424/ECE428 – Distributed Systems

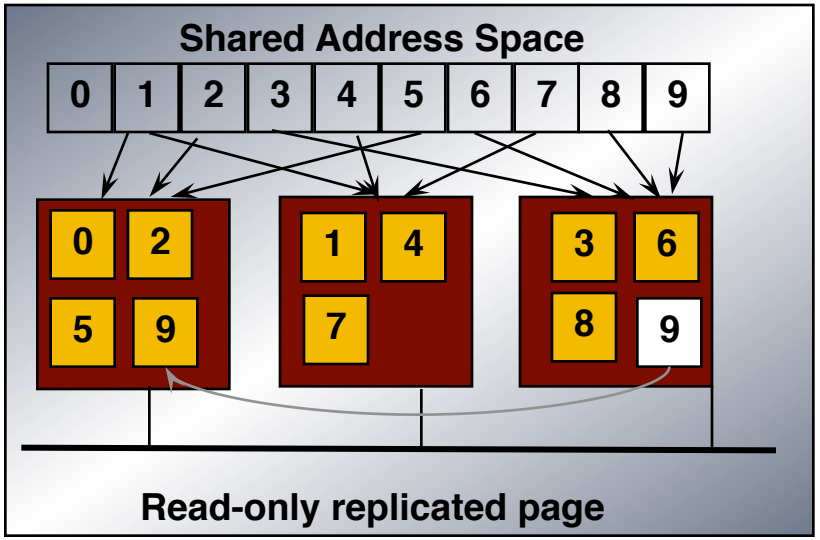
Distributed Shared Memory

Some material derived from slides by I. Gupta, M. Harar
J. Hou, S. Mitra, K. Nahrstedt, N. Vaidya

The Basic Model of DSM





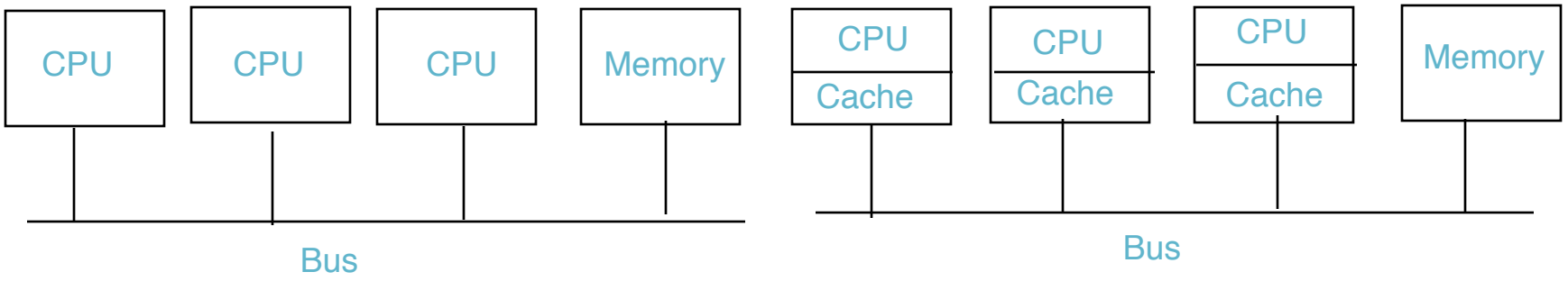


Shared Memory vs. Message Passing

- In a multiprocessor, two or more processors share a common main memory. Any process on a processor can read/write any word in the shared memory. All communication through a bus.
  - E.g., Cray supercomputer
  - Called Shared Memory
- In a multicomputer, each processor has its own private memory. All communication using a network.
  - E.g., CSIL PC cluster
  - Easier to build: One can take a large number of single-board computers, each containing a processor, memory, and a network interface, and connect them together. (called COTS="Components off the shelf")
  - Uses Message passing
- Message passing can be implemented over shared memory.
- Shared memory can be implemented over message passing.
- Let's look at shared memory by itself.

Bus-Based Multiprocessors with Shared Memory

- When any of the CPUs wants to read a word from the memory, it puts the address of the requested word on the address line, and asserts the bus control (read) line.
- To prevent two CPUs from accessing the memory at the same time, a bus arbitration mechanism is used, i.e., if the control line is already asserted, wait.
- To improve performance, each CPU can be equipped with a snooping cache.
- Snooping used in both (a) write-through and (b) write-once models

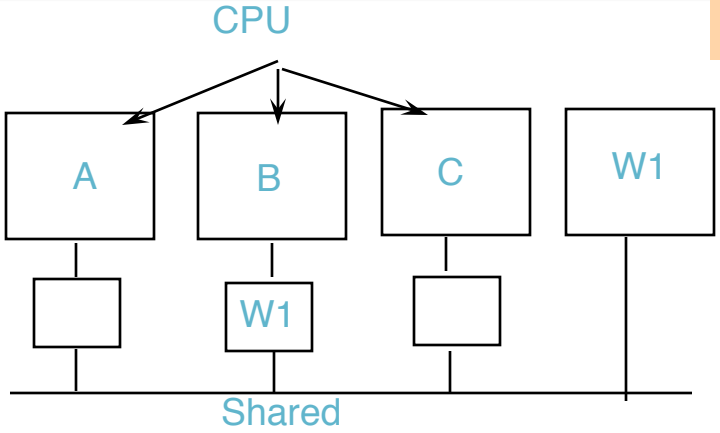


Cache Consistency – Write Through

Event	Action taken by a cache in response to its own operation	Action taken by other caches in response (to a remote operation)
Read hit	Fetch data from local cache	(no action)
Read miss	Fetch data from memory and store in cache	(no action)
Write miss	Update data in memory and store in cache	Invalidate cache entry
Write hit	Update memory and cache	Invalidate cache entry

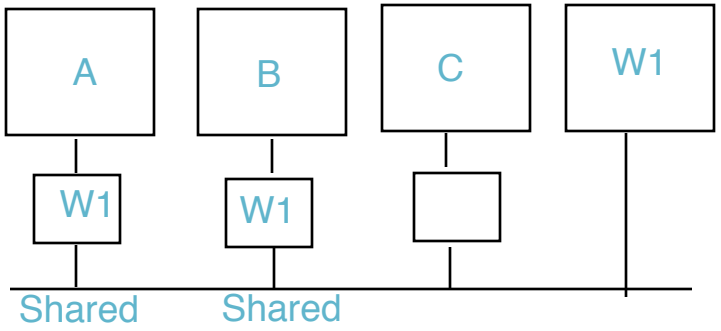
All the other caches see the write (because they are snooping on the bus) and check to see if they are also holding the word being modified. If so, they invalidate the cache entries.

Cache Consistency – Write Once

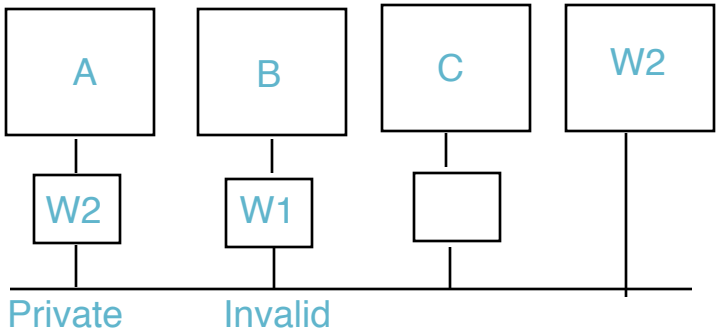


Initially both the memory and B have an updated entry of word W.

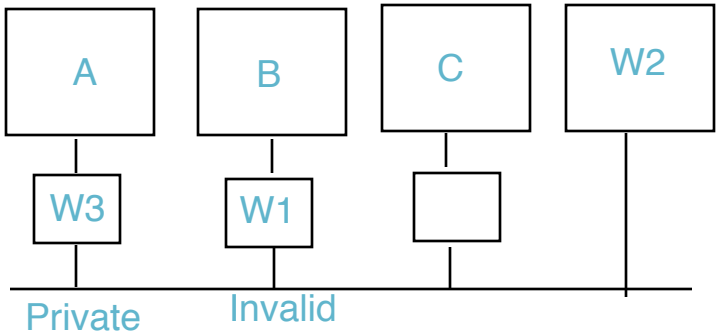
•For write, at most one CPU has valid access



A reads word W and gets W1. B does not respond but the memory does.

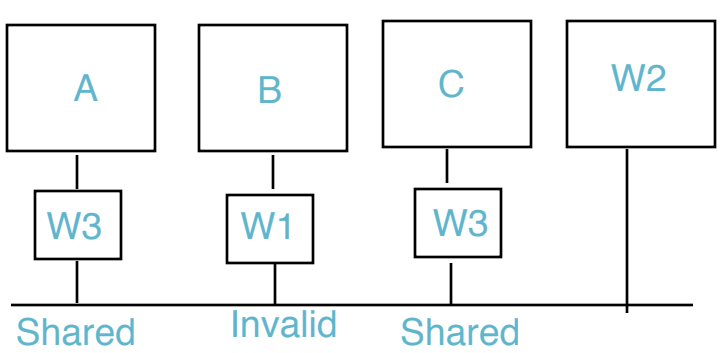


A writes a value W2. B snoops on the bus, and invalidates its entry. A's copy is marked as private.

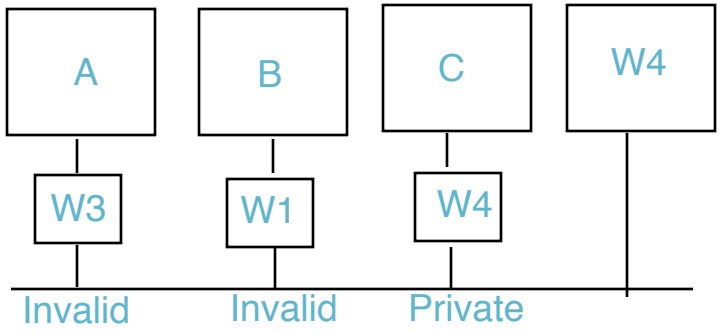


A writes W again. This and subsequent writes by A are done locally, without any bus traffic.

Cache Consistency – Write Once



C reads W. A sees the request by snooping on the bus, asserts a signal that inhibits memory from responding, provides the values. Also changes label to Shared.



C writes W. A invalidates it own entry. C now has the only valid copy.

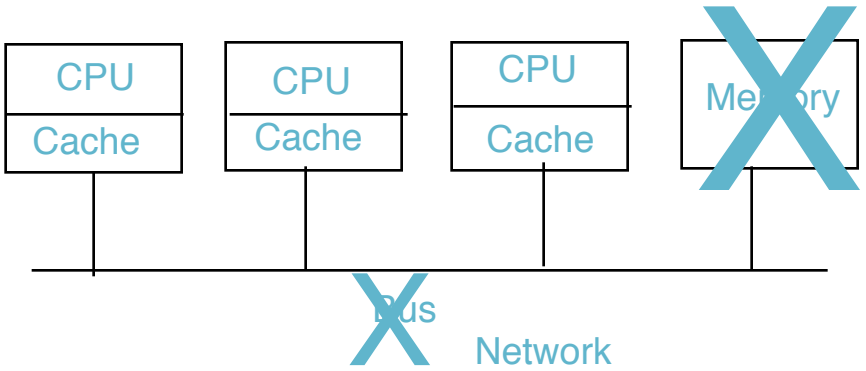
The cache consistency protocol is built upon the notion of snooping and built into the memory management unit (MMU).
All above mechanisms are implemented in hardware for efficiency.

The above shared memory can be implemented using message passing instead of the bus.

Distributed Shared Memory (DSM)

- Basic idea: Create the illusion of global shared address space
- Approach:
  - Divide address space into chunks (pages)
  - Distribute page storage across computers
  - Use the page fault mechanism to migrate chunk to local memory
- *Similar to virtual memory, but missing pages filled from* other computers *instead of disk*

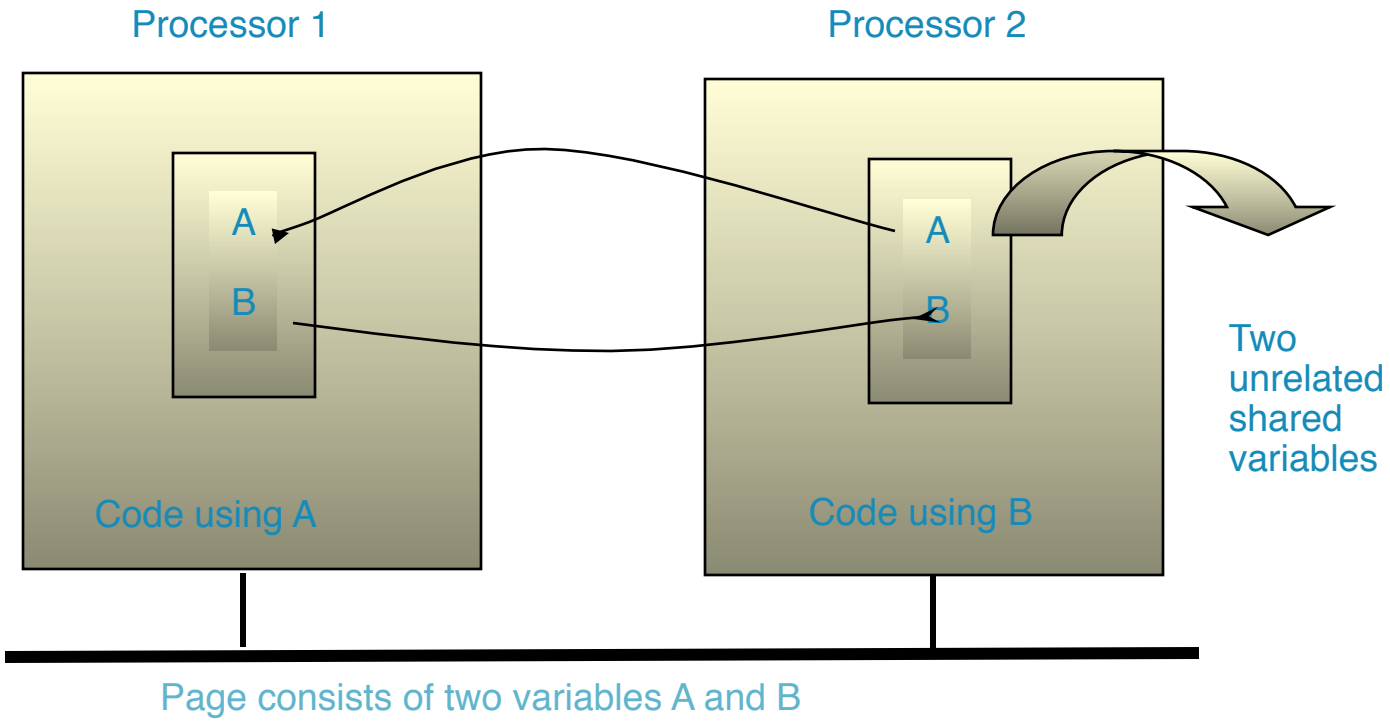
Distributed Shared Memory



Granularity of Chunks

- When a processor references a word that is absent, it causes a page fault.
- On a page fault,
  - the missing page is just brought in from a remote processor.
  - A region of 2, 4, or 8 pages including the missing page may also be brought in.
    - Locality of reference: if a processor has referenced one word on a page, it is likely to reference other neighboring words in the near future.
- Region size
  - Small => too many page transfers
  - Large => False sharing
  - Above tradeoff also applies to page size

False Sharing



Occurs because: Page size > locality of reference
Unrelated variables in a region cause large number of pages transfers
Large page sizes => more pairs of unrelated variables

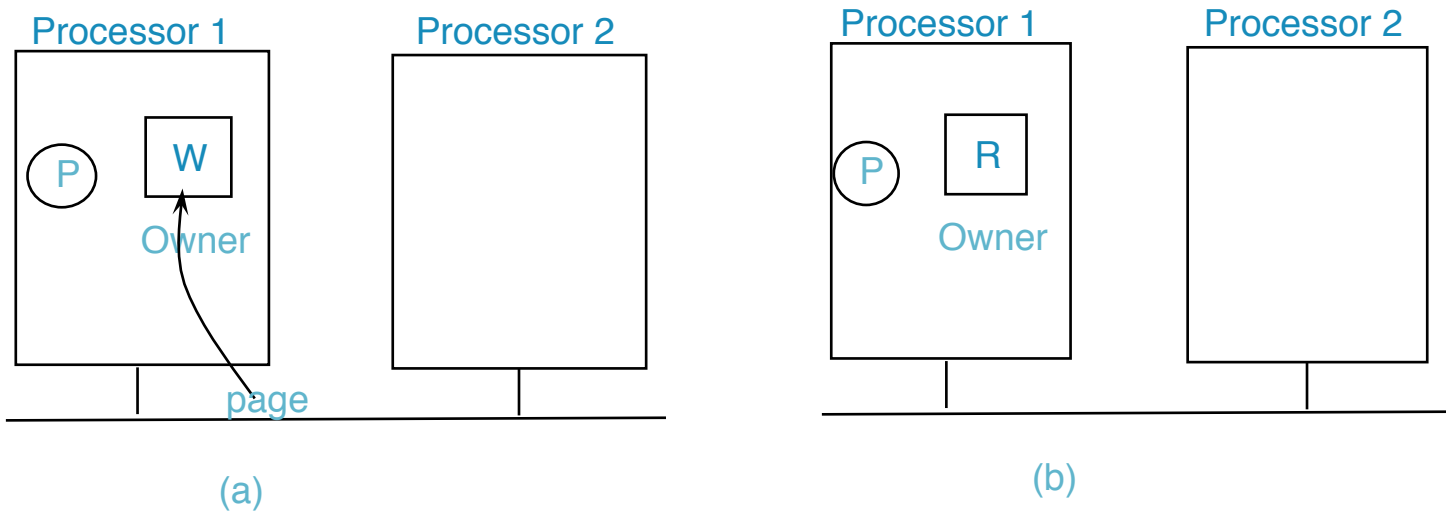
Achieving Sequential Consistency

- Achieving consistency is not an issue if
  - Pages are not replicated, or…
  - Only read-only pages are replicated
- But don't want to compromise performance.
- Two approaches are taken in DSM
  - Update: the write is allowed to take place locally, but the address of the modified word and its new value are broadcast to all the other processors. Each processor holding the word copies the new value, i.e., updates its local value.
  - Invalidate: The address of the modified word is broadcast, but the new value is not. Other processors invalidate their copies. (Similar to example in first few slides for multiprocessor)
  - Page-based DSM systems typically use an invalidate protocol instead of an update protocol. ? [Why?]

Invalidation Protocol to Achieve Consistency

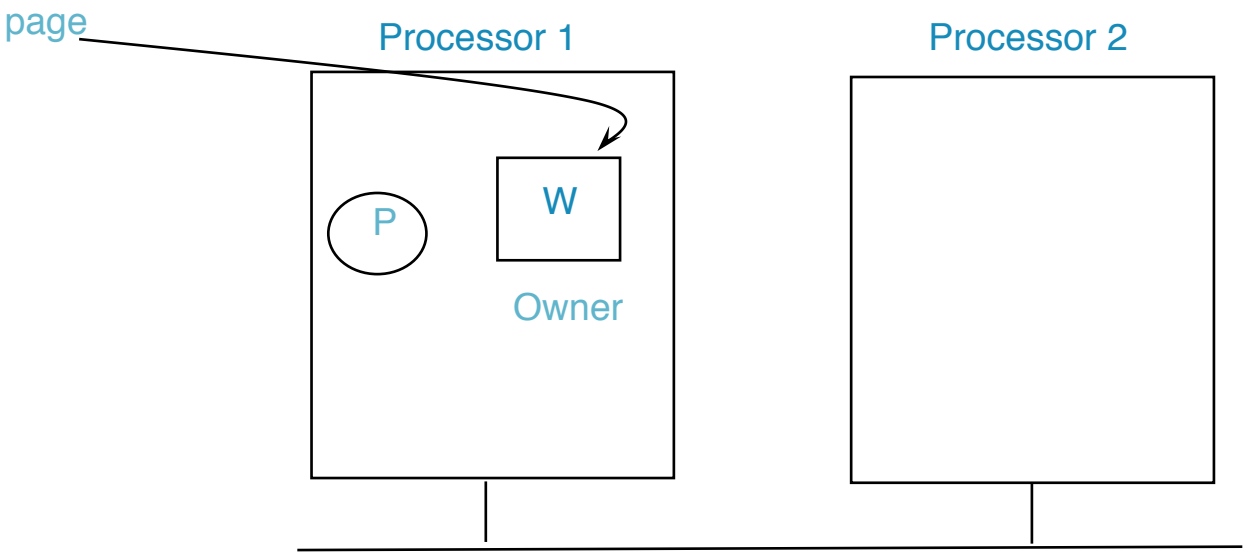
- Each page is either in R or W state.
  - When a page is in W state, only one copy exists, located at one processor (called current "owner") in read-write mode.
  - When a page is in R state, the current/latest owner has a copy (mapped read-only), but other processors may have copies.

Suppose Processor 1 is attempting a read: Different scenarios



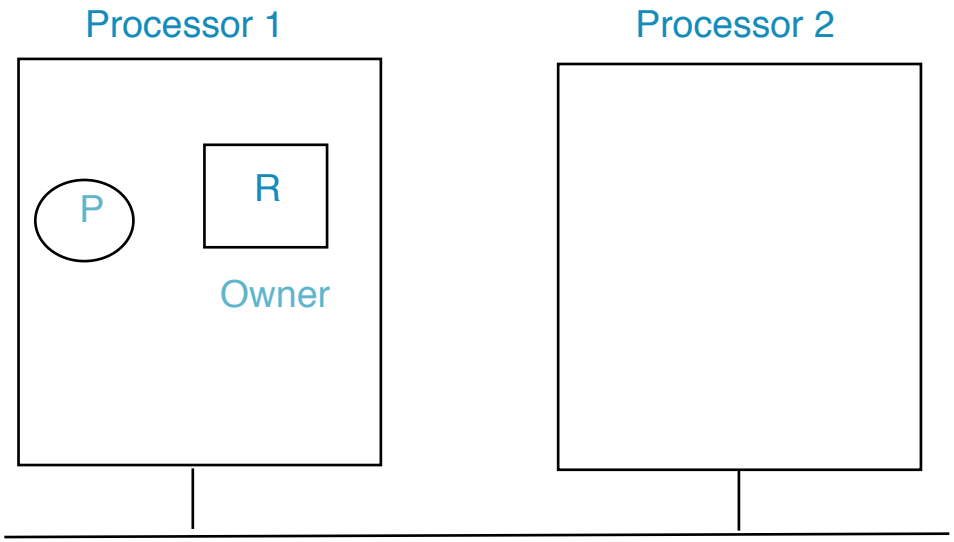
Invalidation Protocol: Read

Suppose Processor 1 is attempting a read: Different scenarios



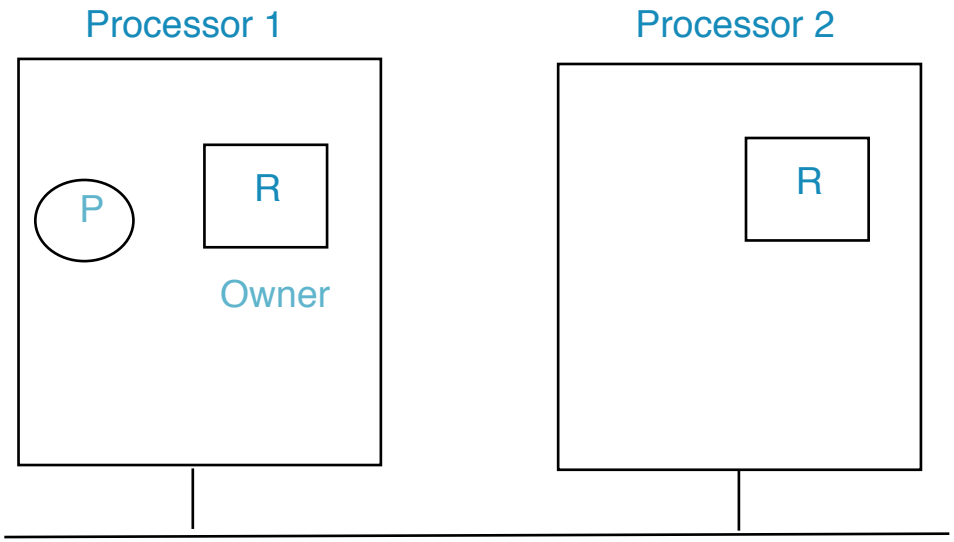
Exclusive write access, no trap

Invalidation Protocol: Read



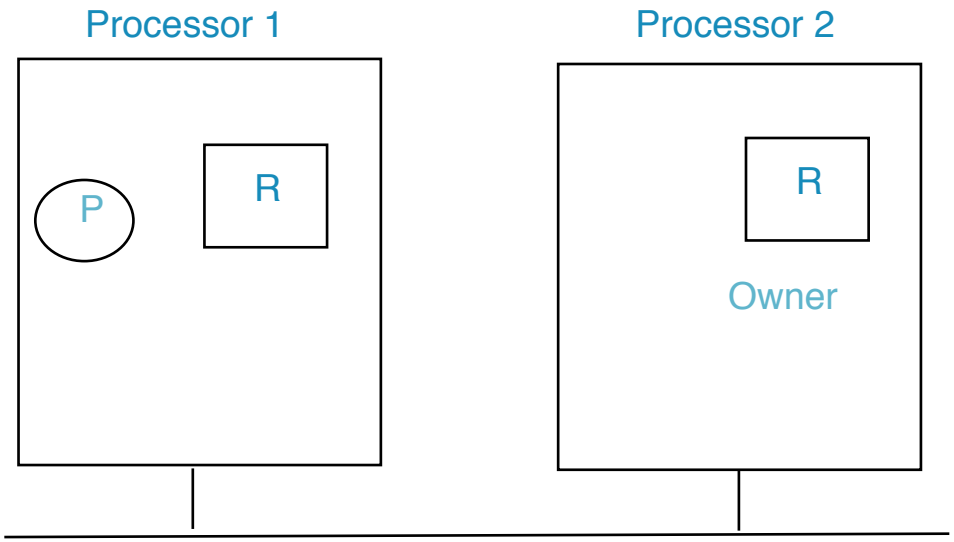
Exclusive read access, no trap

Invalidation Protocol: Read



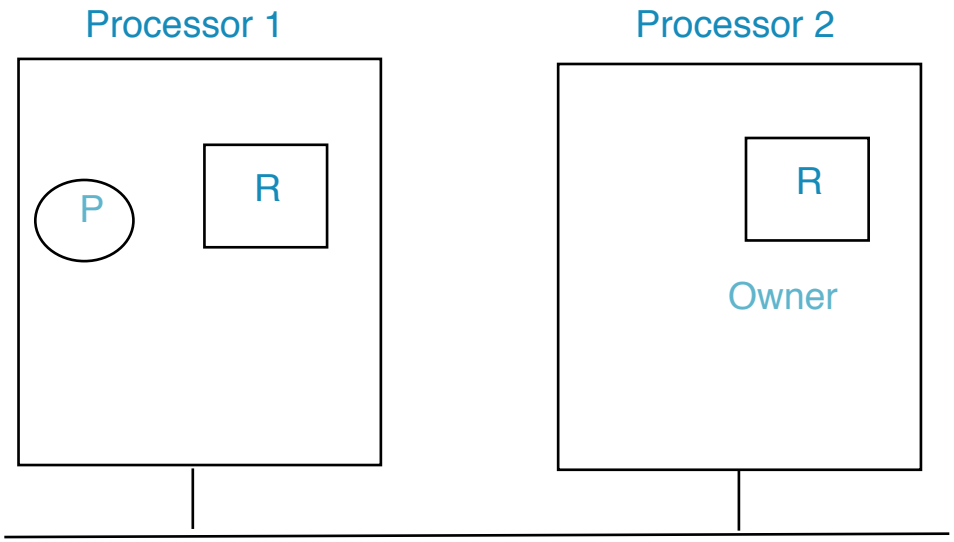
Shared read access, no trap

Invalidation Protocol: Read



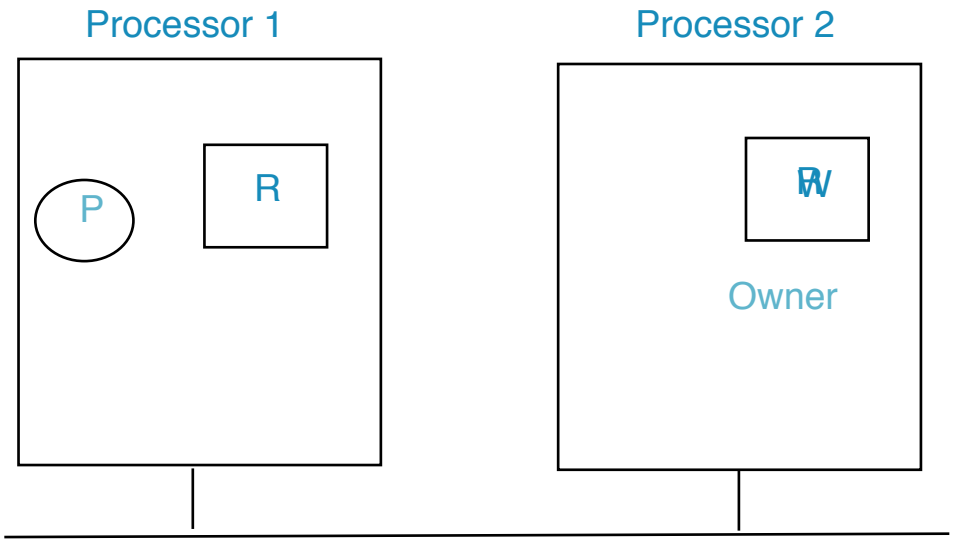
Shared read access, no trap

Invalidation Protocol: Read



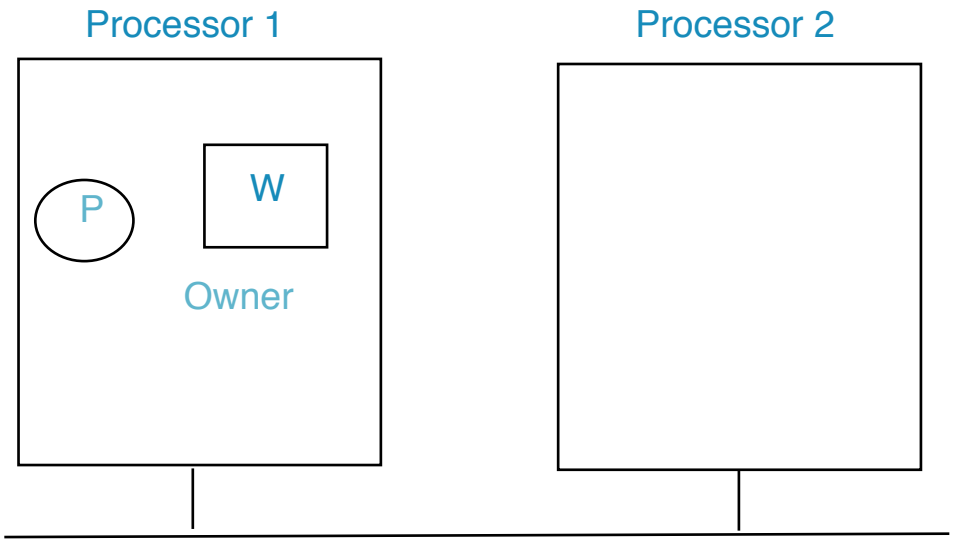
Read miss
1. Ask for copy
2. Mark as R
3. Satisfy read

Invalidation Protocol: Read



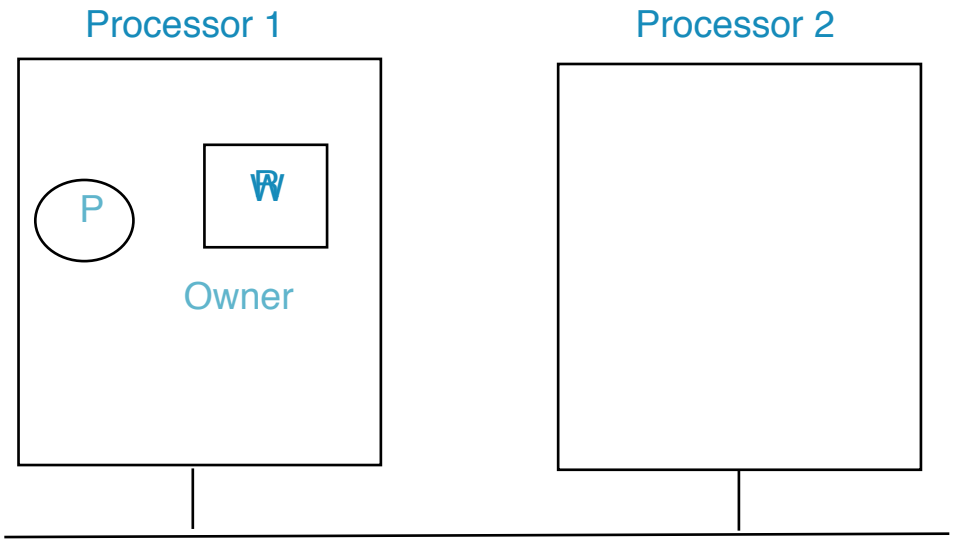
Read miss
1. Ask to downgrade from W to R
2. Ask for copy
3. Mark as R
4. Satisfy read

Invalidation Protocol: Write



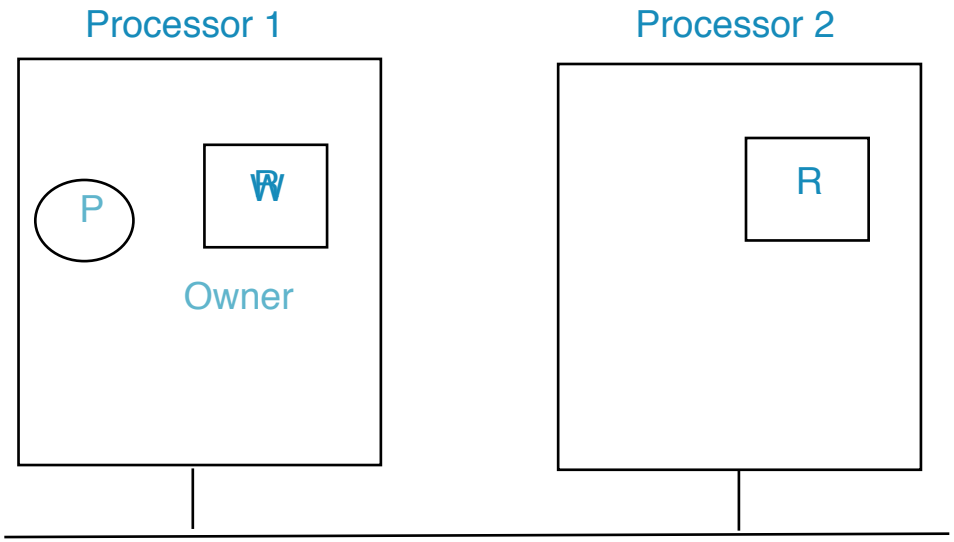
Write hit: No trap

Invalidation Protocol: Write



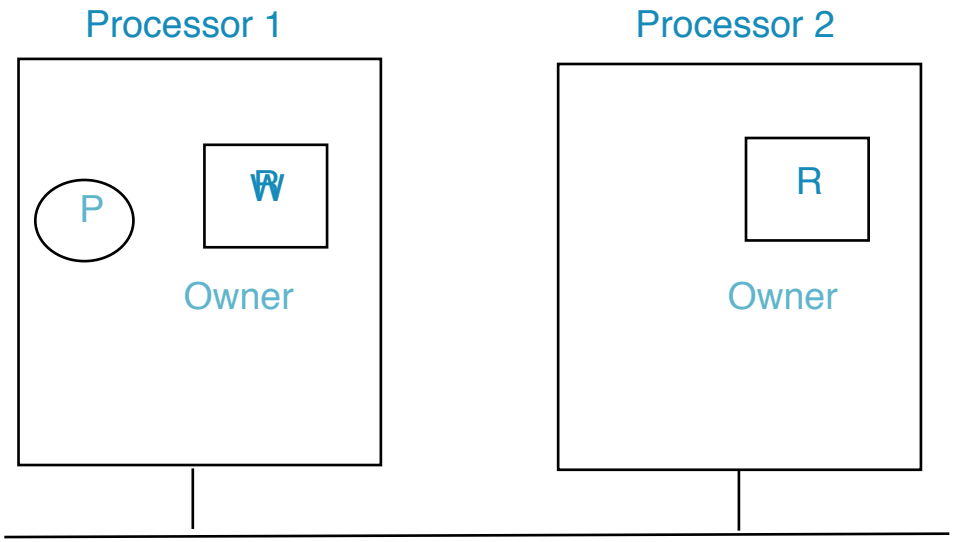
Upgrade to write

Invalidation Protocol: Write



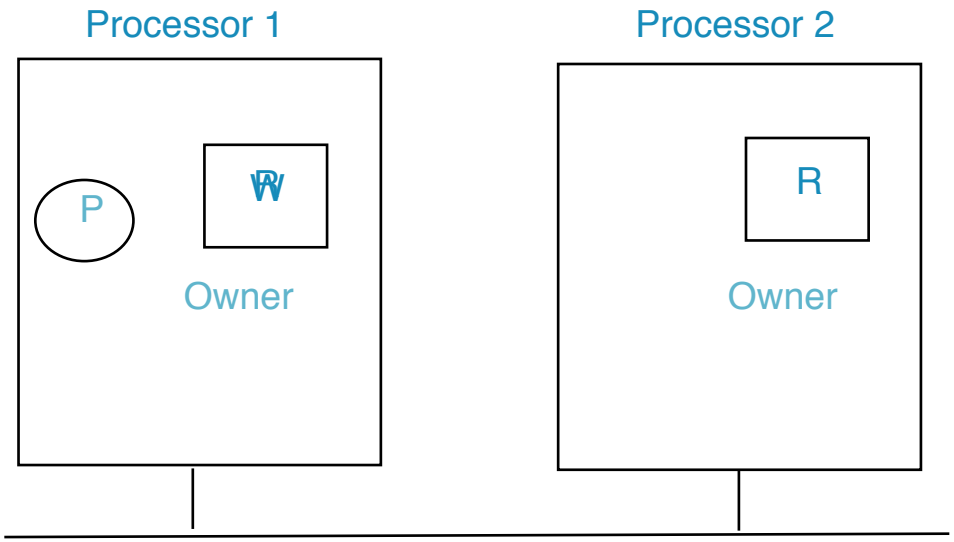
1. Invalidate other copies
2. Upgrade to write

Invalidation Protocol: Write



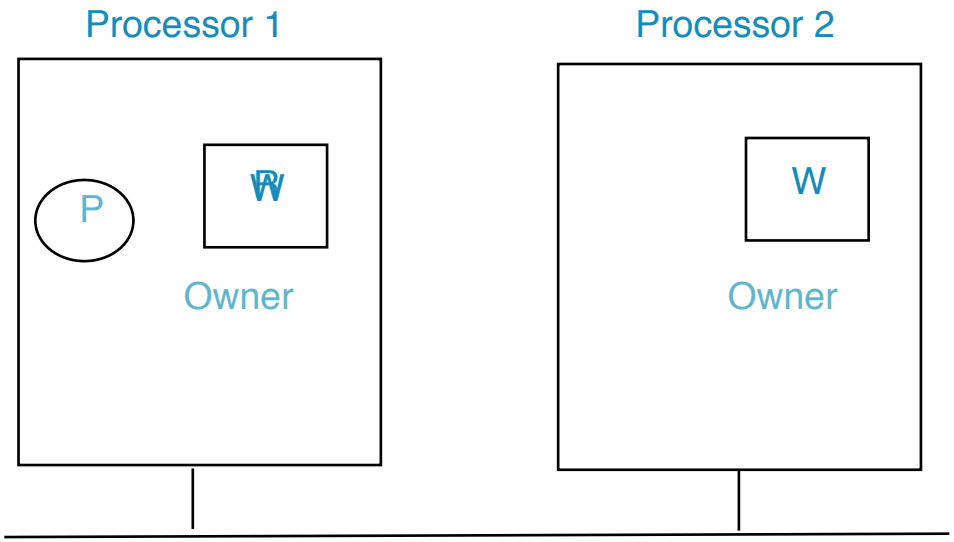
1. Ask for ownership
2. Invalidate other copies
3. Upgrade to write

Invalidation Protocol: Write



1. Ask for a copy
2. Ask for ownership
3. Invalidate other copies
4. Upgrade to write

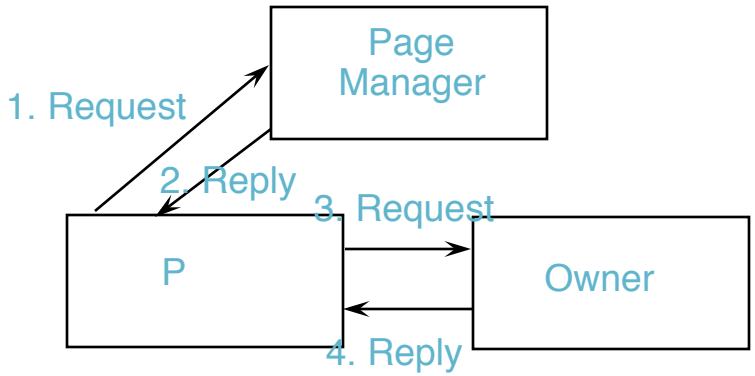
Invalidation Protocol: Write

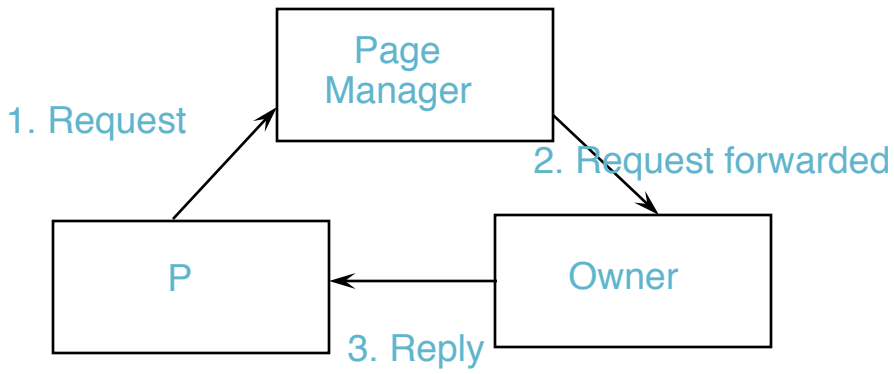


1. Ask for a copy
2. P2 downgrades to R
3. Ask for ownership
4. Invalidate other copies
5. Upgrade to write

Finding the Owner

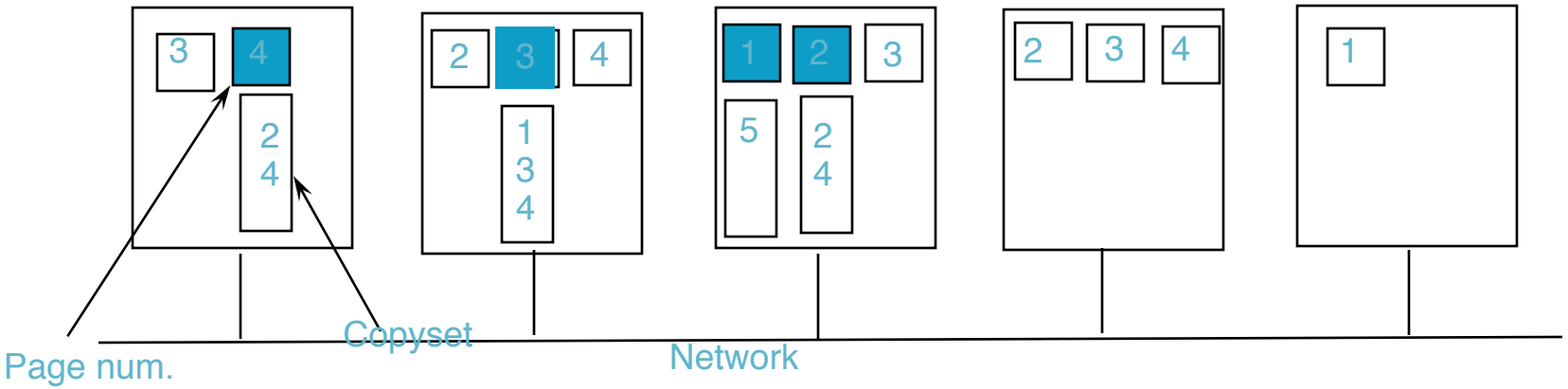
- Owner is the processor with latest updated copy. How do you locate it?
- 1. Do a broadcast, asking for the owner to respond.
  - Broadcast interrupts each processor, forcing it to inspect the request packet.
  - An optimization is to include in the message whether the sender wants to read/write and whether it needs a copy.
- 2. Designate a page manager to keep track of who owns which page.
  - A page manager uses incoming requests not only to provide replies but also to keep track of changes in ownership.
  - Potential performance bottleneck → multiple page managers
    - The lower-order bits of a page number are used as an index into a table of page managers.





How does the Owner Find the Copies to Invalidate

- Broadcast a msg giving the page num. and asking processors holding the page to invalidate it.
  - Works only if broadcast messages are reliable and can never be lost. Also expensive.
- The owner (or page manager) for a page maintains a copyset list giving processors currently holding the page.
  - When a page must be invalidated, the owner (or page manager) sends a message to each processor holding the page and waits for an acknowledgement.



Strict and Sequential Consistency

- Different types of consistency: a tradeoff between accuracy and performance.
- Strict Consistency (one-copy semantics)
  - Any read to a memory location x returns the value stored by the most recent write operation to x.
  - When memory is strictly consistent, all writes are instantaneously visible to all processes and a total order is achieved.
  - Similar to "Linearizability"
- Sequential Consistency
  - For any execution, a sequential order can be found for all ops in the execution so that
  - The sequential order is consistent with individual program orders (FIFO at each processor)
  - Any read to a memory location x should have returned (in the actual execution) the value stored by the most recent write operation to x in this sequential order.

Sequential Consistency

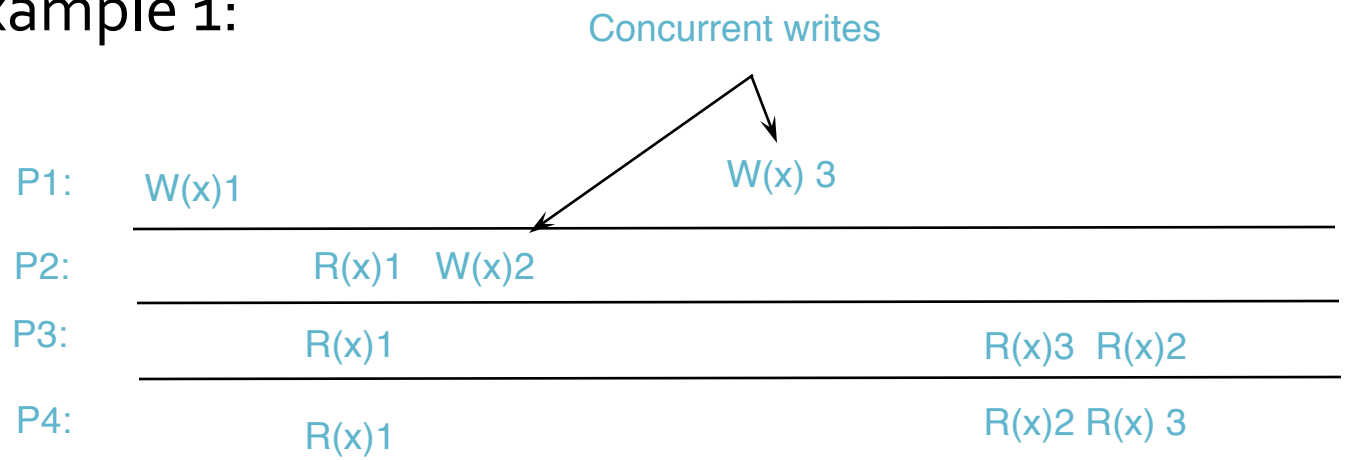
- In this model, writes must occur in the same order on all copies, reads however can be interleaved on each system, as convenient. Stale reads can occur.
- Can be realized in a system with causal-totally ordered reliable broadcast mechanism.

How to Determine the Sequential Order?

- Example: Given H1= W(x)1 and H2= R(x)0 R(x)1, how do we come up with a sequential order (single string S of ops) that satisfies seq. cons.
  - Program order must be maintained
  - Memory coherence must be respected: a read to some location, x must always return the value most recently written to x.
- Answer: S= R(x)0 W(x)1 R(x)1

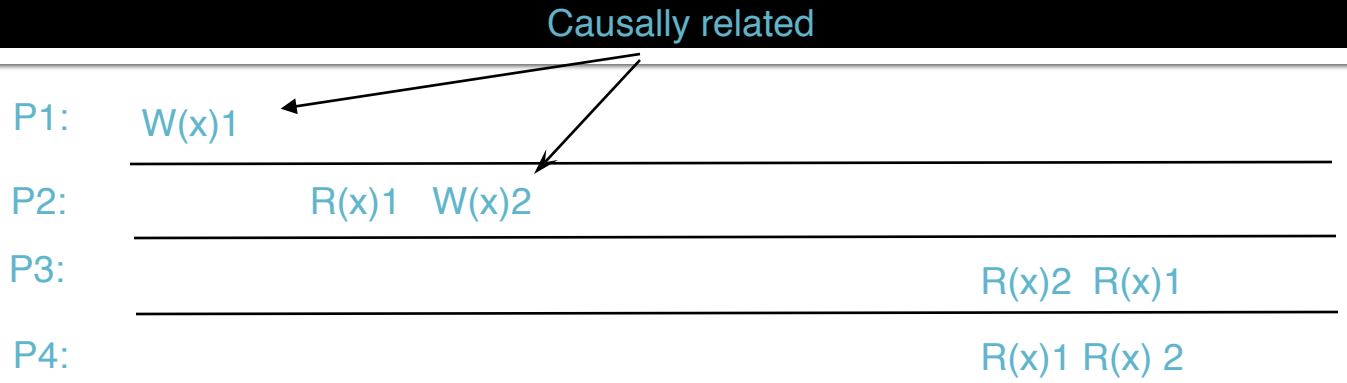
Causal Consistency

- 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.
- Example 1:

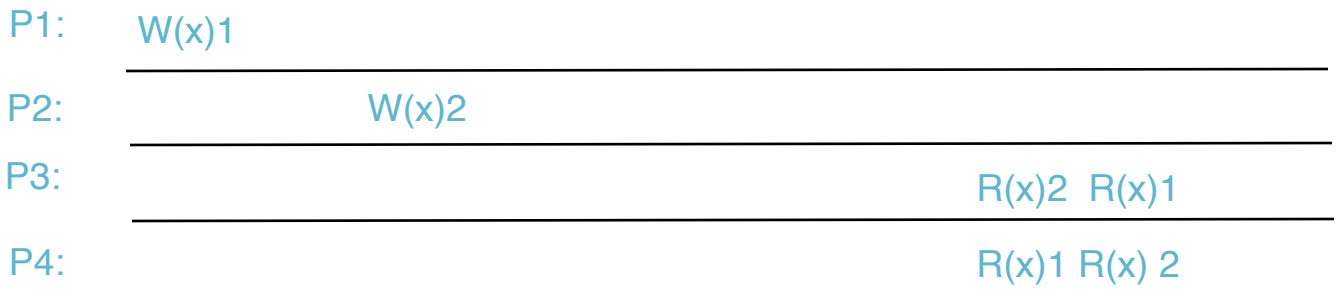


This sequence obeys causal consistency

Causal Consistency



This sequence does not obey causal consistency



This sequence obeys causal consistency

DSM vs. Message Passing

- Advantages
  - Simple programming model
  - No marshalling
- Disadvantages
  - Higher overhead due to false sharing
  - Does not handle failures well
  - Difficult with heterogeneous systems

Summary

- DSM: usually implemented in multicomputer where there is no global memory
- Invalidate versus Update protocols
- Consistency models – tradeoff between accuracy and performance
  - strict, sequential, causal, etc.
- Some of the material is from Tanenbaum (on reserve at library), but slides ought to be enough.
  - Reading from Coulouris textbook: Chap 18 (4^{th} ed; relevant parts – topics covered in the slides), 6.5.1 (5^{th} ed)