EECS 645
Computer Architecture

Lecture 10 Multiprocessor and Memory Coherence

Chenyun Pan

Department of Electrical Engineering &

Computer Science

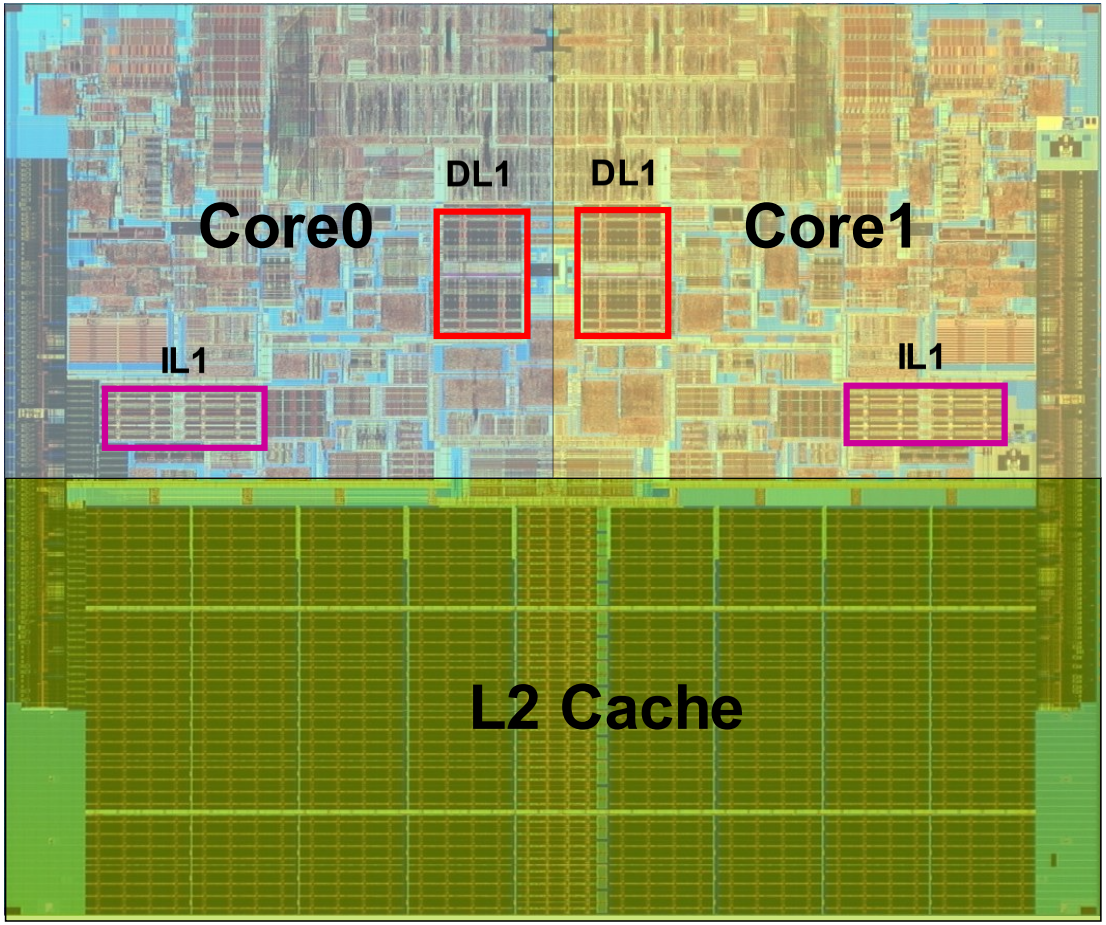
University of Kansas

Slide Courtesy of Dr. Hsien-Hsin Sean Lee

KU THE UNIVERSITY OF KANSAS

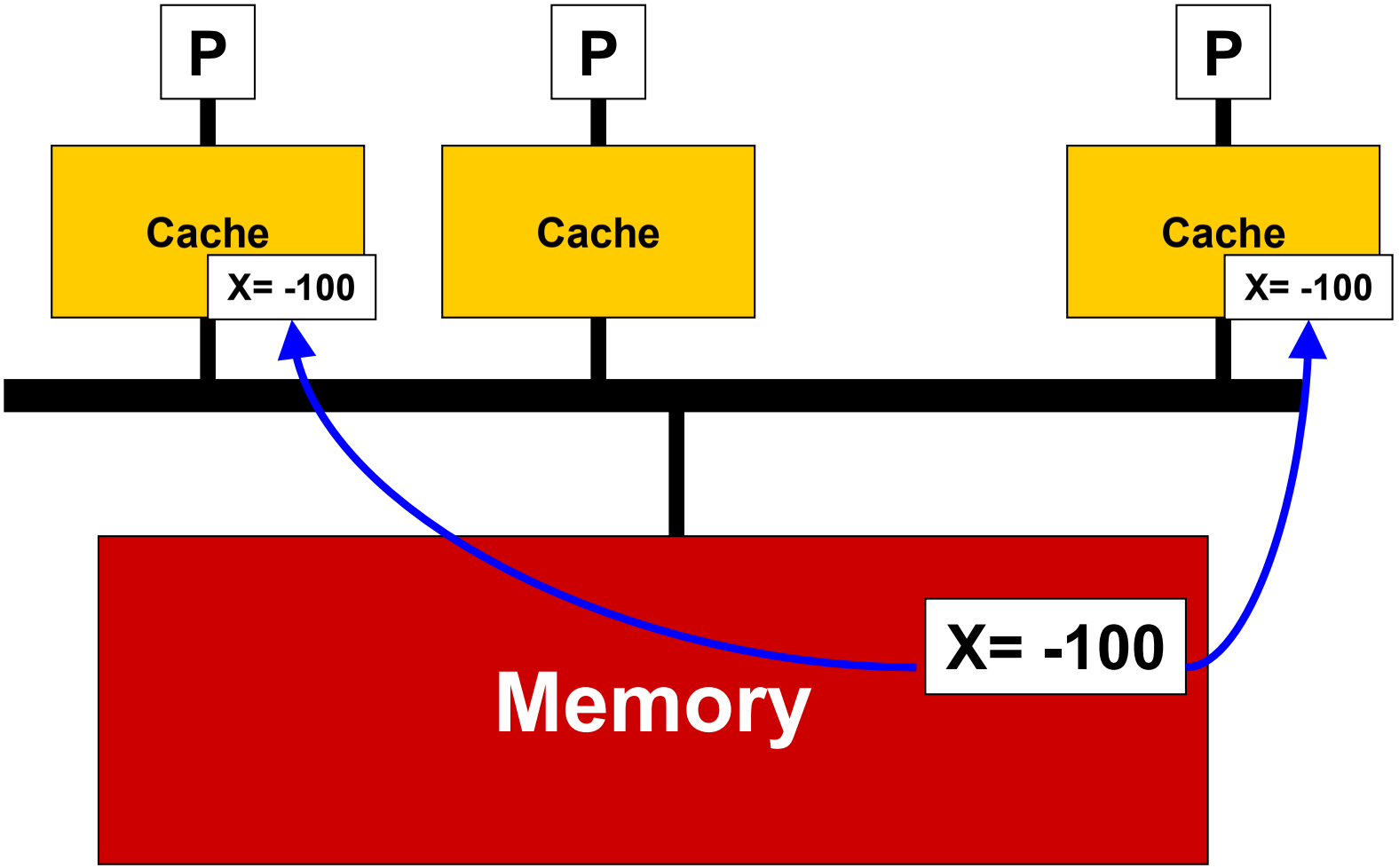
Intel Core2 Duo

L1	32 KB, 8-Way, 64 Byte/Line, LRU, WB 3 Cycle Latency
L2	4.0 MB, 16-Way, 64 Byte/Line, LRU, WB 14 Cycle Latency



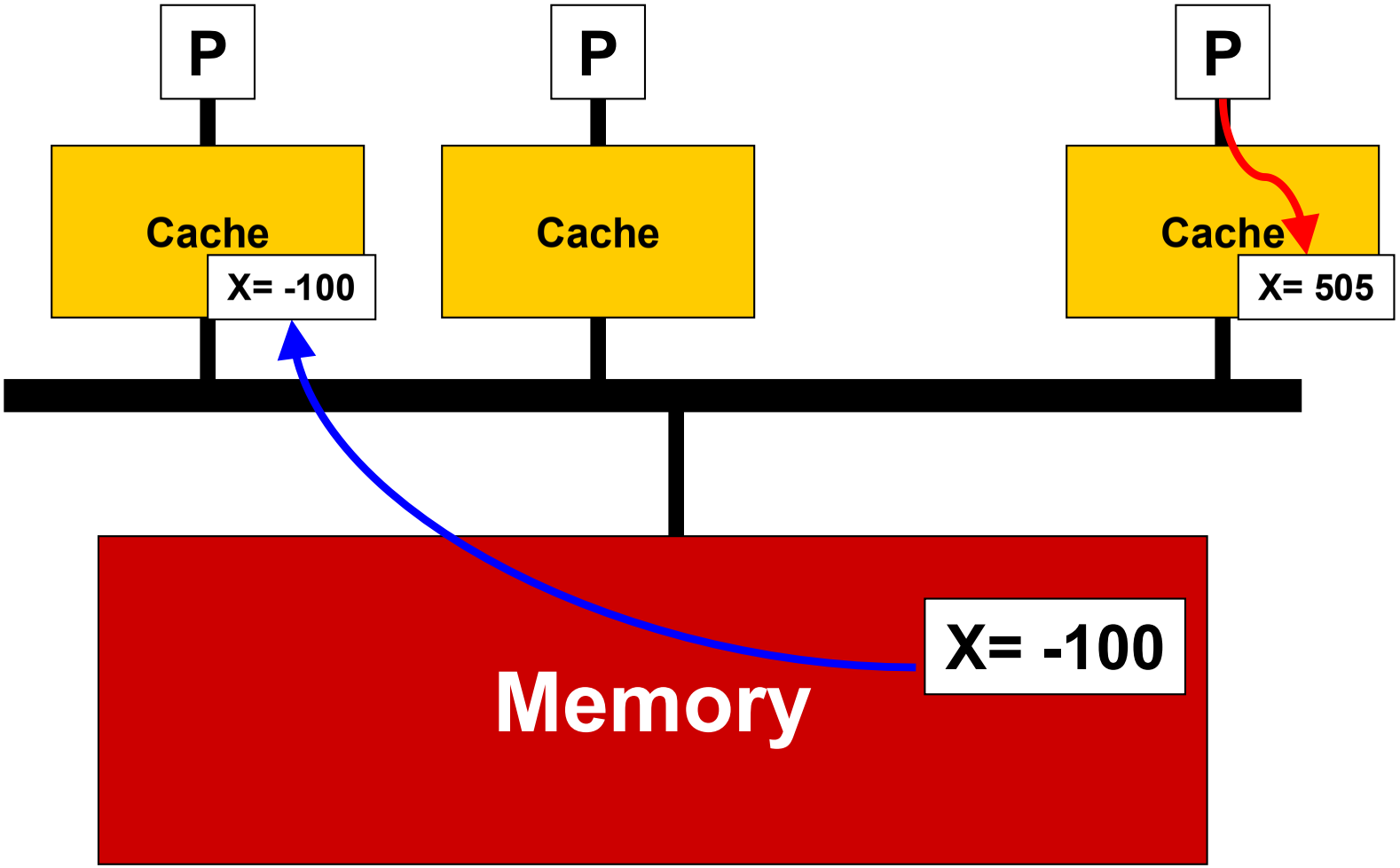
What happens if two cores are running the same program and modifying the same variable?

Example (Writeback Cache)



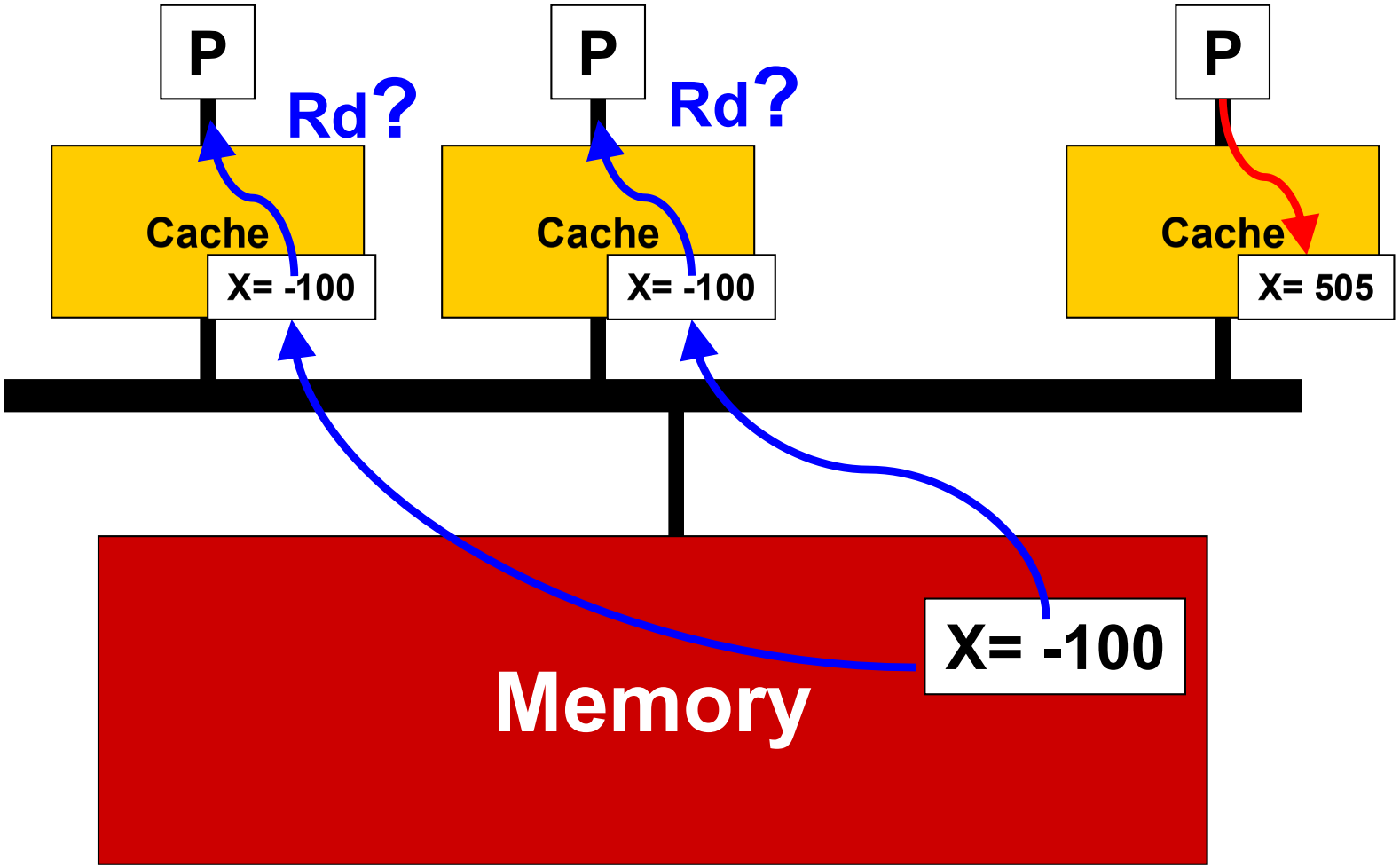
- Closest cache level is private
- Multiple copies of cache line can be present across different cores
- Local updates may lead to incoherent state

Example (Writeback Cache)



- Closest cache level is private
- Multiple copies of cache line can be present across different cores
- Local updates may lead to incoherent state

Example (Writeback Cache)



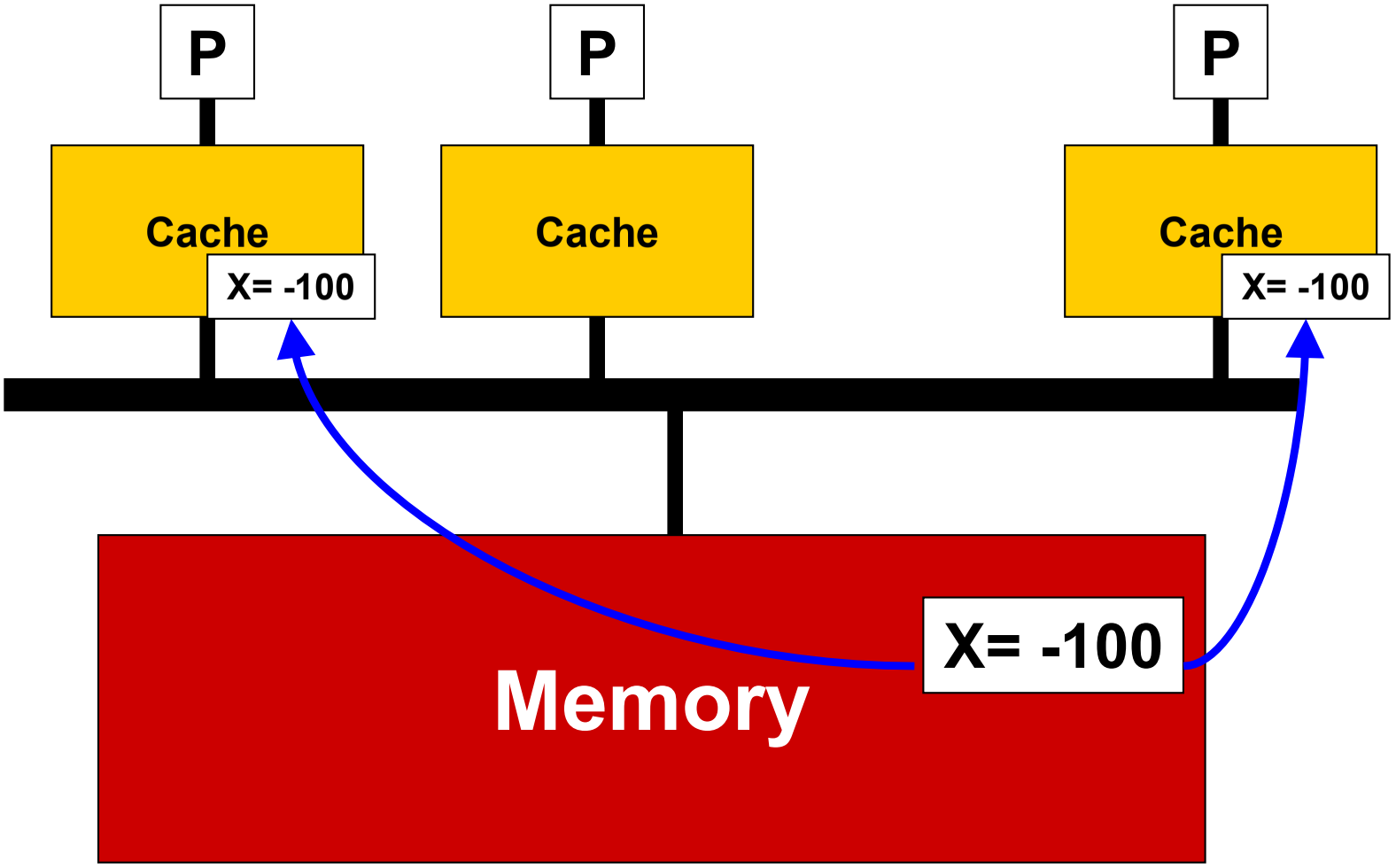
- Closest cache level is private
- Multiple copies of cache line can be present across different cores
- Local updates may lead to incoherent state

Cache Coherence Protocol

- Write propagation
  - Writes are visible to other processes
- All the writes will be shown as a transaction on the shared bus to memory

- Two protocols
  - Update-based Protocol
  - Invalidation-based Protocol

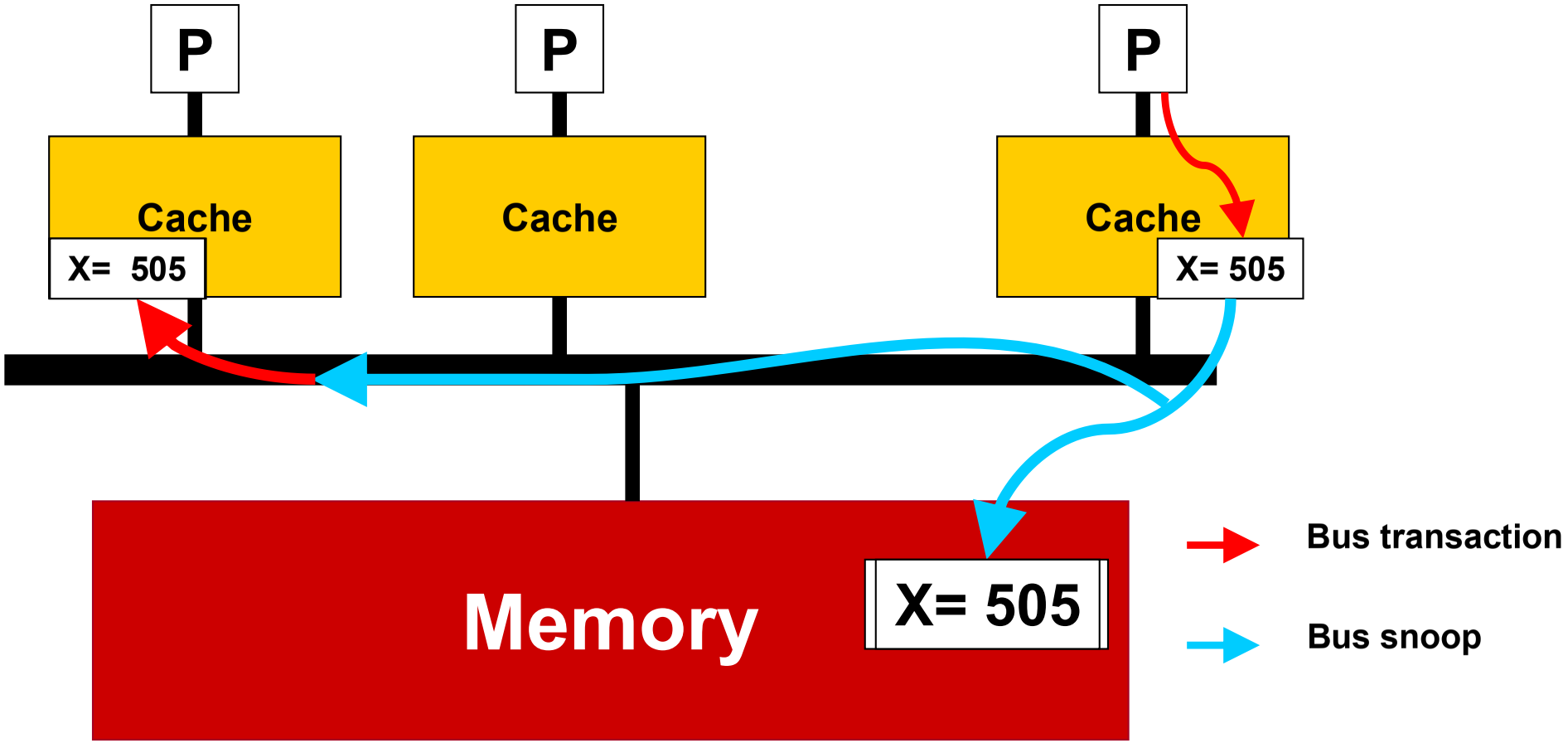
Example (Writeback Cache)



- Closest cache level is private
- Multiple copies of cache line can be present across different cores
- Local updates may lead to incoherent state

Bus Snooping

(Update-based Protocol on Write-Through cache)



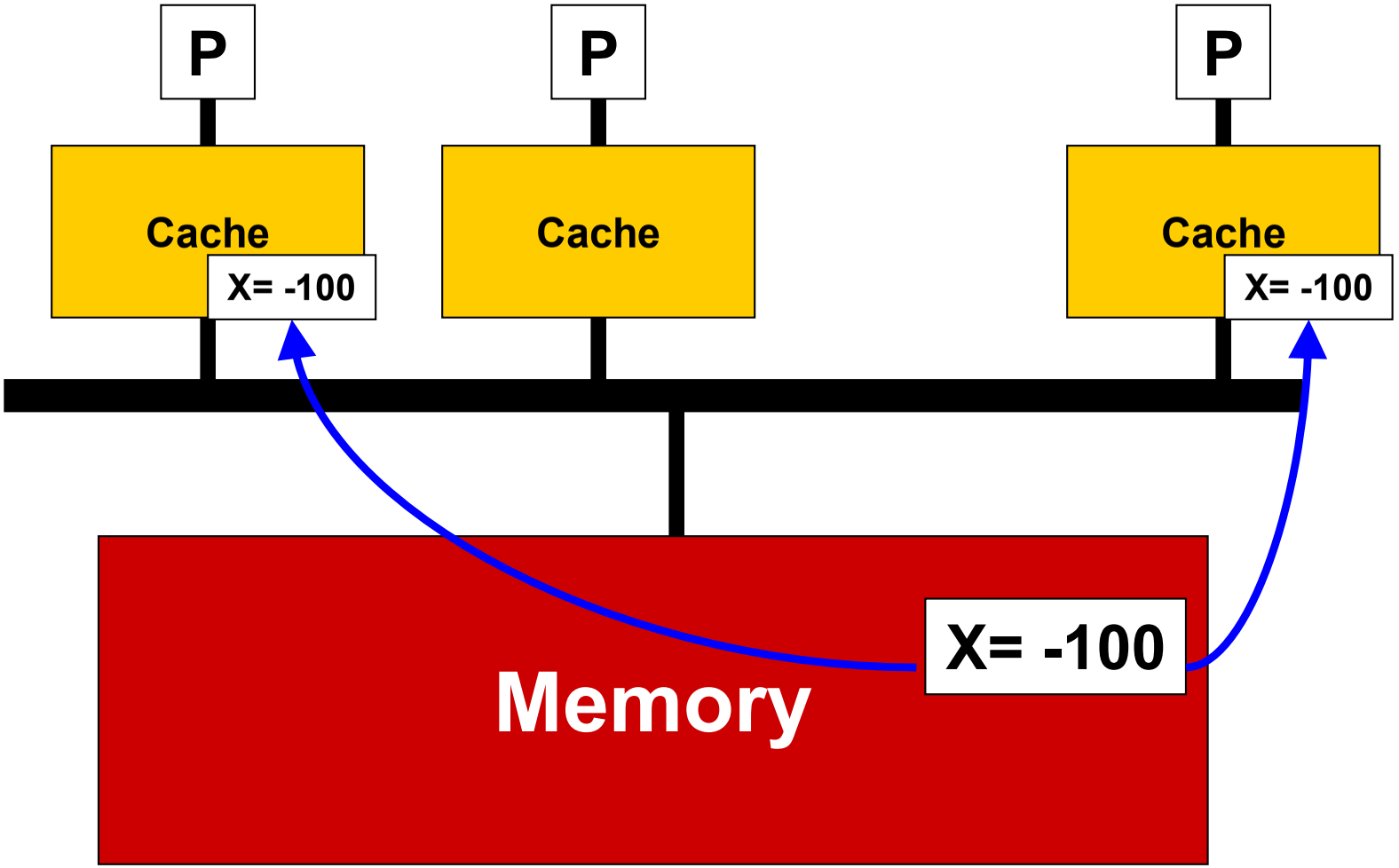
- Each processor's cache controller constantly snoops on the bus
- Update local copies upon snoop hit
- What is the drawback of update-based protocol?
  - Heavy traffic on the bus to supply both address and data

Cache Coherence Protocol

- Write propagation
  - Writes are visible to other processes
- All the writes will be shown as a transaction on the shared bus to memory

- Two protocols
  - Update-based Protocol
  - Invalidation-based Protocol

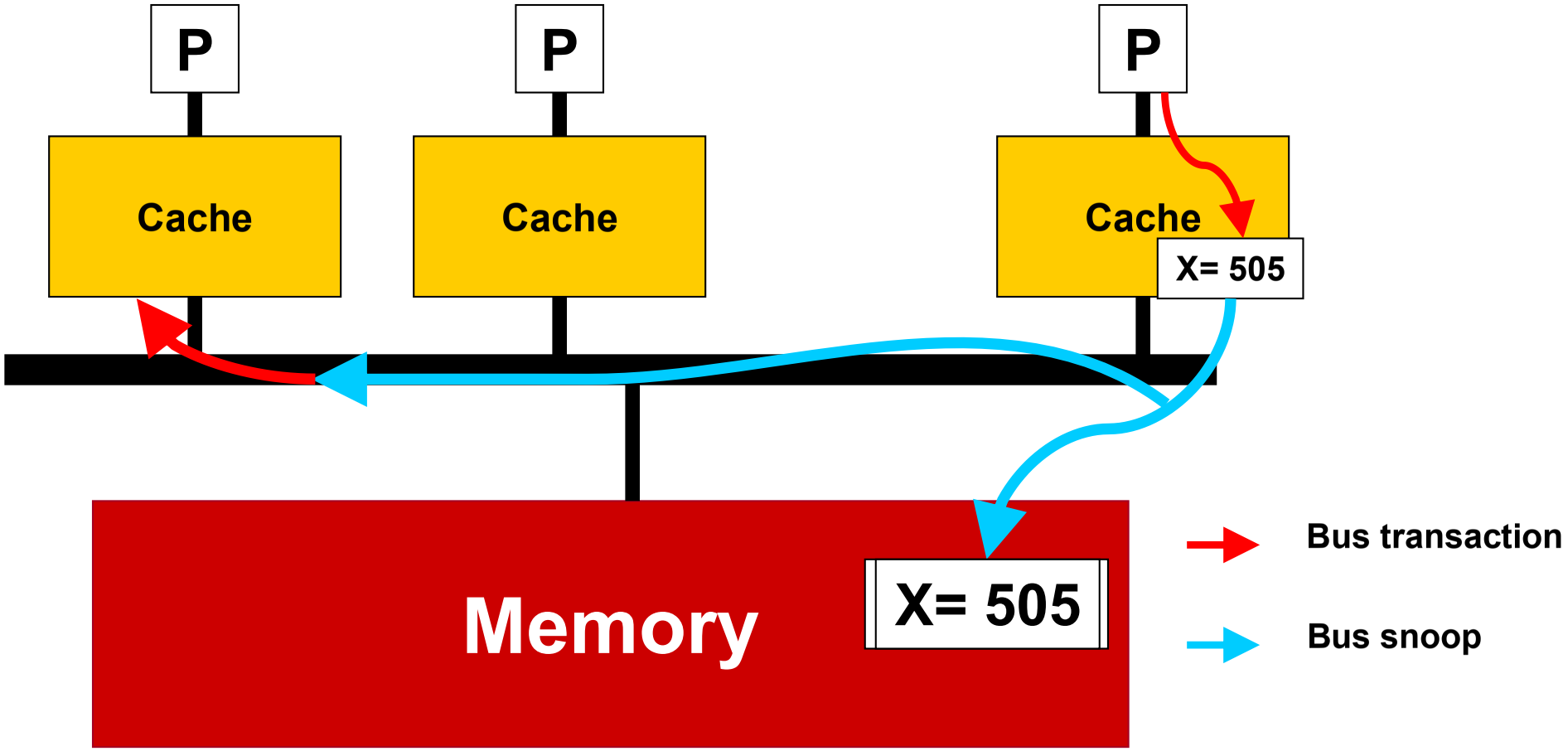
Example (Writeback Cache)



- Closest cache level is private
- Multiple copies of cache line can be present across different cores
- Local updates may lead to incoherent state

Bus Snooping

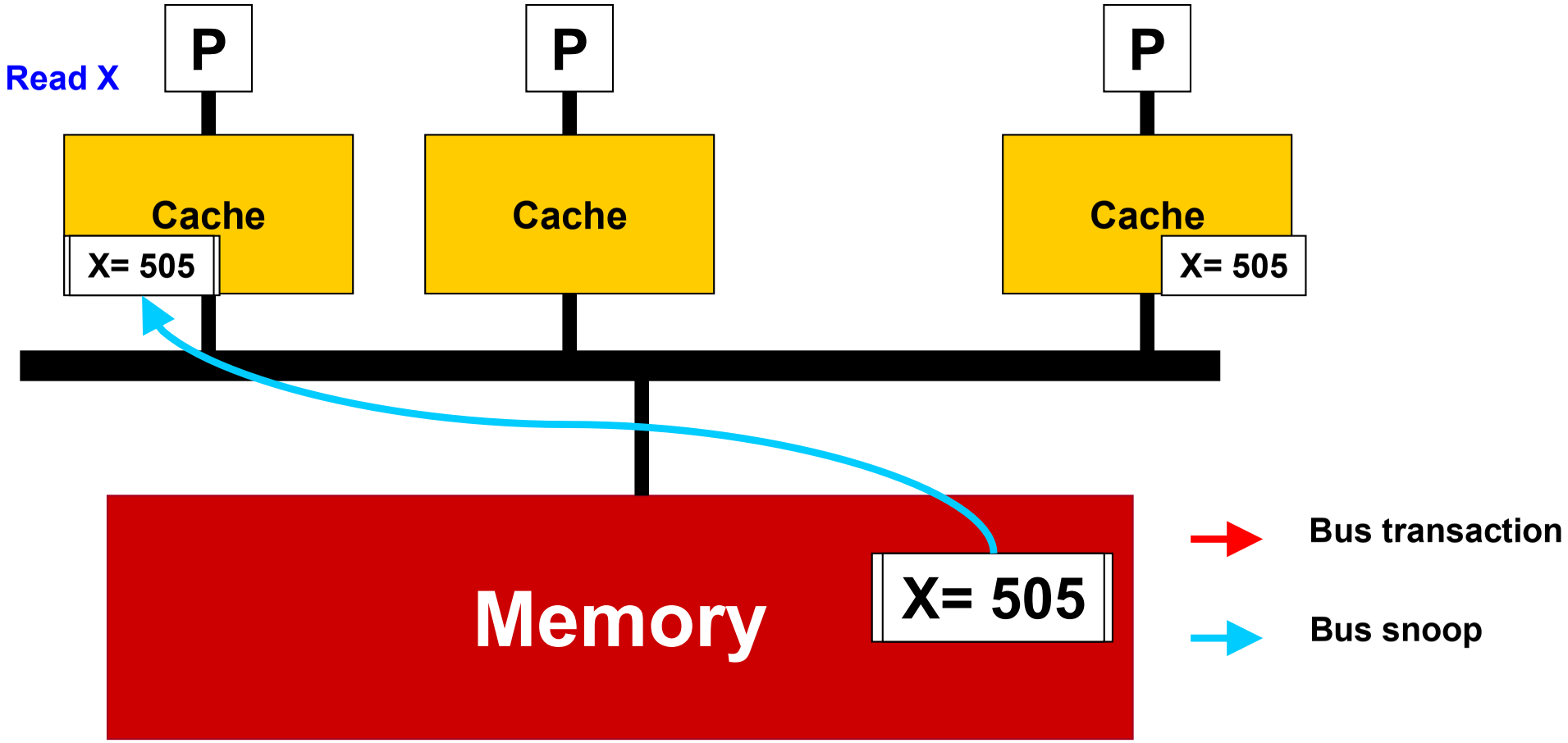
(Invalidation-based Protocol on Write-Through cache)



- Each processor's cache controller constantly snoops on the bus
- Invalidate local copies upon snoop hit
- What is the drawback of write-through cache based protocol?
  - Frequent access to memory

Bus Snooping

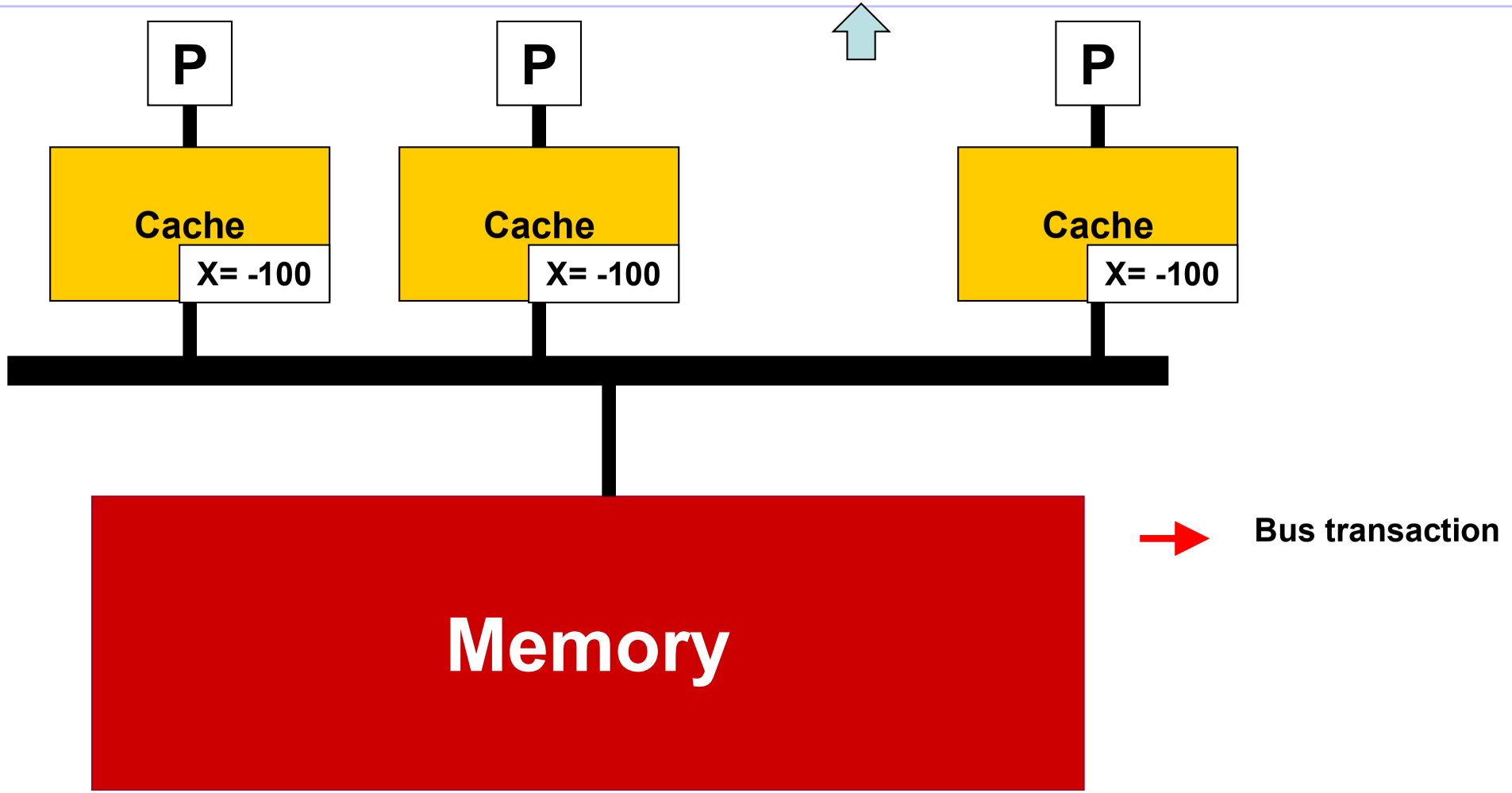
(Invalidation-based Protocol on Write-Through cache)



- Each processor's cache controller constantly snoops on the bus
- Invalidate local copies upon snoop hit
- What is the drawback of write-through cache based protocol?
  - Frequent access to memory

Cache Coherence Protocol

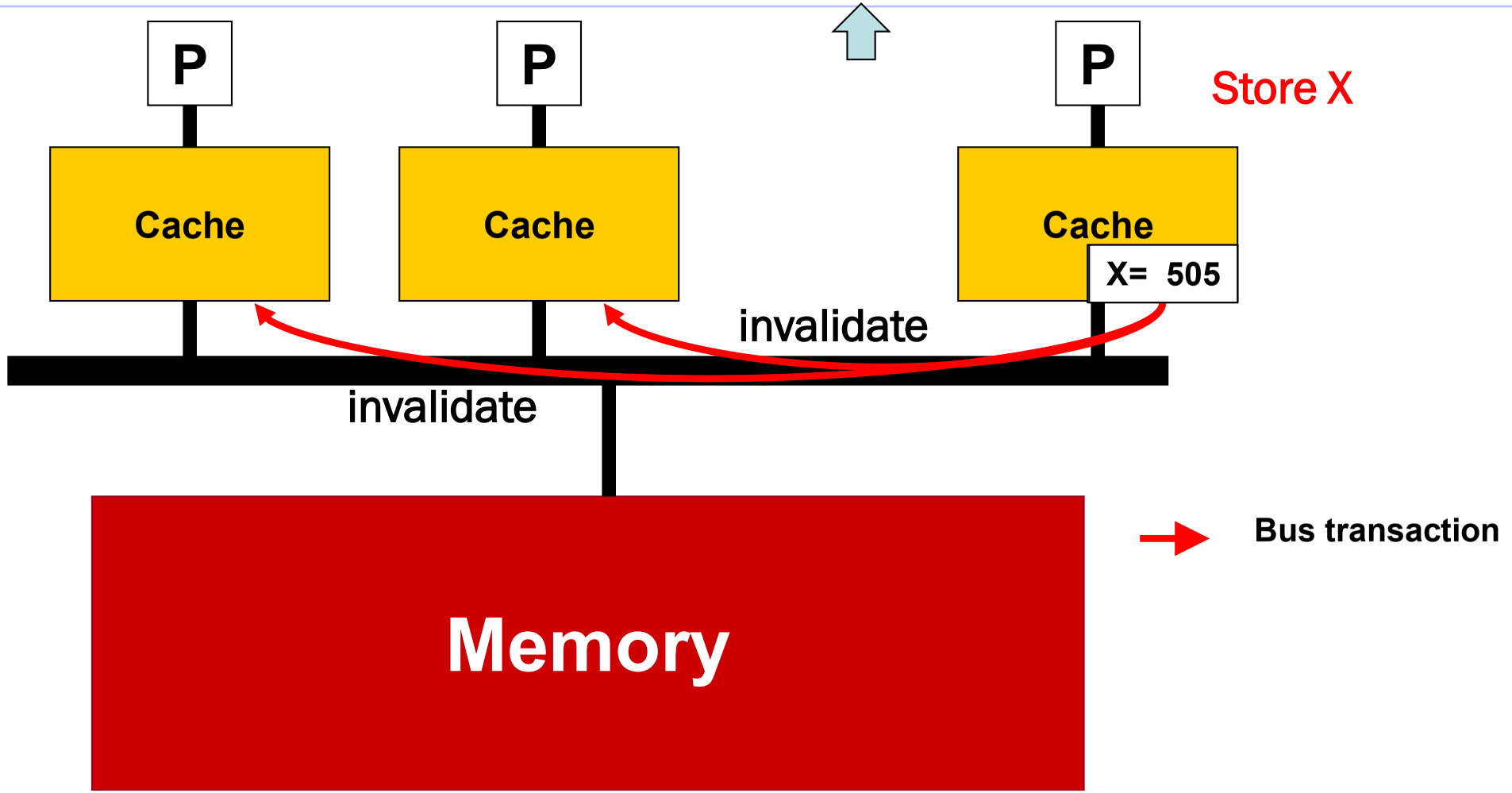
(Invalidation-based Protocol on Writeback cache)



- Invalidate the data copies for the sharing processor nodes
- Reduced traffic when a processor node keeps updating the same memory location

Cache Coherence Protocol

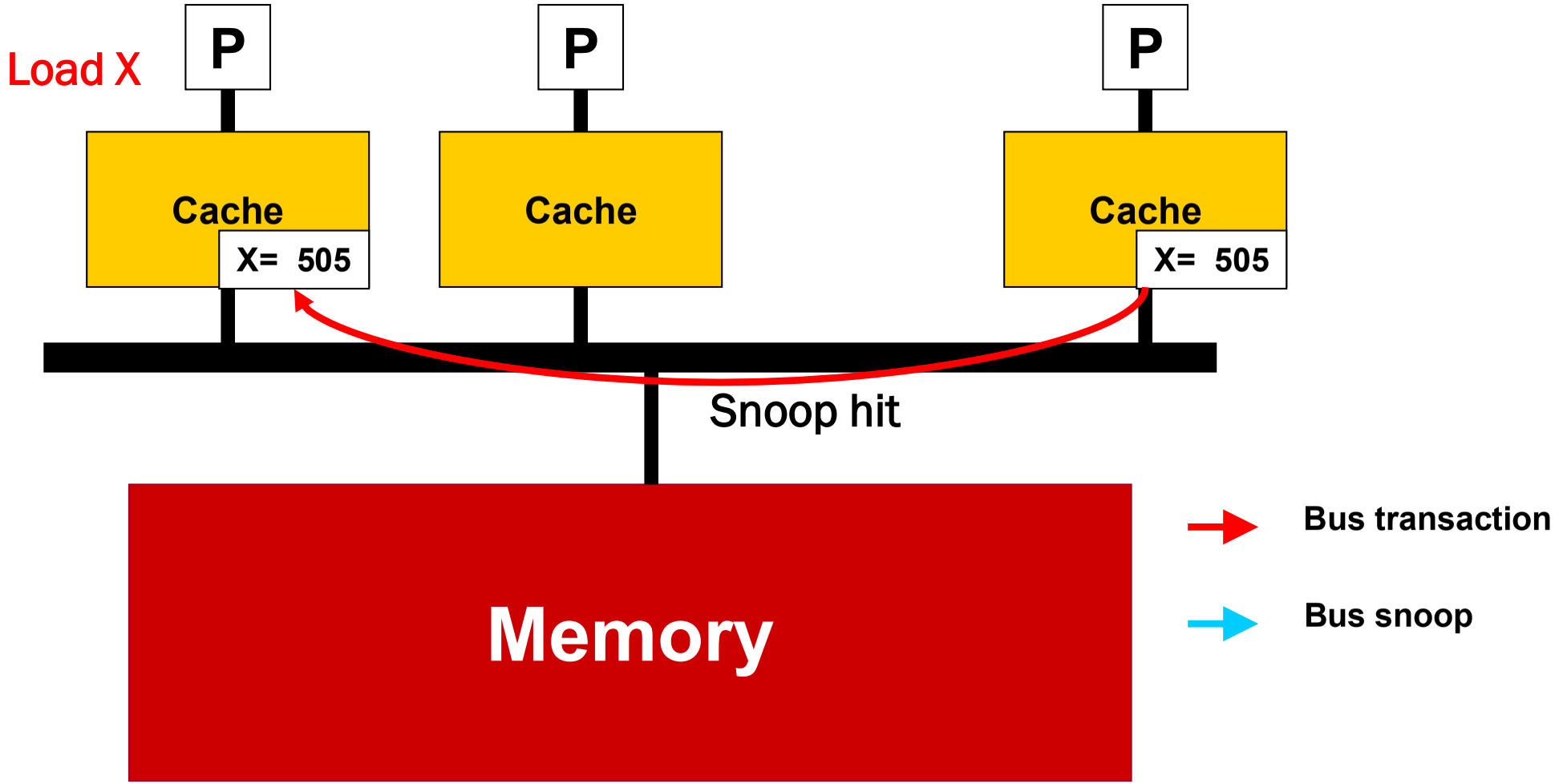
(Invalidation-based Protocol on Writeback cache)



- Invalidate the data copies for the sharing processor nodes
- Reduced traffic when a processor node keeps updating the same memory location

Cache Coherence Protocol

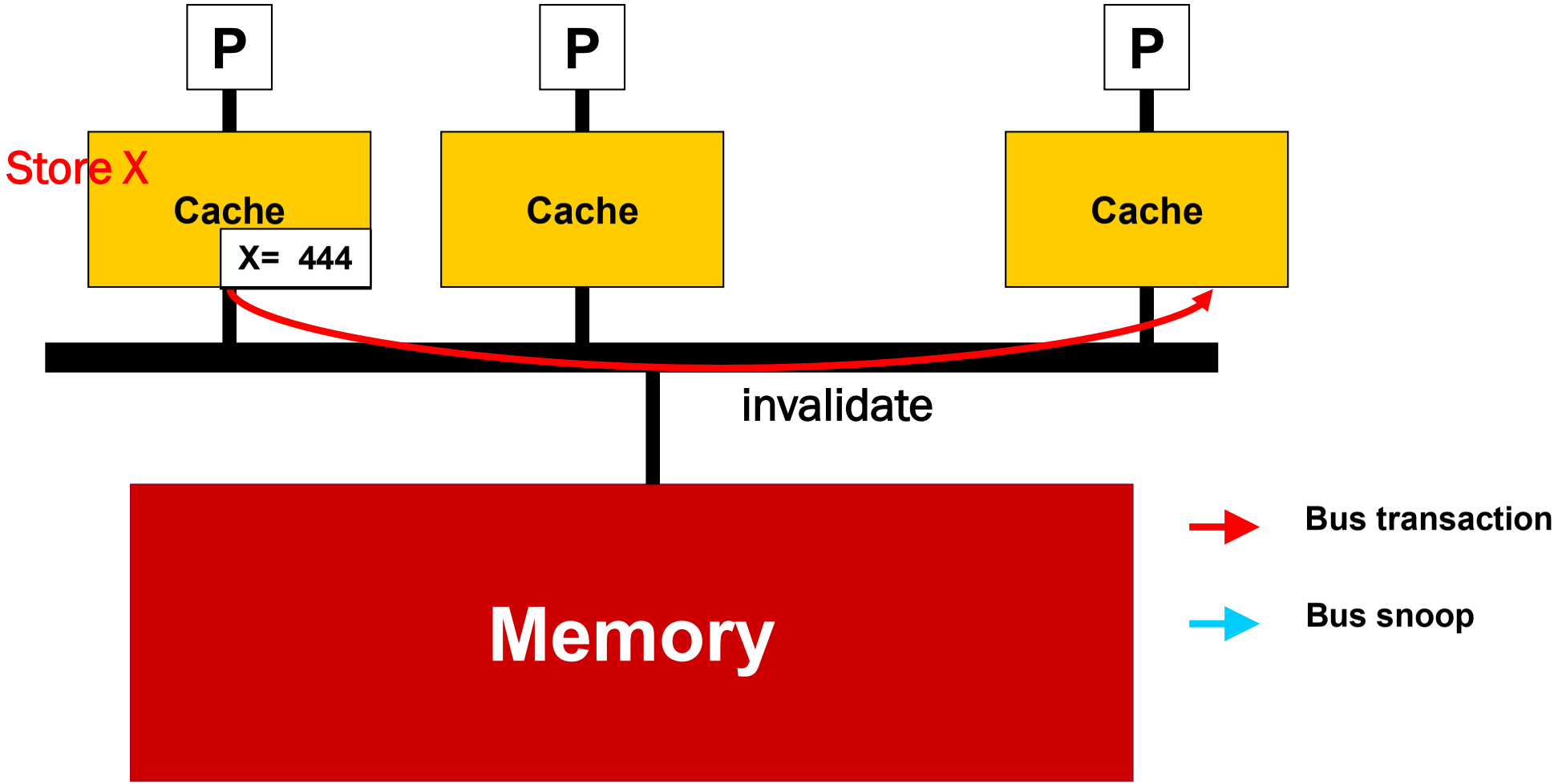
(Invalidation-based Protocol on Writeback cache)



- Invalidate the data copies for the sharing processor nodes
- Reduced traffic when a processor node keeps updating the same memory location

Cache Coherence Protocol

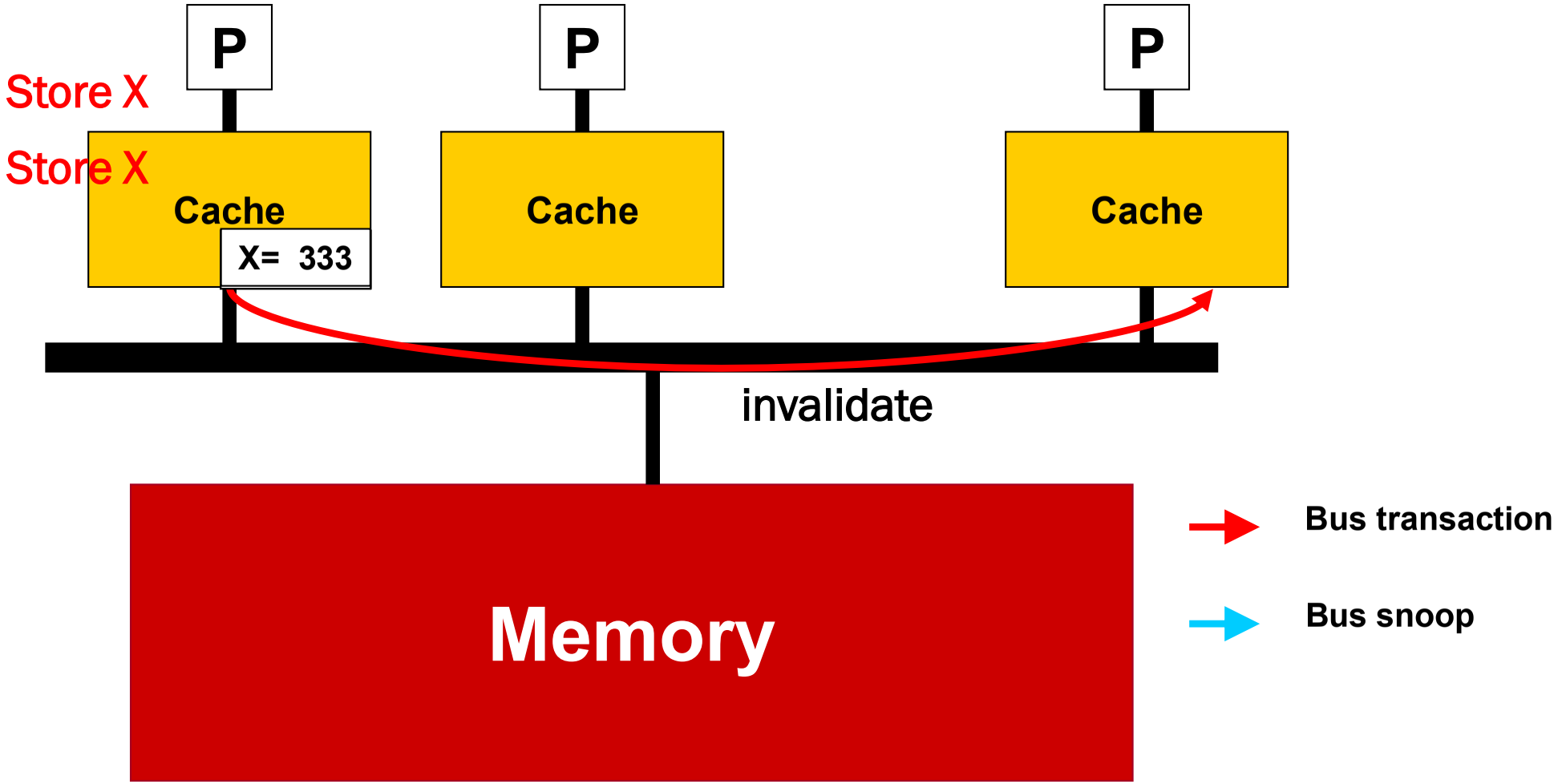
(Invalidation-based Protocol on Writeback cache)



- Invalidate the data copies for the sharing processor nodes
- Reduced traffic when a processor node keeps updating the same memory location

Cache Coherence Protocol

(Invalidation-based Protocol on Writeback cache)



- Invalidate the data copies for the sharing processor nodes
- Reduced traffic when a processor node keeps updating the same memory location

MSI Writeback Invalidation Protocol

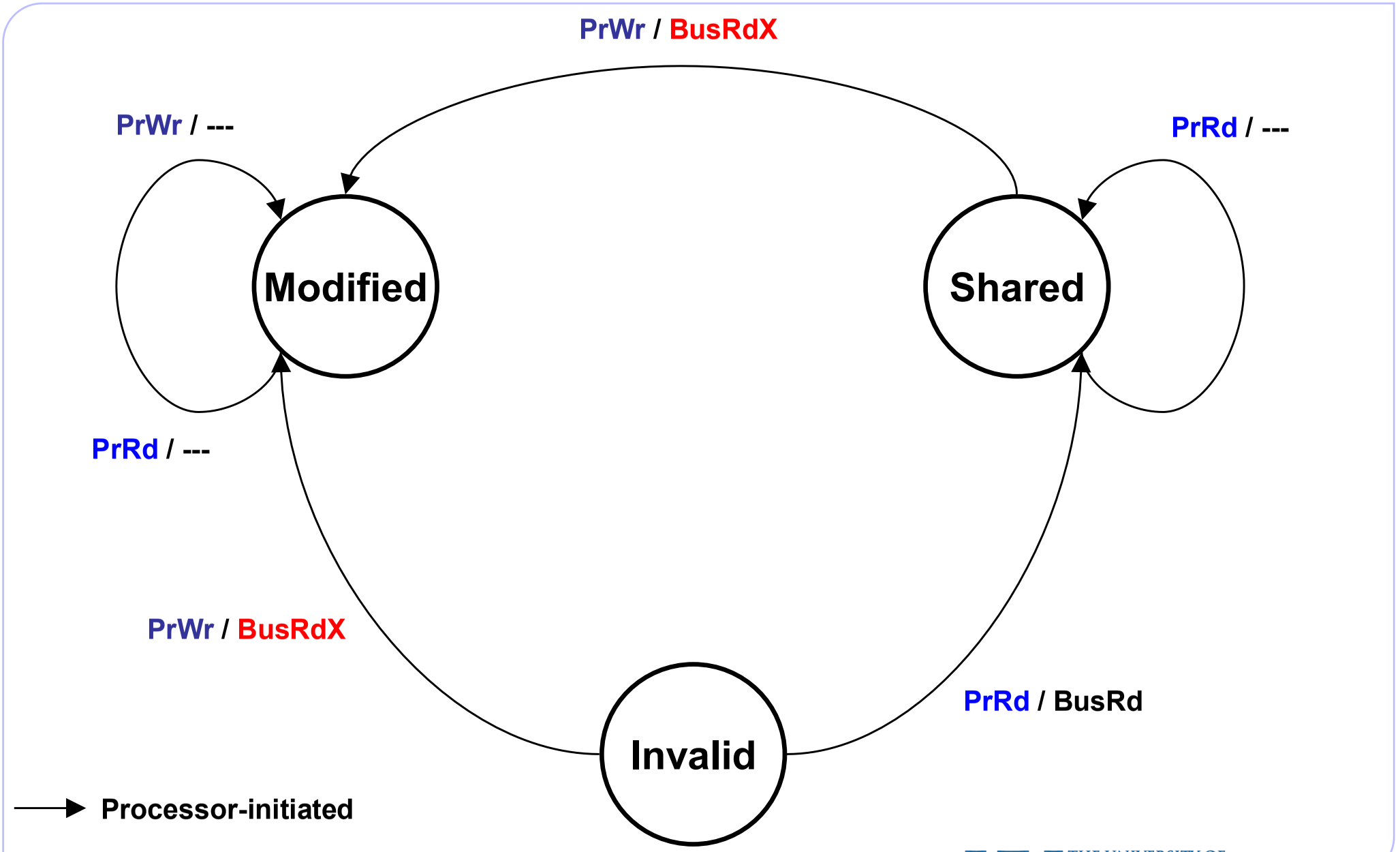
- Modified
  - Dirty
  - Only this cache has a valid copy
- Shared
  - Memory is consistent
  - One or more caches have a valid copy
- Invalid

- Writeback protocol: A cache line can be written multiple times before the memory is updated.

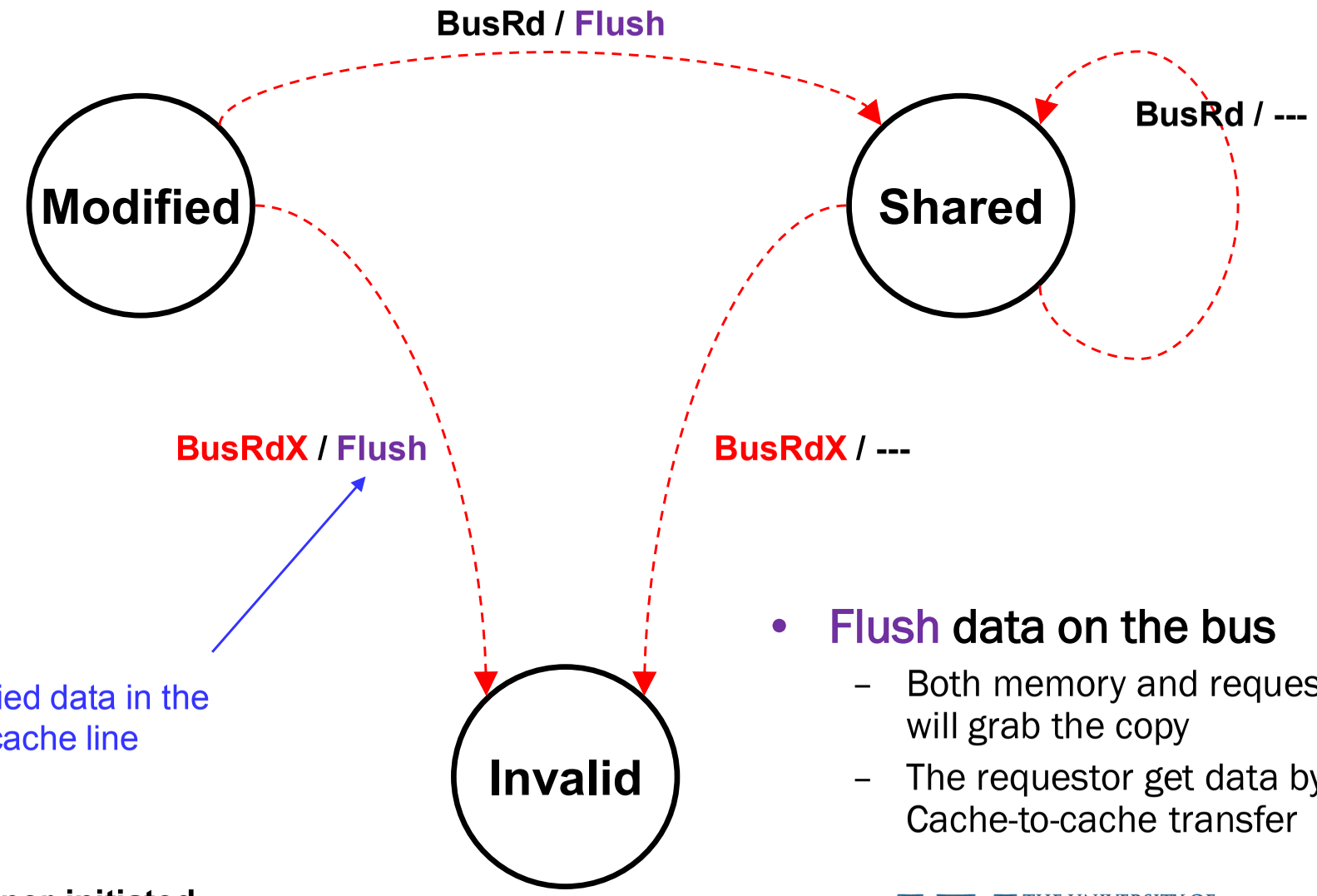
MSI Writeback Invalidation Protocol

- Two types of local request from the processor
  - PrRd
  - PrWr
- Two types of bus transactions post by cache controller
  - BusRd
    - PrRd misses the cache
    - Memory or another cache supplies the line
  - BusRd eXclusive (Read-to-own)
    - PrWr is issued to a line which is not in the Modified state

MSI Writeback Invalidation Protocol (Initiated by Current Processor)

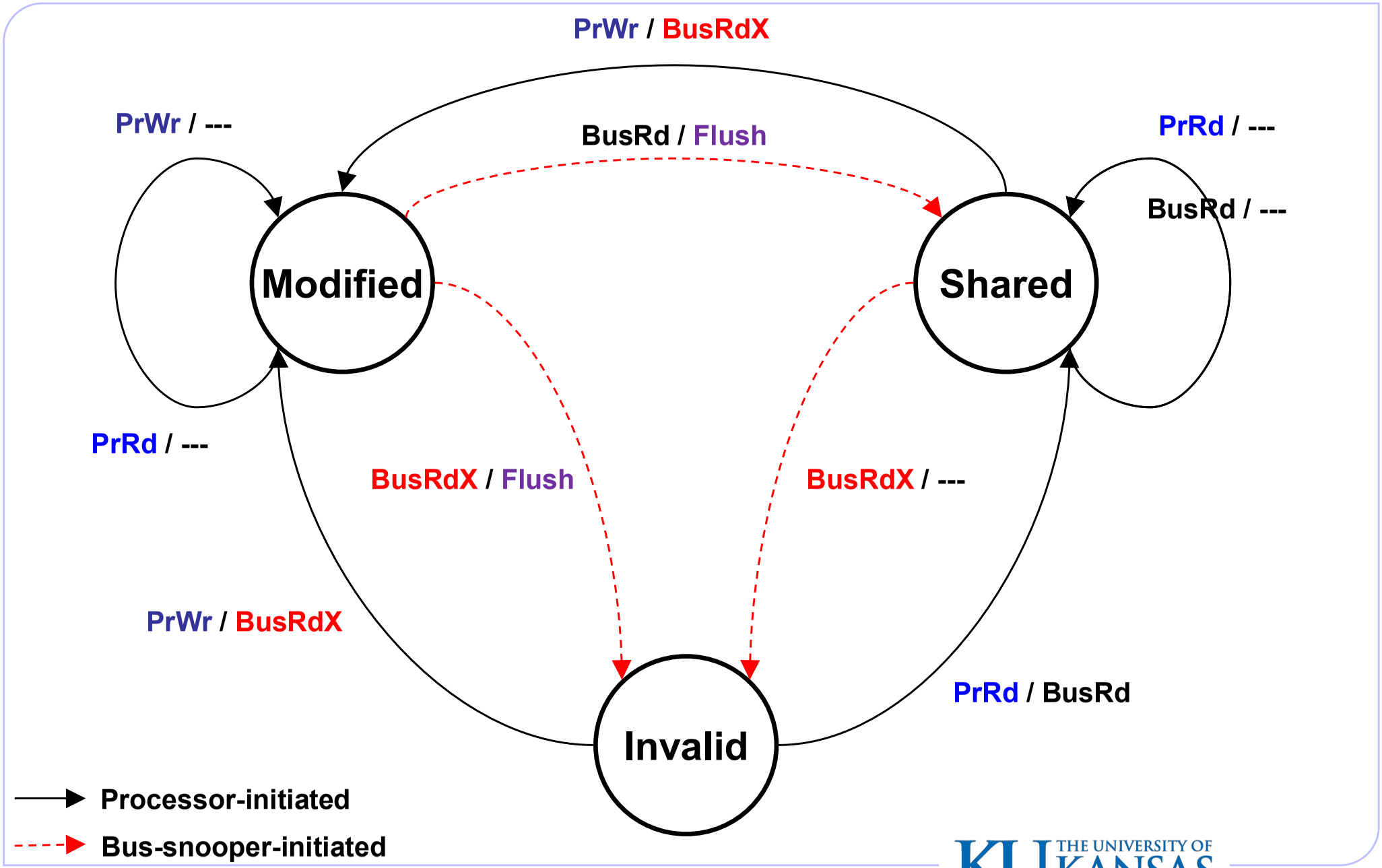


MSI Writeback Invalidation Protocol (Bus Transaction Initiated by Other Processors)

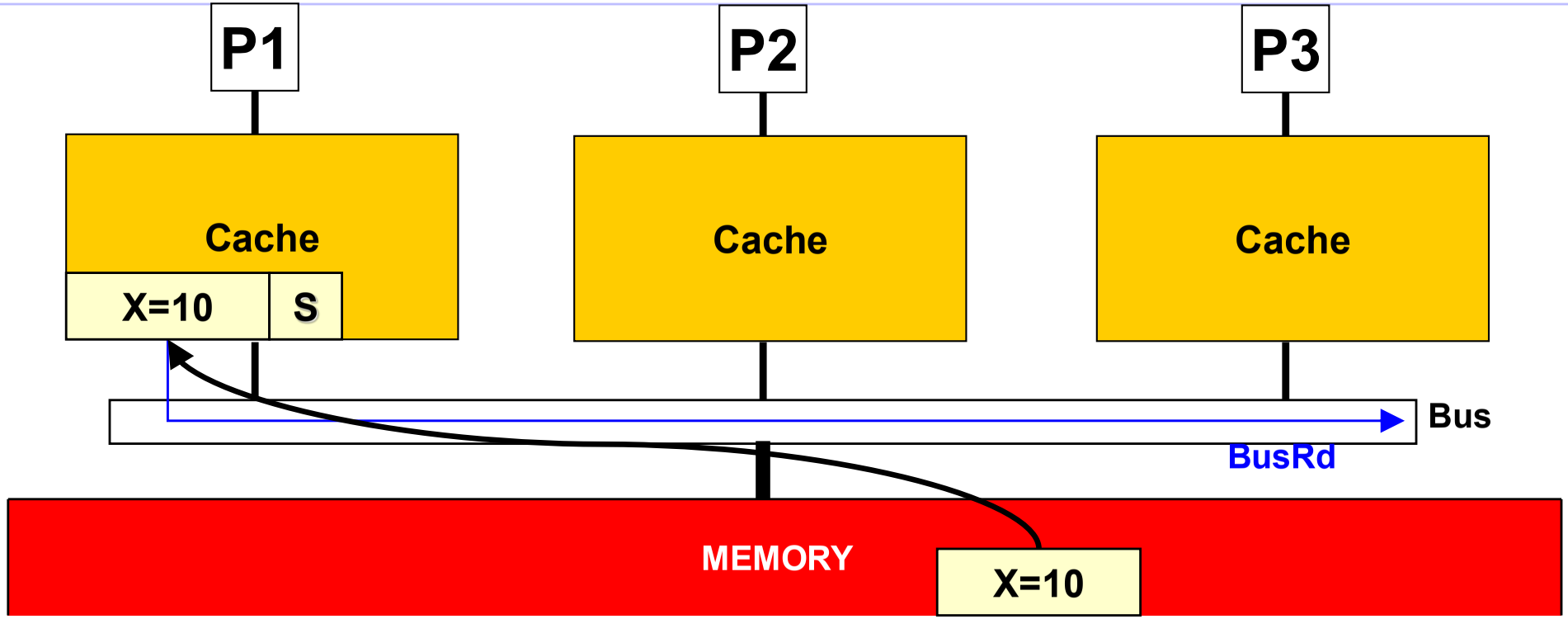


- Flush data on the bus
  - Both memory and requestor will grab the copy
  - The requestor get data by Cache-to-cache transfer

MSI Writeback Invalidation Protocol

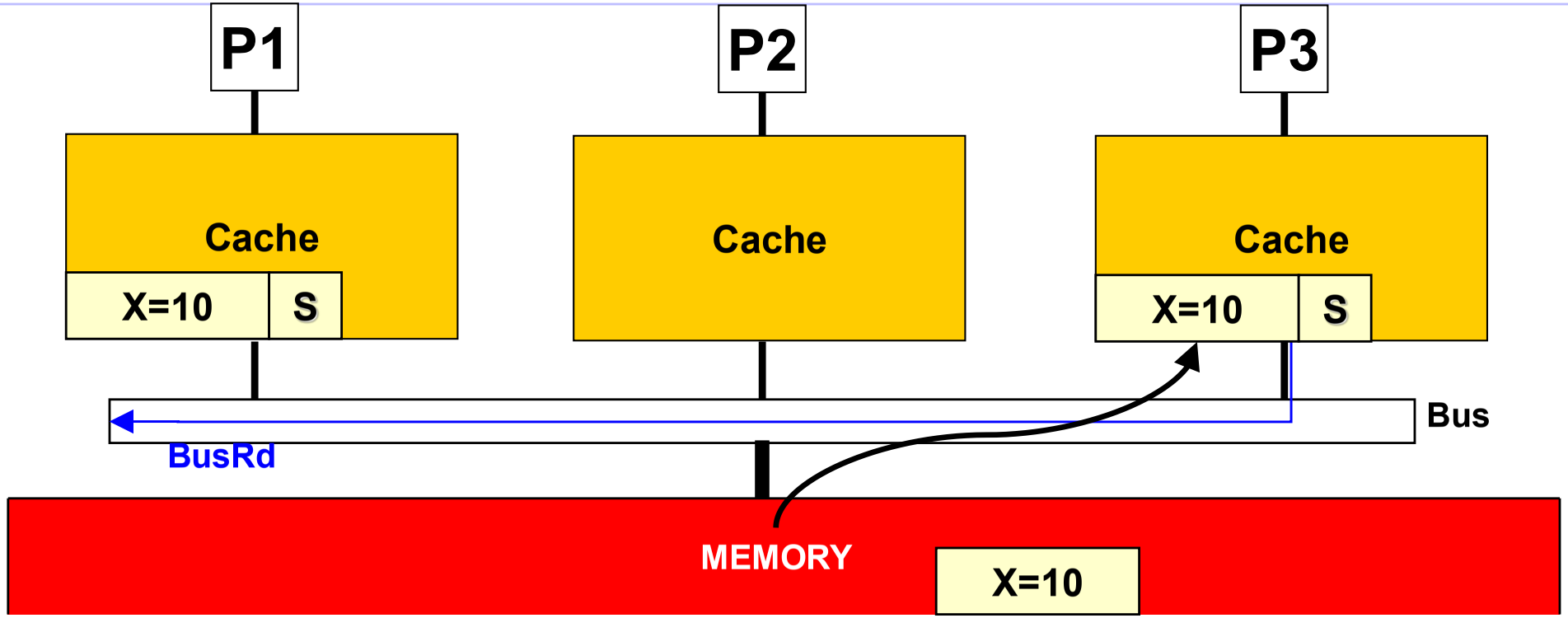


MSI Example



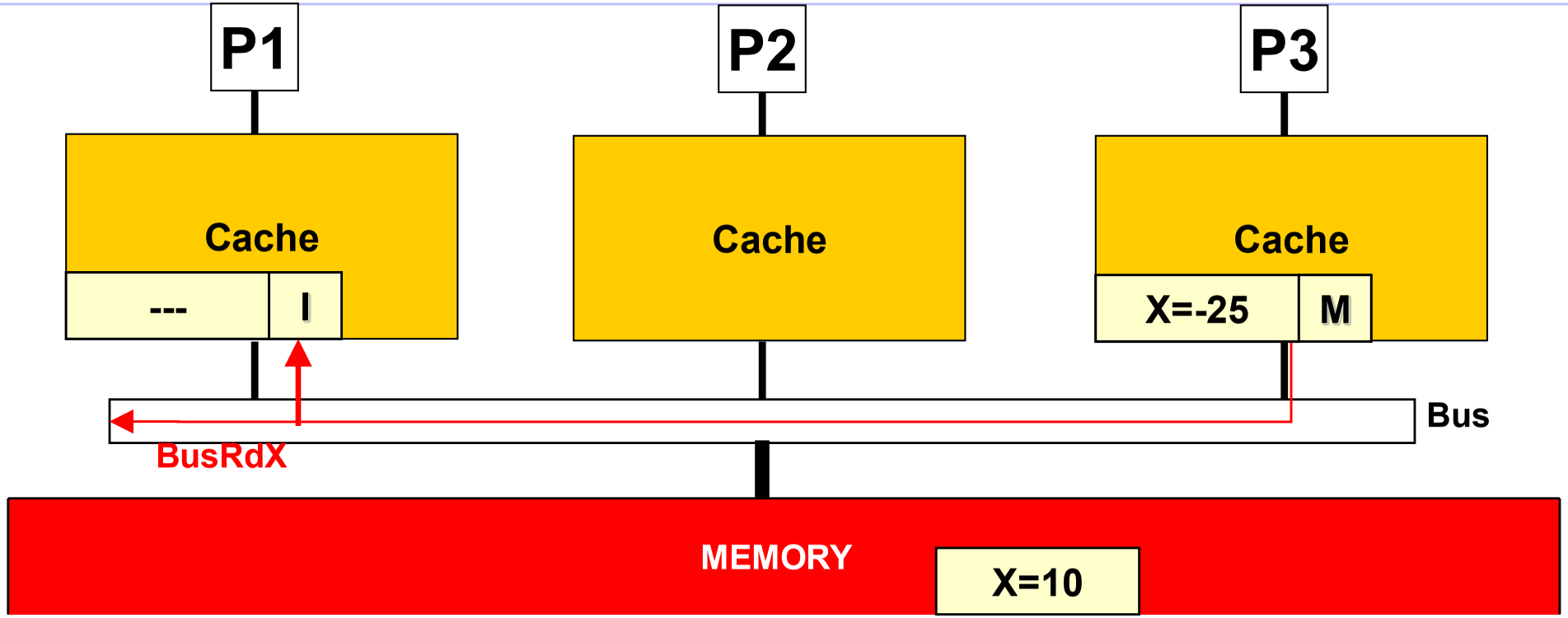
Processor Action	State in P1	State in P2	State in P3	Bus Transaction	Data Supplier
P1 reads X	S	I	I	BusRd	Memory

MSI Example



Processor Action	State in P1	State in P2	State in P3	Bus Transaction	Data Supplier
P1 reads X	S	I	I	BusRd	Memory
P3 reads X	S	I	S	BusRd	Memory

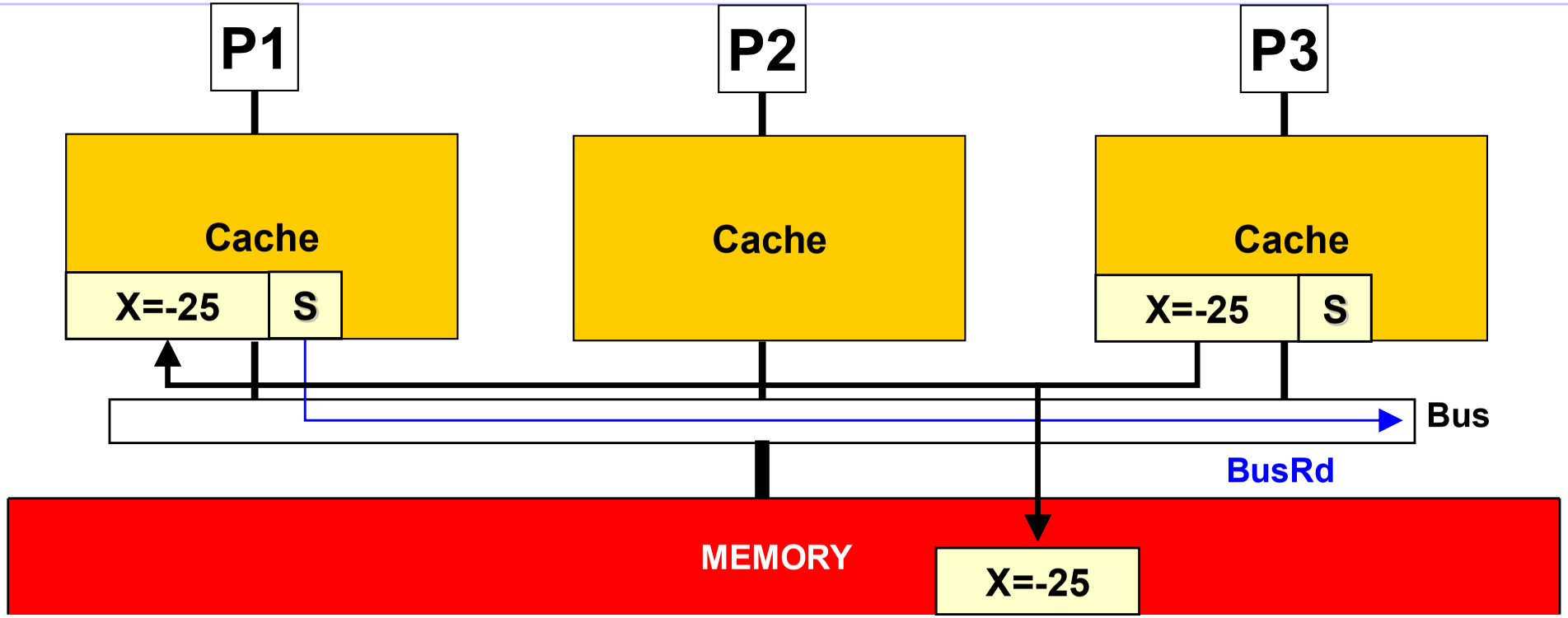
MSI Example



Processor Action	State in P1	State in P2	State in P3	Bus Transaction	Data Supplier
P1 reads X	S	I	I	BusRd	Memory
P3 reads X	S	I	S	BusRd	Memory
P3 writes X	I	I	M	BusRdX	---

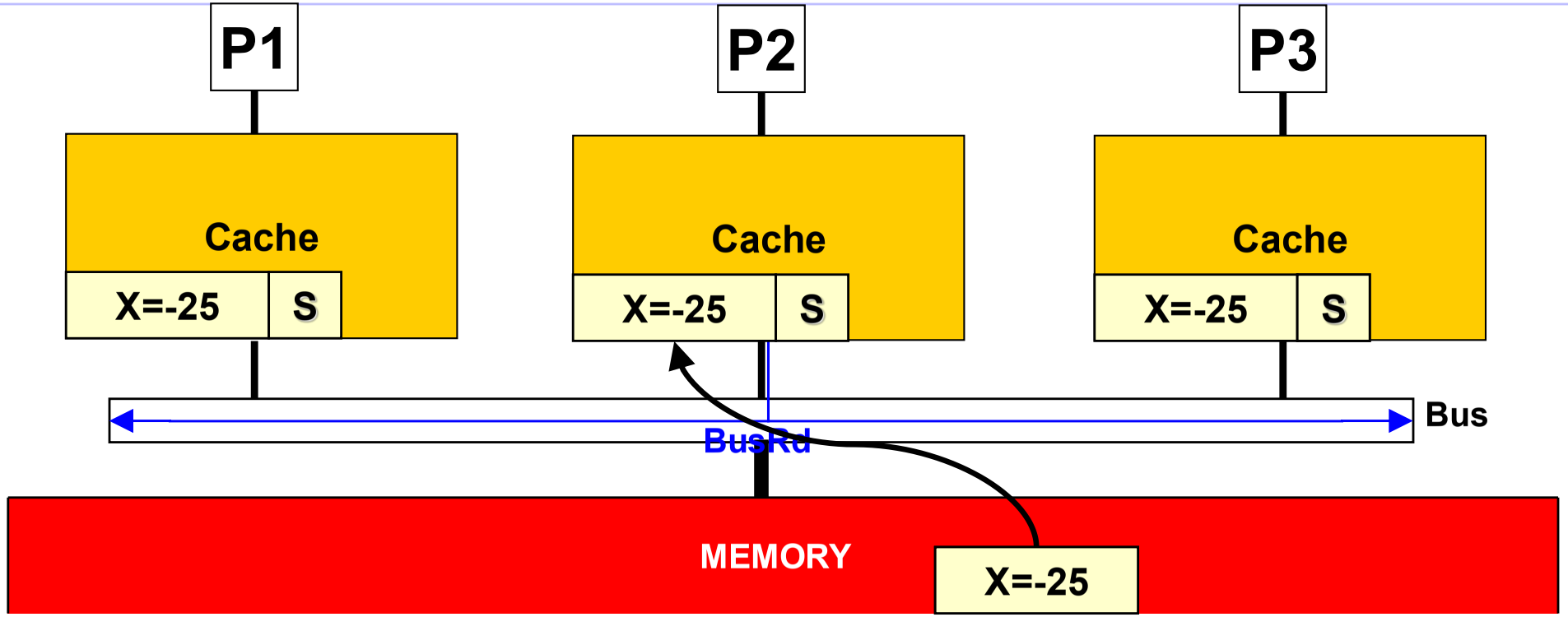
MSI Example

Both P1 and memory get data during the flush



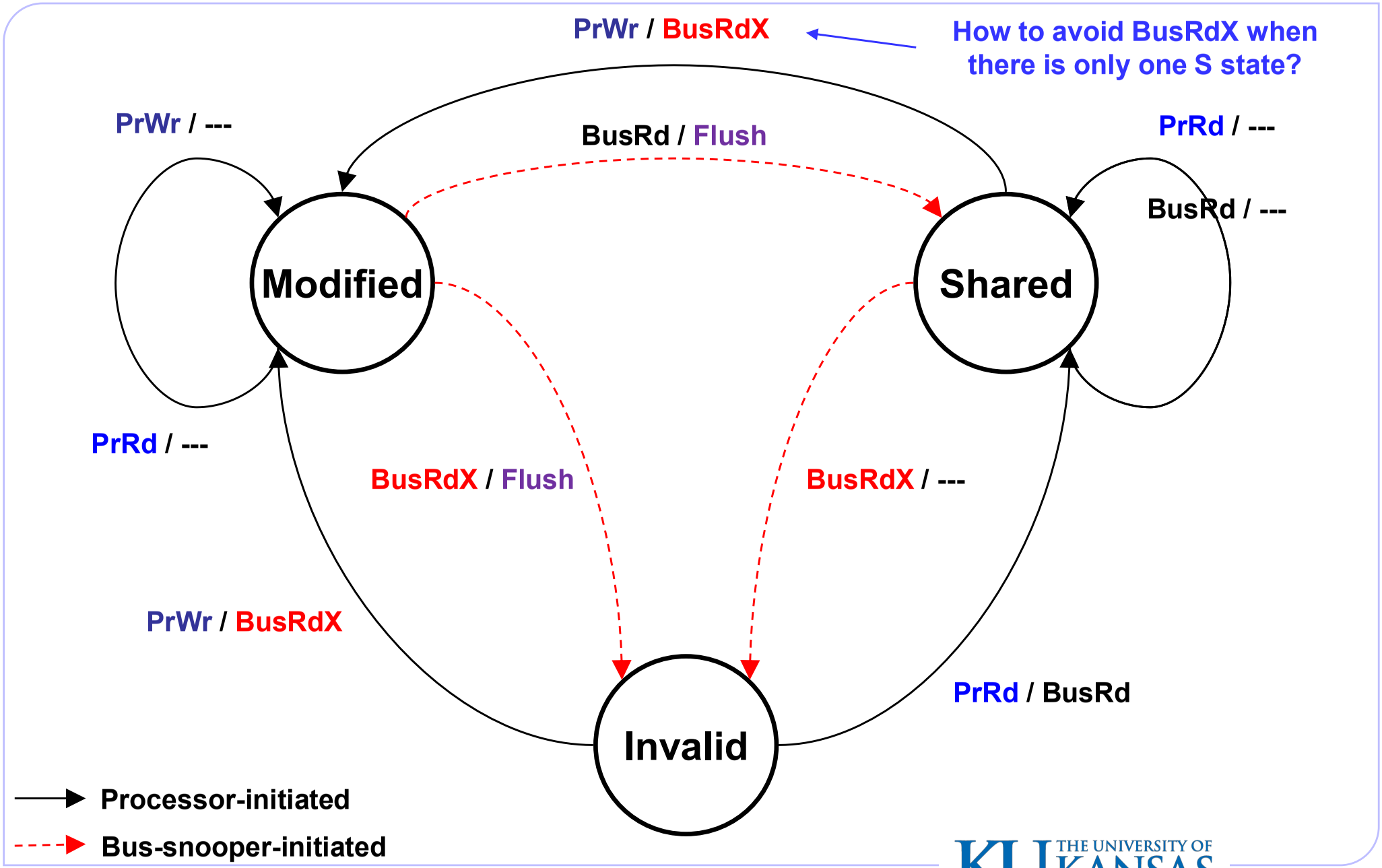
Processor Action	State in P1	State in P2	State in P3	Bus Transaction	Data Supplier
P1 reads X	S	I	I	BusRd	Memory
P3 reads X	S	I	S	BusRd	Memory
P3 writes X	I	I	M	BusRdX	---
P1 reads X	S	I	S	BusRd	P3 Cache

MSI Example



Processor Action	State in P1	State in P2	State in P3	Bus Transaction	Data Supplier
P1 reads X	S	I	I	BusRd	Memory
P3 reads X	S	I	S	BusRd	Memory
P3 writes X	I	I	M	BusRdX	---
P1 reads X	S	I	S	BusRd	P3 Cache
P2 reads X	S	S	S	BusRd	Memory

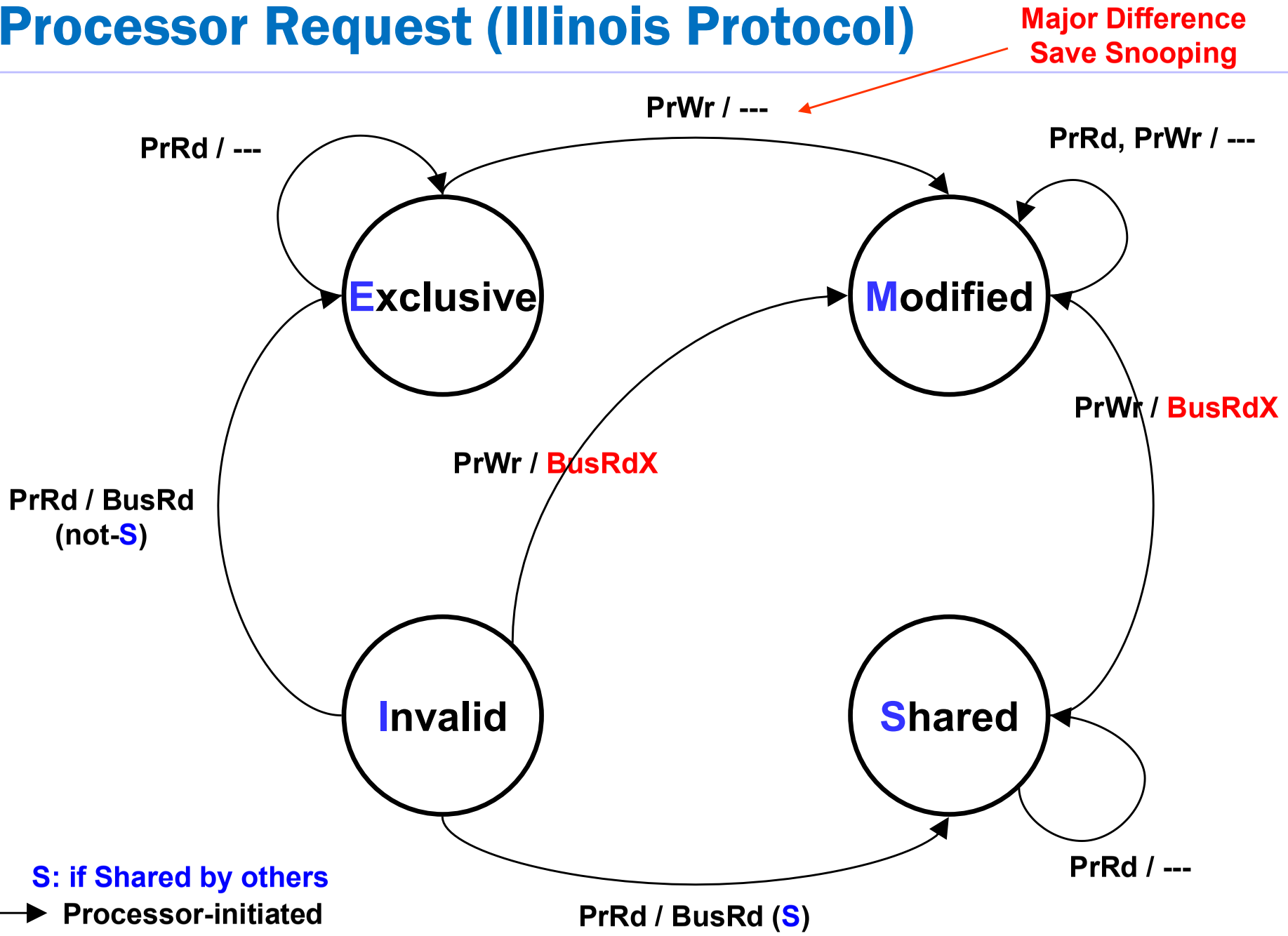
MSI Writeback Invalidation Protocol



MESI Writeback Invalidation Protocol

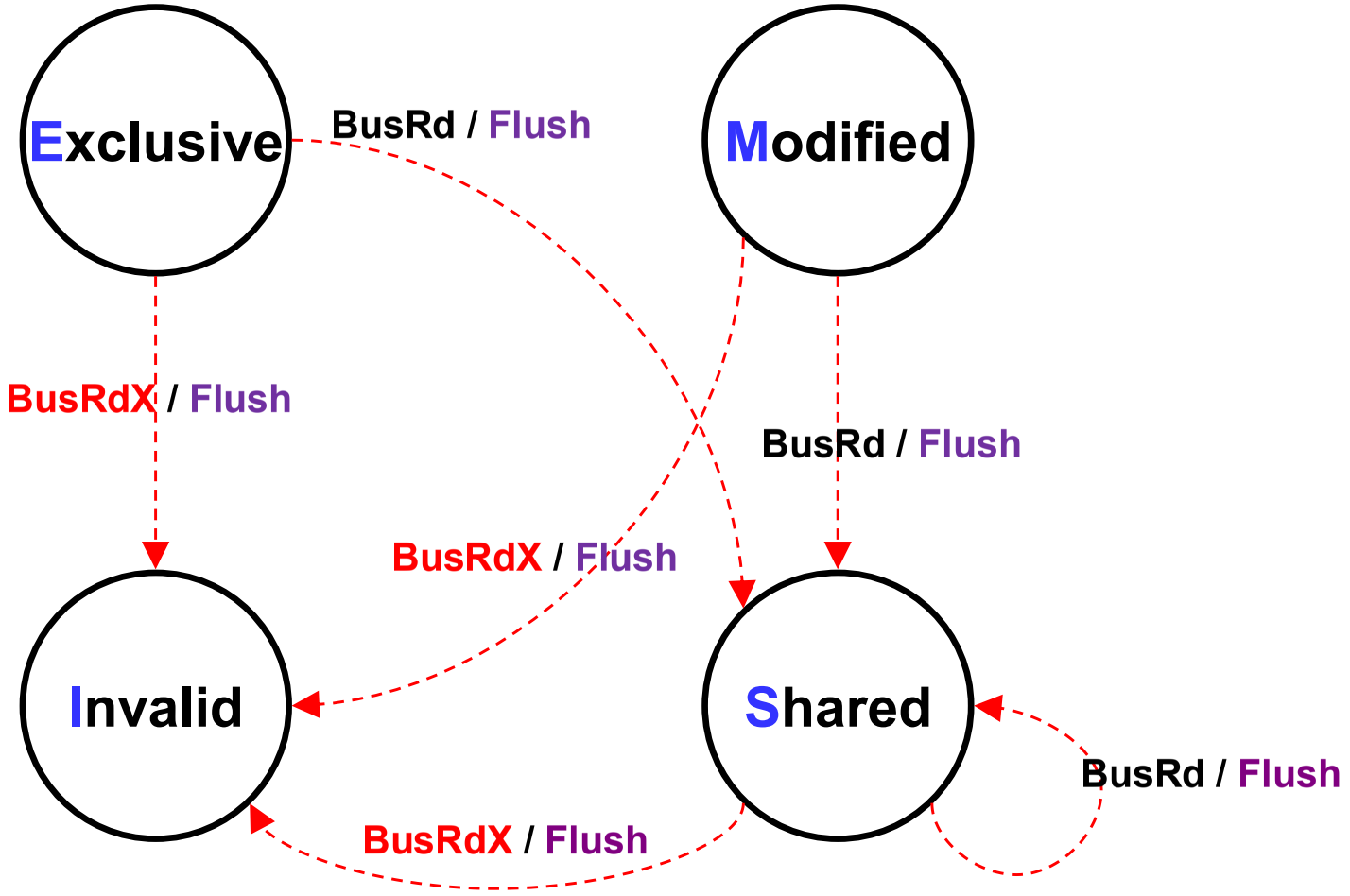
- Introduce the Exclusive state
  - One can write to the single copy without generating BusRdX
  - Very useful for running single-thread program
- Illinois Protocol: Proposed by Pamarcos and Patel in 1984
- Employed in Intel, PowerPC, MIPS

MESI Writeback Invalidation Protocol
Processor Request (Illinois Protocol)

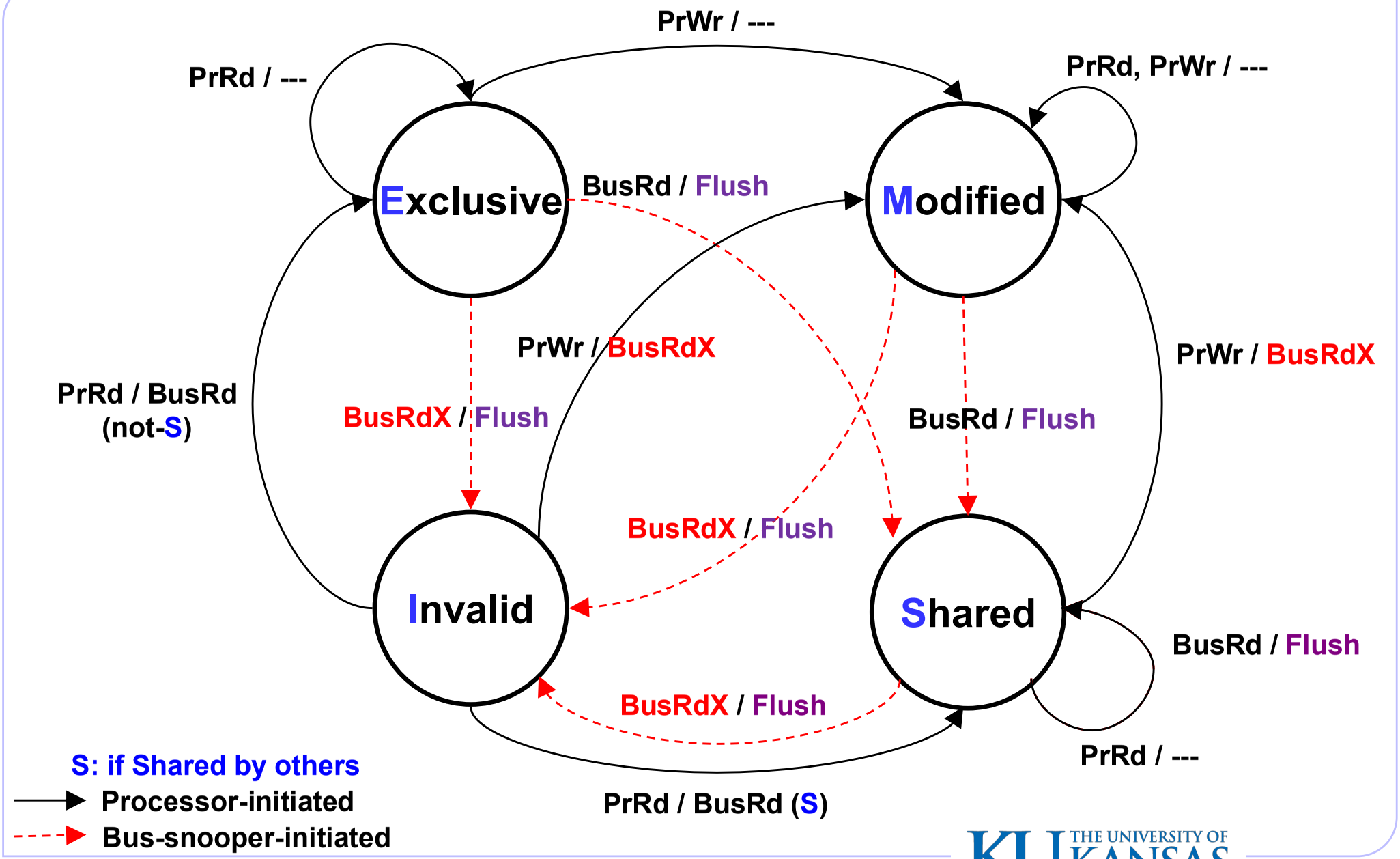


MESI Writeback Invalidation Protocol
Bus Transactions (Illinois Protocol)

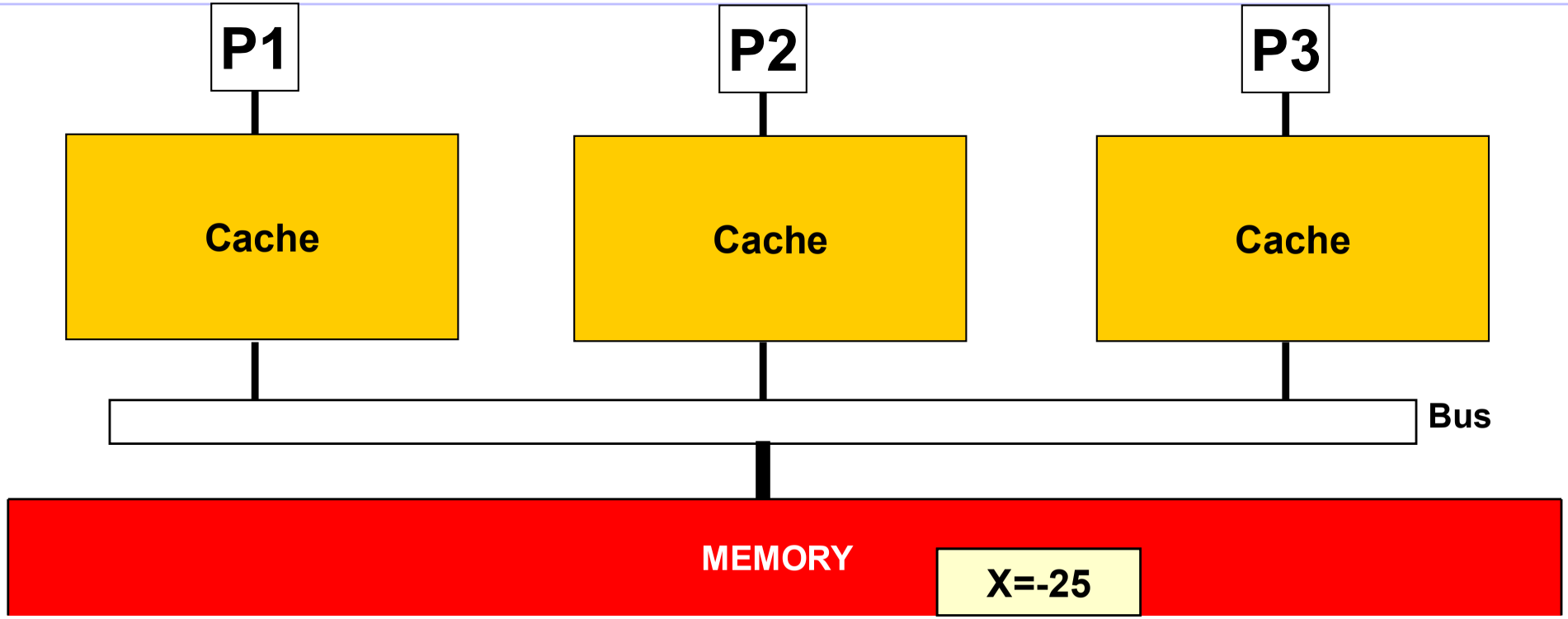
Whenever possible, MESI protocol performs $-to-$ transfer rather than having memory to supply the data. Use a Selection algorithm if there are multiple suppliers



MESI Writeback Invalidation Protocol (Illinois Protocol)

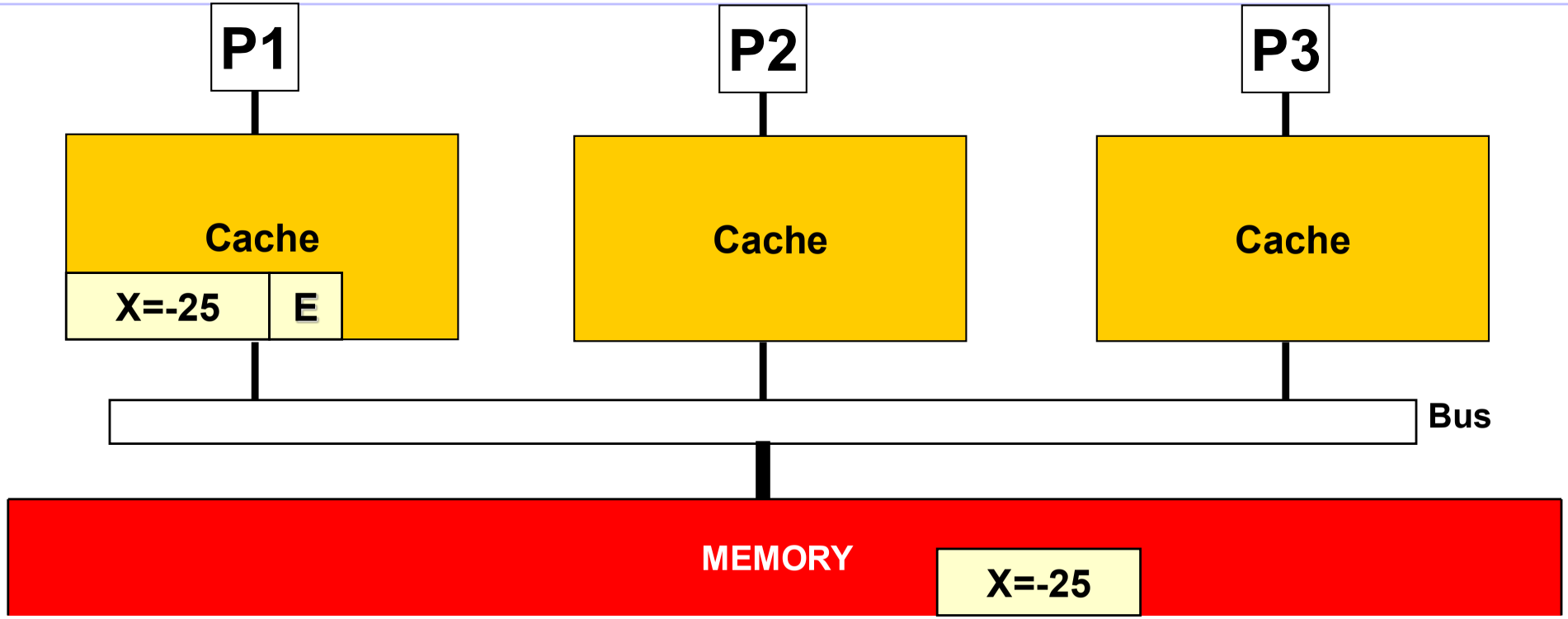


MESI Example



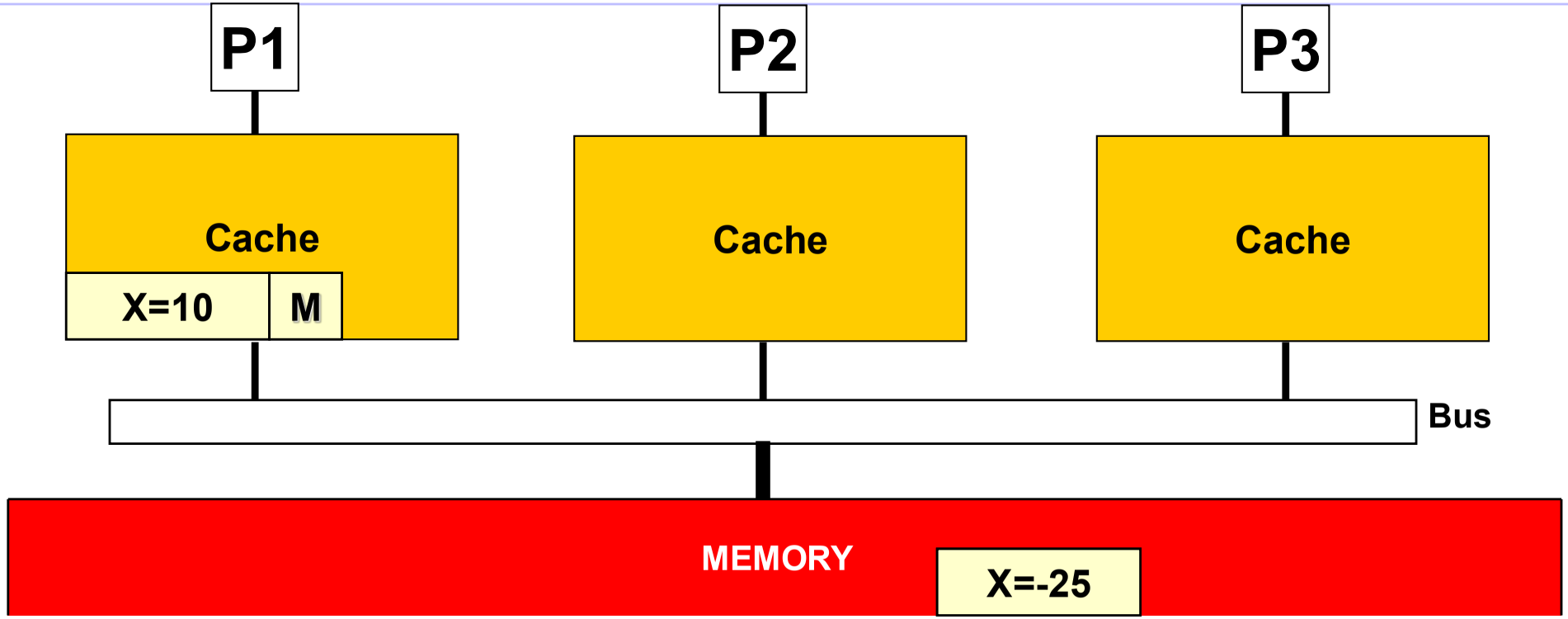
Processor Action	State in P1	State in P2	State in P3	Bus Transaction	Data Supplier
P1 reads X					
P1 writes X					
P3 writes X					
P1 reads X					
P2 reads X					

MESI Example



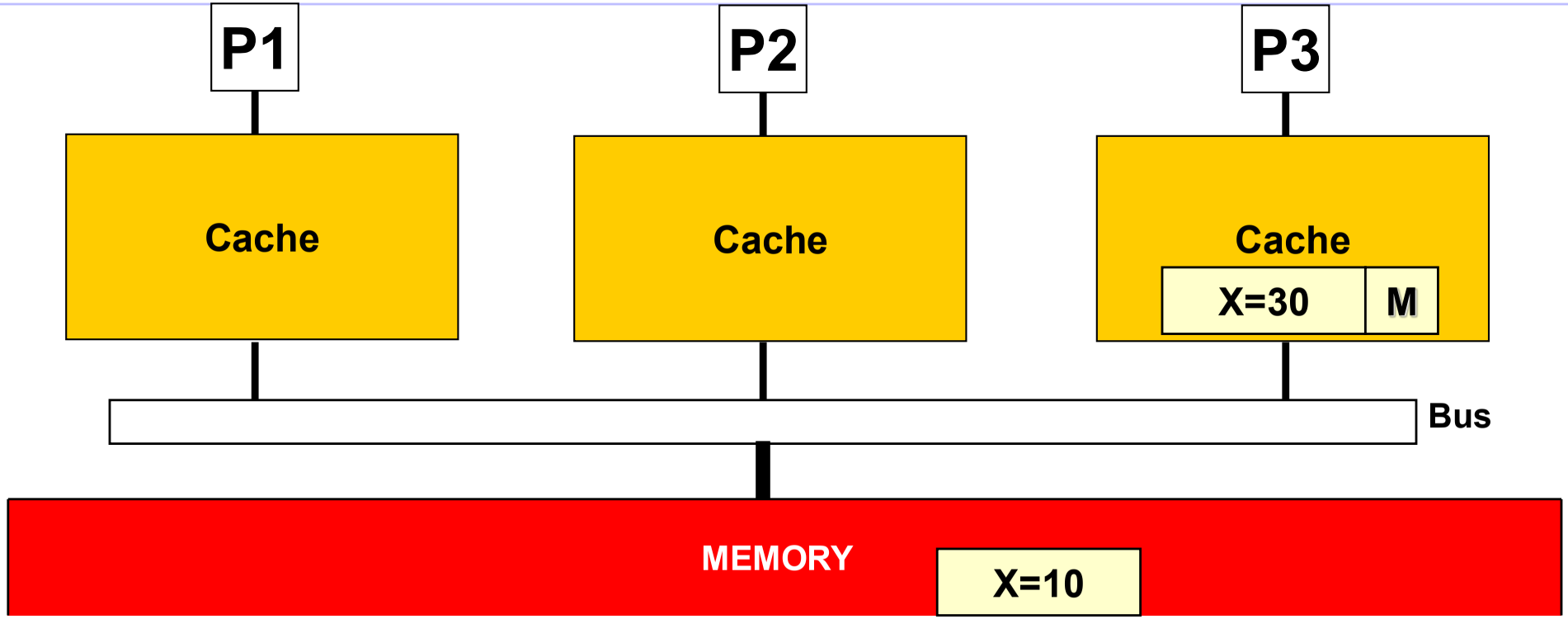
Processor Action	State in P1	State in P2	State in P3	Bus Transaction	Data Supplier
P1 reads X	E	I	I	BusRd	Memory
P1 writes X					
P3 writes X					
P1 reads X					
P2 reads X					

MESI Example



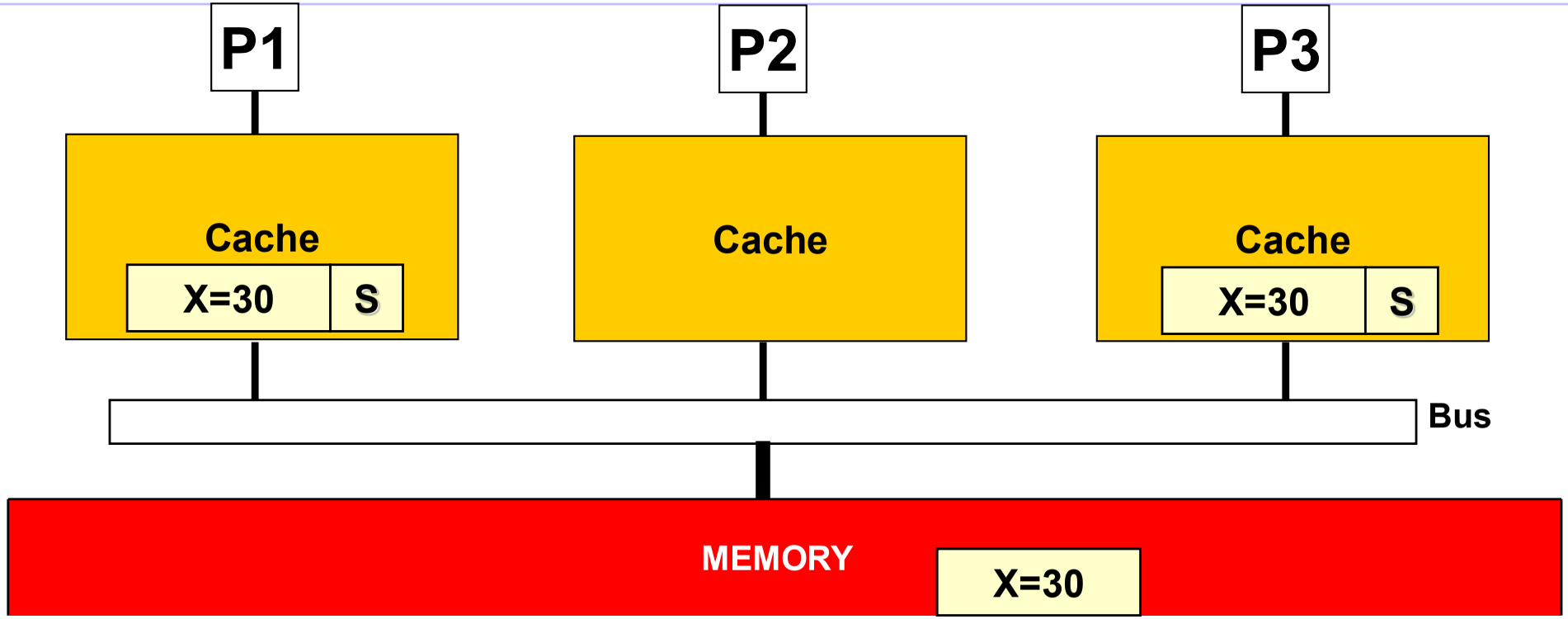
Processor Action	State in P1	State in P2	State in P3	Bus Transaction	Data Supplier
P1 reads X	E	I	I	BusRd	Memory
P1 writes X	M	I	I	---	---
P3 writes X					
P1 reads X					
P2 reads X					

MESI Example



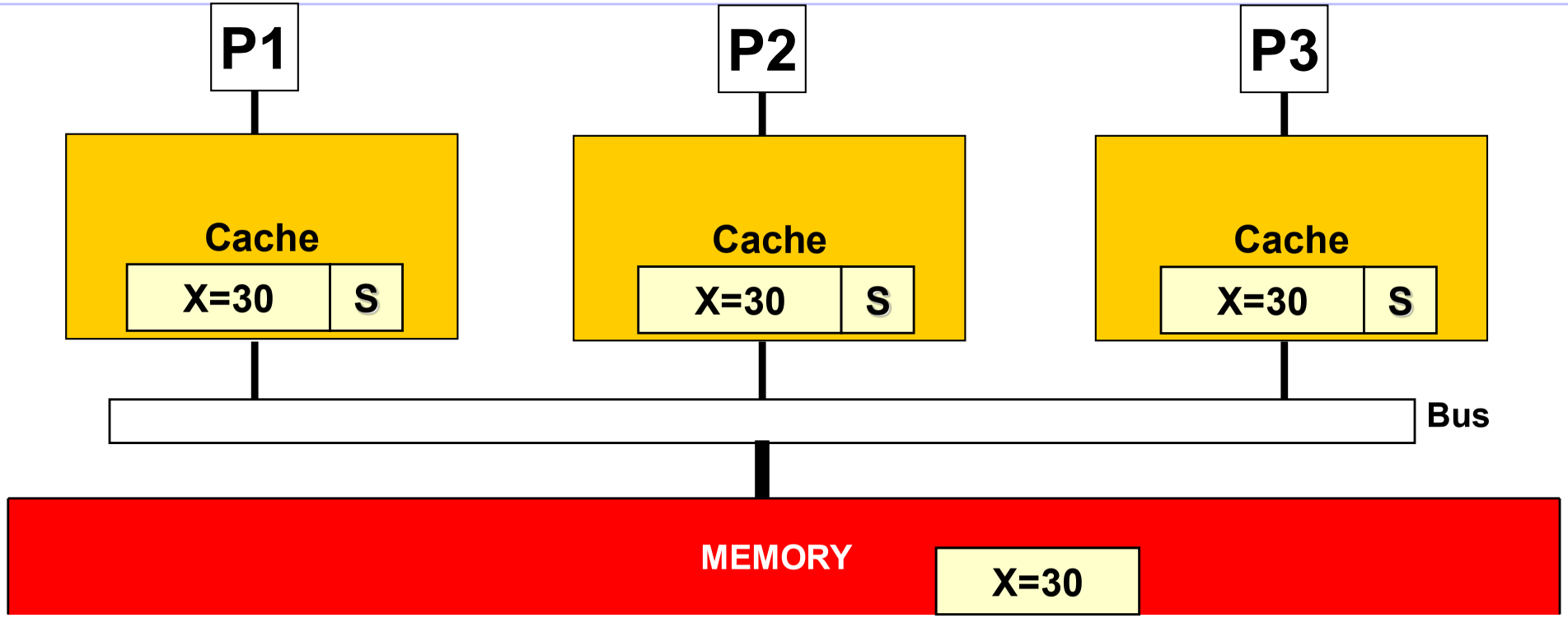
Processor Action	State in P1	State in P2	State in P3	Bus Transaction	Data Supplier
P1 reads X	E	I	I	BusRd	Memory
P1 writes X	M	I	I	---	---
P3 writes X	I	I	M	BusRdX	P1 Cache
P1 reads X					
P2 reads X					

MESI Example



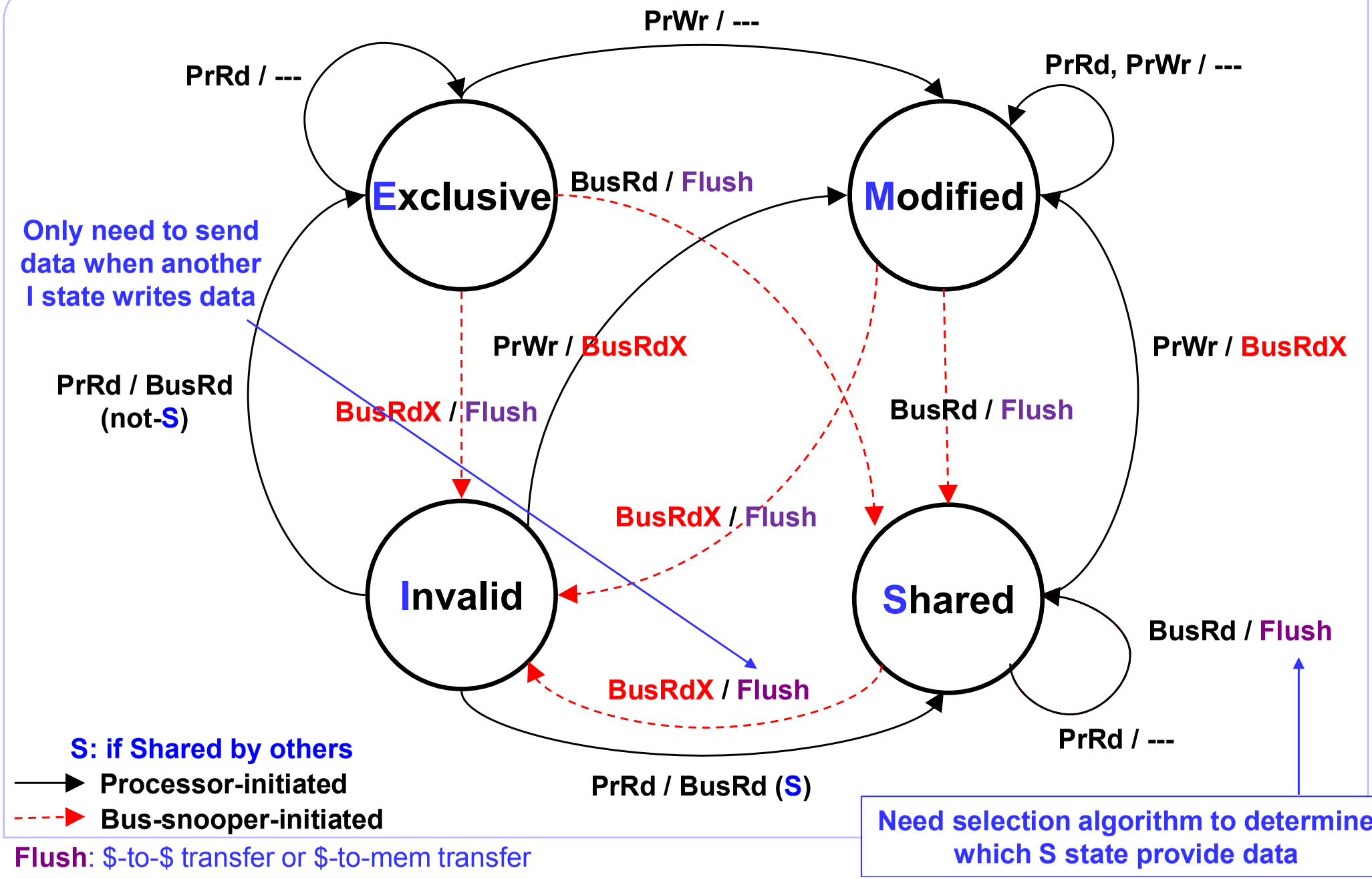
Processor Action	State in P1	State in P2	State in P3	Bus Transaction	Data Supplier
P1 reads X	E	I	I	BusRd	Memory
P1 writes X	M	I	I	---	---
P3 writes X	I	I	M	BusRdX	P1 Cache
P1 reads X	S	I	S	BusRd	P3 Cache
P2 reads X					

MESI Example



Processor Action	State in P1	State in P2	State in P3	Bus Transaction	Data Supplier
P1 reads X	E	I	I	BusRd	Memory
P1 writes X	M	I	I	---	---
P3 writes X	I	I	M	BusRdX	P1 Cache
P1 reads X	S	I	S	BusRd	P3 Cache
P2 reads X	S	S	S	BusRd	P1 or P3

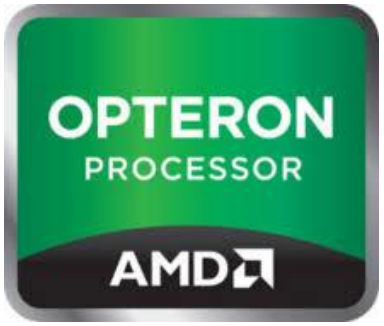
MESI Writeback Invalidation Protocol (Illinois Protocol)



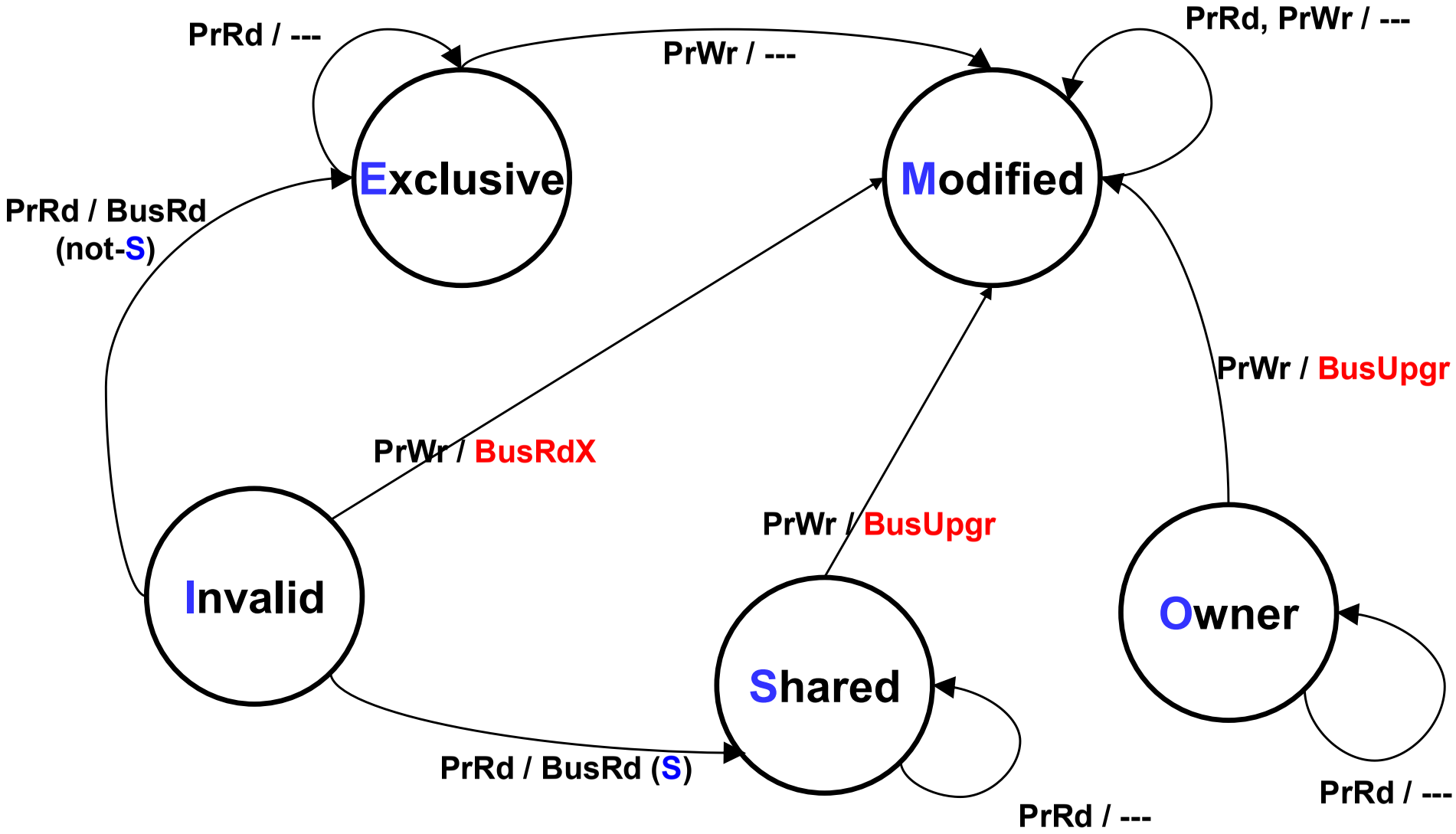
Flush: $-to-$ transfer or $-to-mem transfer

MOESI Protocol

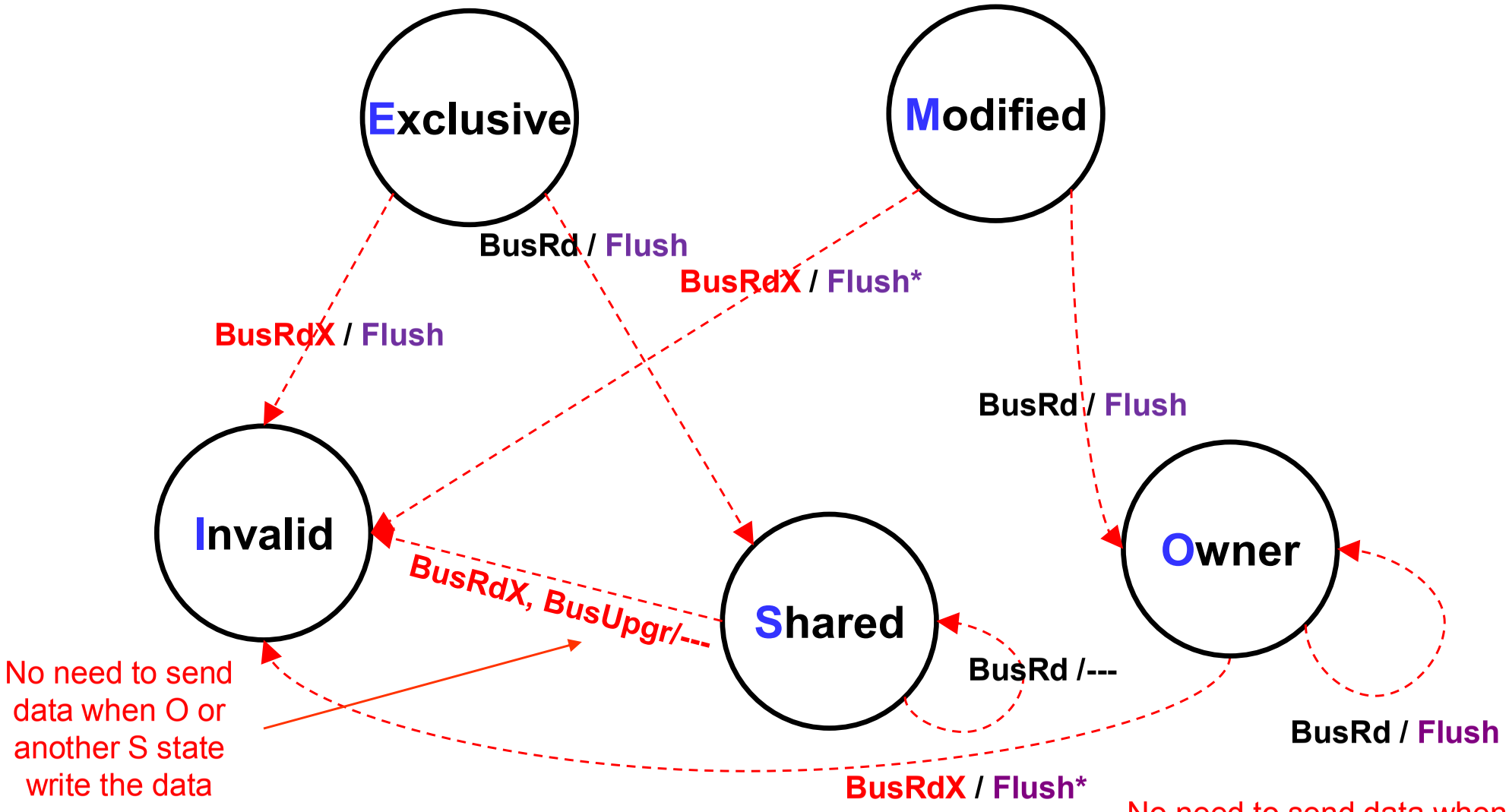
- Add one additional state — Owner state — to further save bus snooping signals
- Similar to Shared state, and S state in MOESI may be dirty, which is different from MESI
- The O state is dirty and will be responsible for supplying data and update memory
- Employed by
  - Sun UltraSparc
  - AMD Opteron



MOESI Writeback Invalidation Protocol

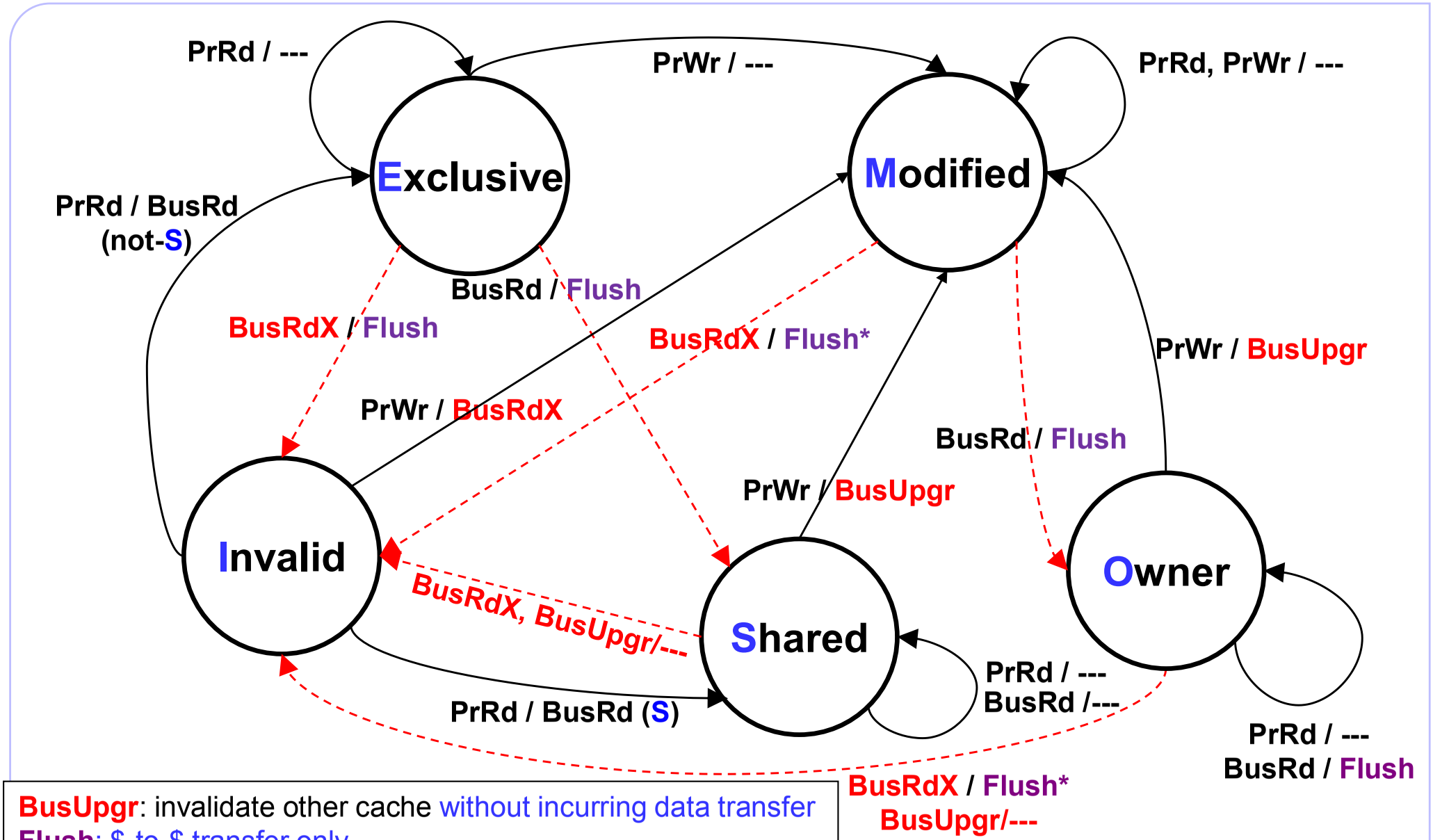


MOESI Writeback Invalidation Protocol



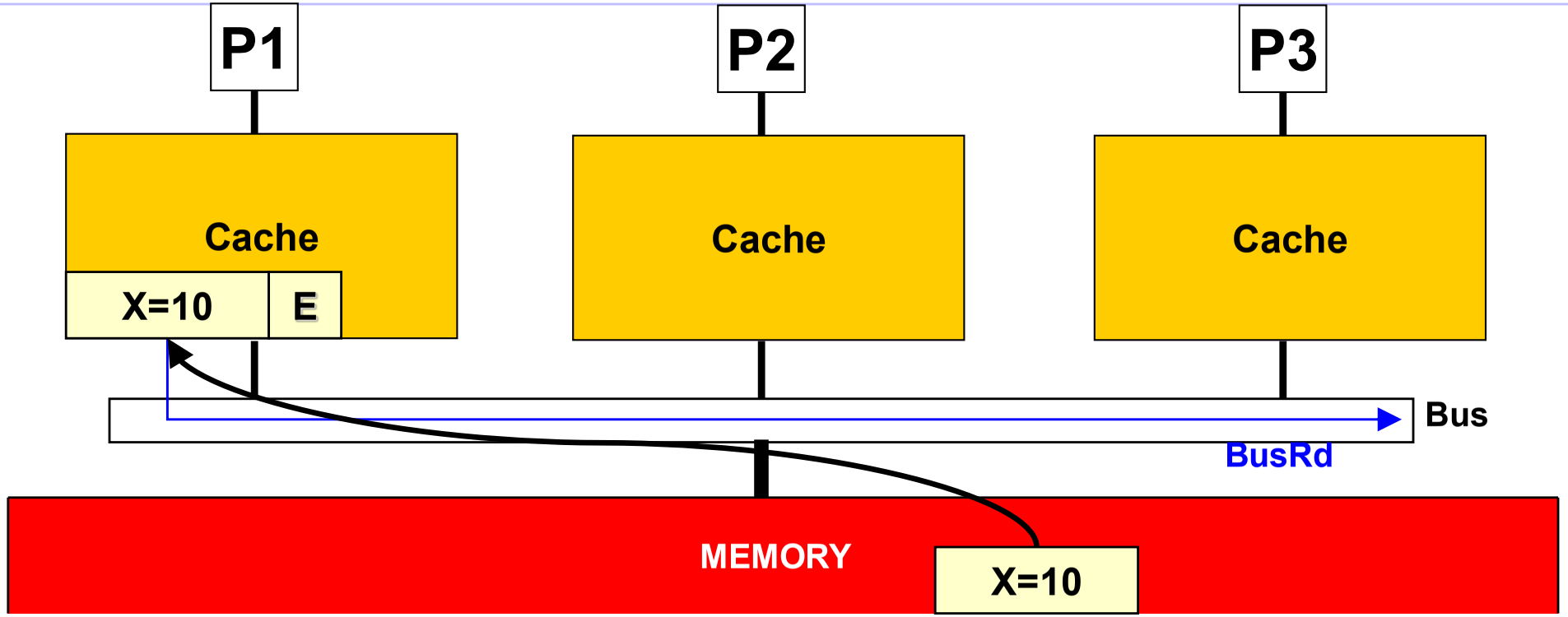
BusUpgr: invalidate other cache without incurring data transfer
Flush: $-to-$ transfer only
Flush*: $-to-$ transfer and $-to-mem transfer

MOESI Writeback Invalidation Protocol



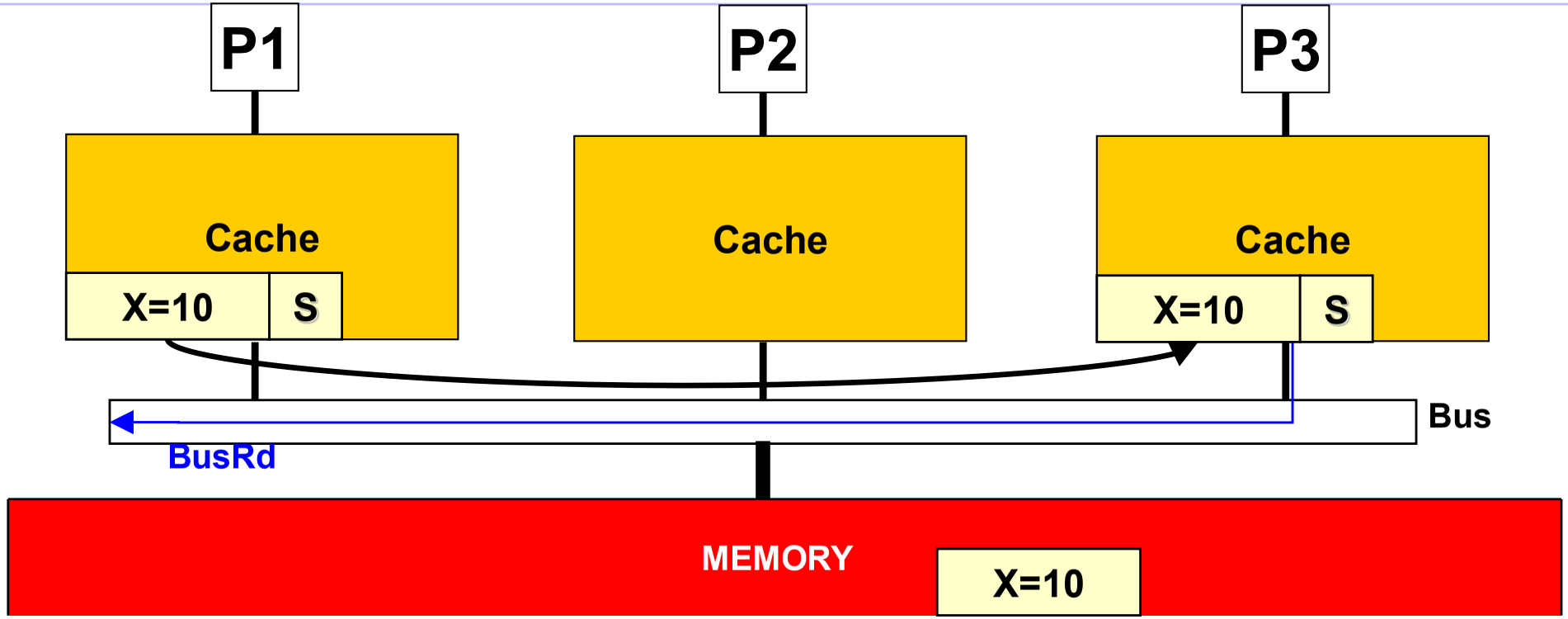
BusUpgr: invalidate other cache without incurring data transfer
Flush: $-to-$ transfer only
Flush*: $-to-$ transfer and $-to-mem transfer

MOESI Example



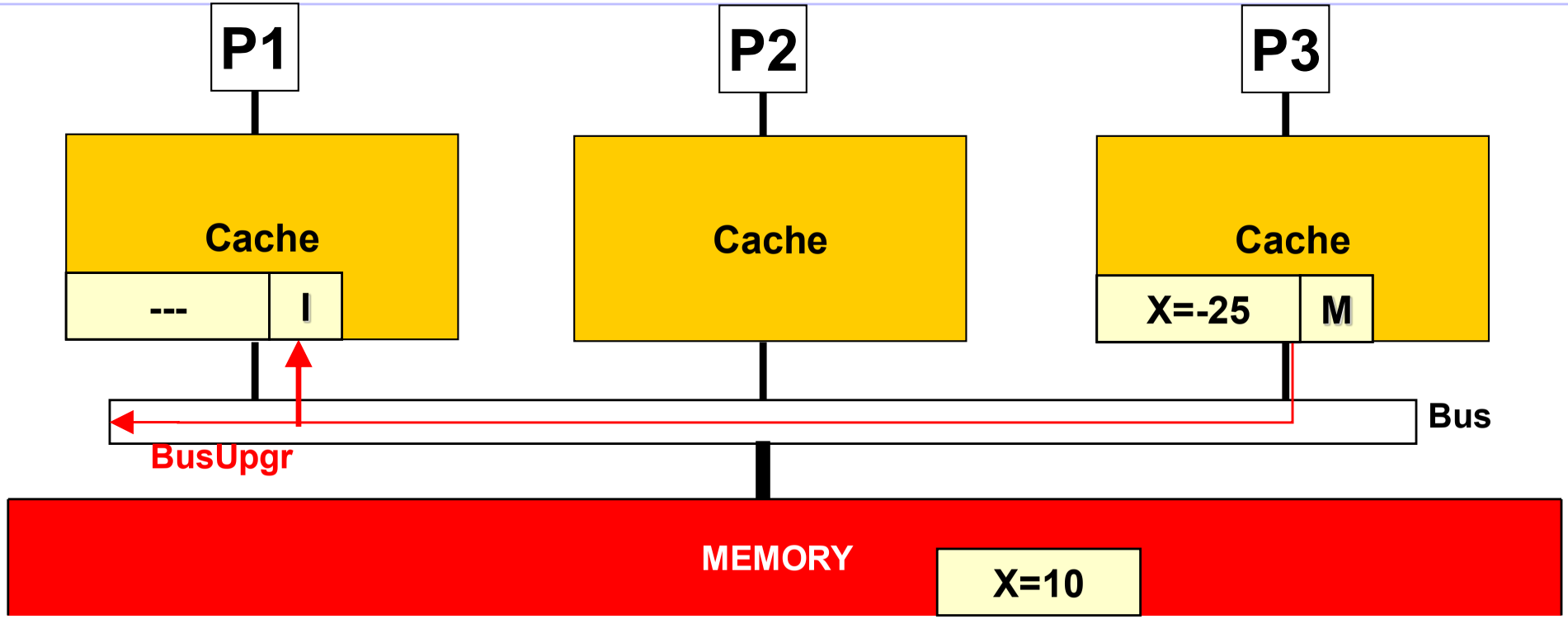
Processor Action	State in P1	State in P2	State in P3	Bus Transaction	Data Supplier
P1 reads X	E	I	I	BusRd	Memory

MOESI Example



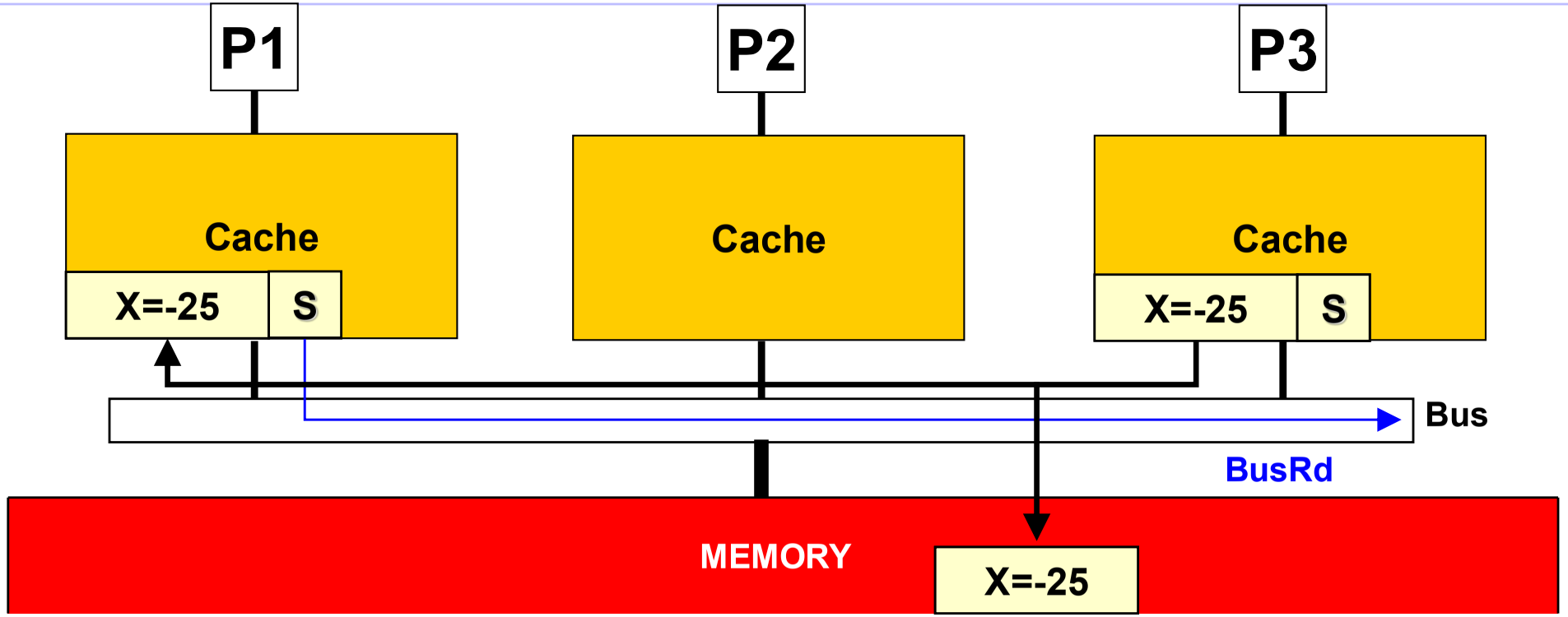
Processor Action	State in P1	State in P2	State in P3	Bus Transaction	Data Supplier
P1 reads X	E	I	I	BusRd	Memory
P3 reads X	S	I	S	BusRd	P1 Cache

MOESI Example



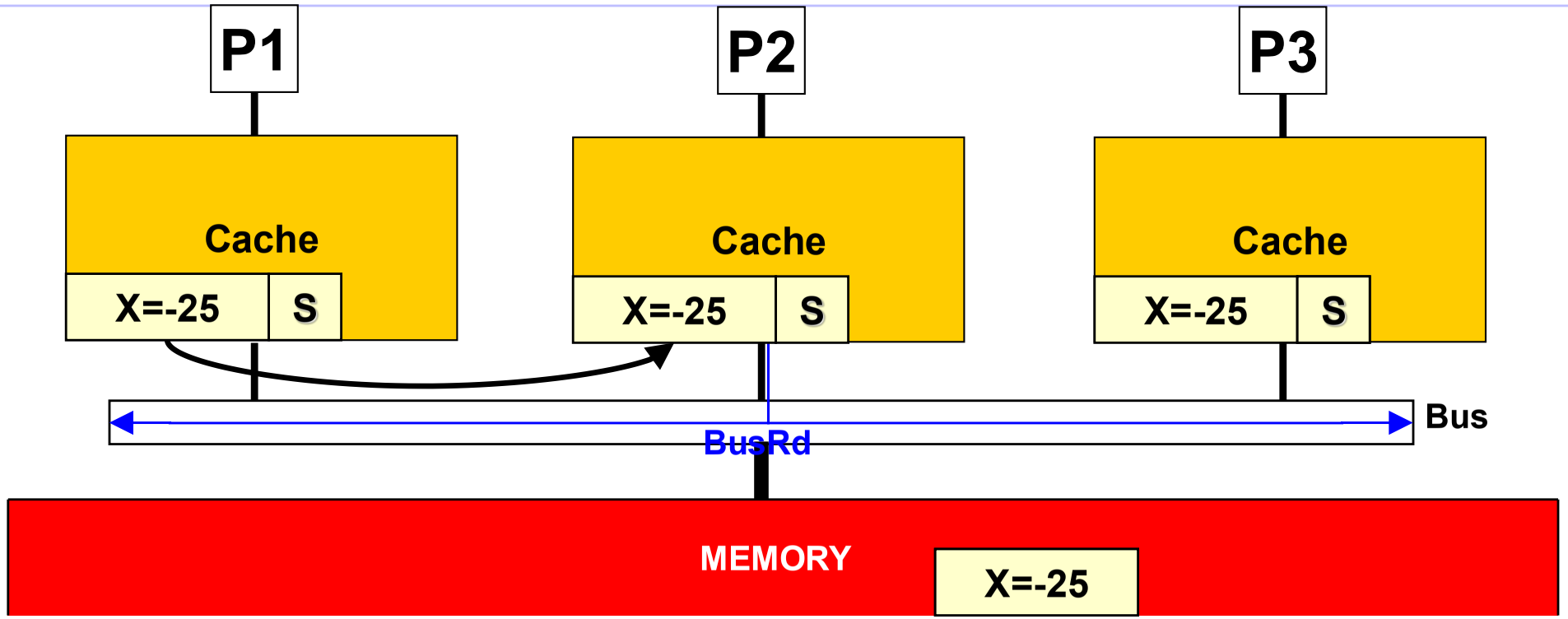
Processor Action	State in P1	State in P2	State in P3	Bus Transaction	Data Supplier
P1 reads X	E	I	I	BusRd	Memory
P3 reads X	S	I	S	BusRd	P1 Cache
P3 writes X	I	I	M	BusUpgr	---

MOESI Example



Processor Action	State in P1	State in P2	State in P3	Bus Transaction	Data Supplier
P1 reads X	E	I	I	BusRd	Memory
P3 reads X	S	I	S	BusRd	P1 Cache
P3 writes X	I	I	M	BusUpgr	---
P1 reads X	S	I	O	BusRd	P3 Cache

MOESI Example



Processor Action	State in P1	State in P2	State in P3	Bus Transaction	Data Supplier
P1 reads X	E	I	I	BusRd	Memory
P3 reads X	S	I	S	BusRd	P1 Cache
P3 writes X	I	I	M	BusUpgr	---
P1 reads X	S	I	O	BusRd	P3 Cache
P2 reads X	S	S	O	BusRd	P3 Cache

How to avoid unnecessary bus snooping?