SHARED MEMORY SYSTEMS

Mahdi Nazm Bojnordi

Assistant Professor

School of Computing

University of Utah

Overview

- Shared memory systems
  - Inconsistent vs. consistent data
- Cache coherence with write back policy
  - MSI protocol
  - MESI protocol
- Memory consistency
  - Sequential consistency

Recall: Shared Memory Systems

- Multiple threads employ a shared memory system
  - Easy for programmers
- Complex synchronization mechanisms are required
  - Cache coherence
    - All the processors see the same data for a particular memory address as they should have if there were no caches in the system
    - e.g., snoopy protocol with write-through, write no-allocate
      - Inefficient
  - Memory consistency
    - All memory instructions appear to execute in the program order
    - e.g., sequential consistency

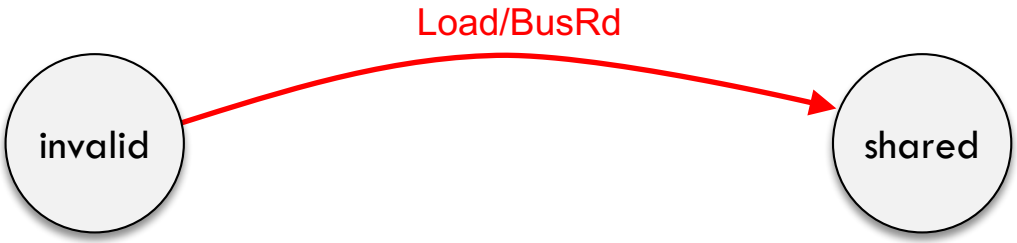
Snooping with Writeback Policy

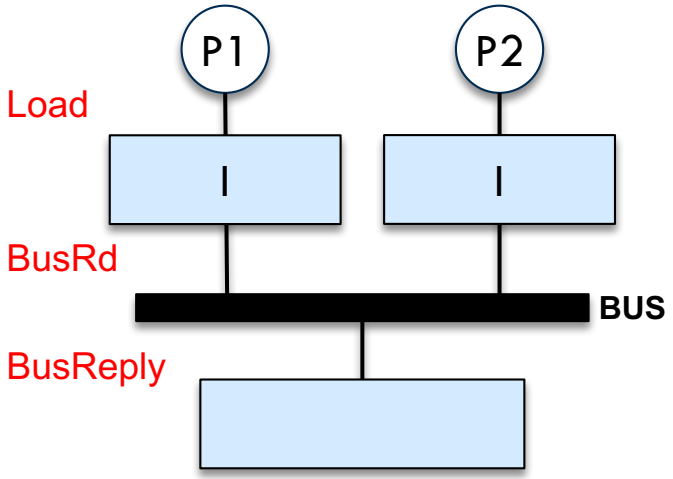
- Problem: writes are not propagated to memory until eviction
  - Cache data maybe different from main memory
- Solution: identify the owner of the most recently updated replica
  - Every data may have only one owner at any time
  - Only the owner can update the replica
  - Multiple readers can share the data
    - No one can write without gaining ownership first

Modified-Shared-Invalid Protocol

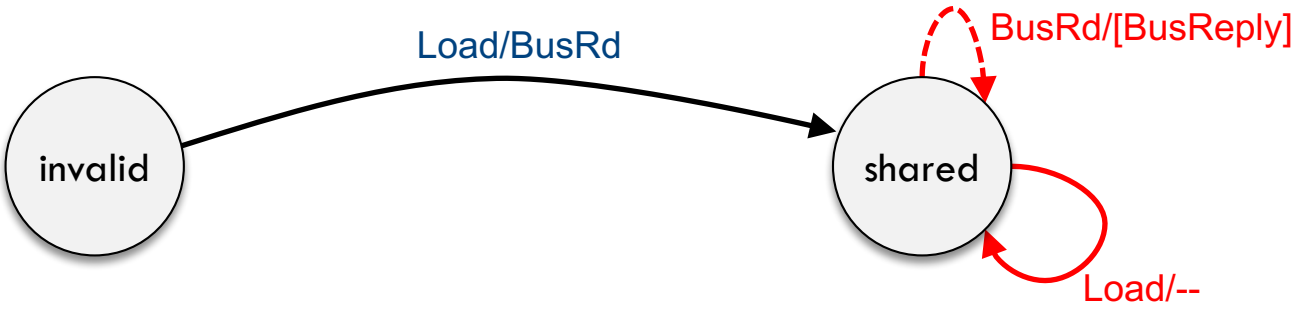
- Every cache block transitions among three states
  - Invalid: no replica in the cache
  - Shared: a read-only copy in the cache
    - Multiple units may have the same copy
  - Modified: a writable copy of the data in the cache
    - The replica has been updated
    - The cache has the only valid copy of the data block
- Processor actions
  - Load, store, evict
- Bus messages
  - BusRd, BusRdX, BusInv, BusWB, BusReply

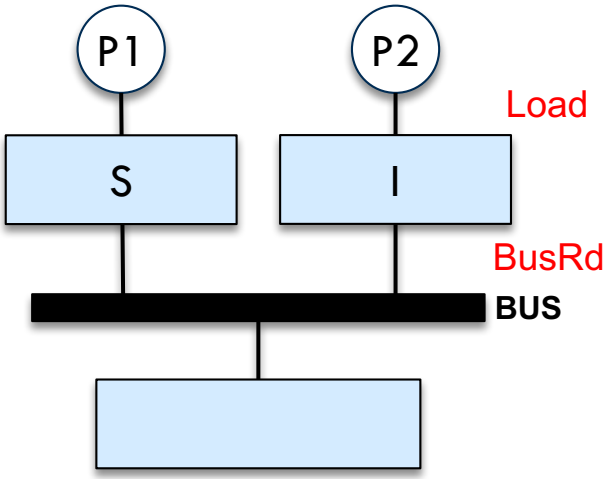
MSI Example



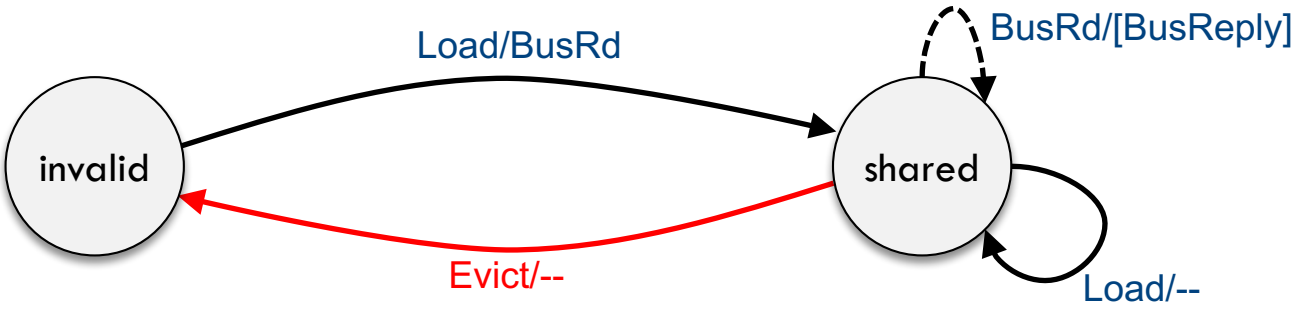


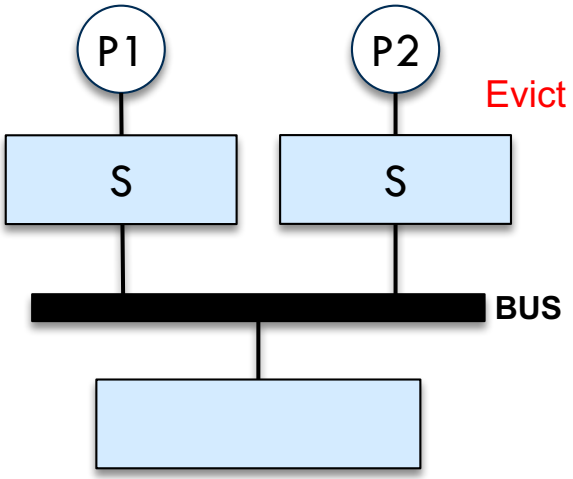
MSI Example


Load/BusRd
BusRd/[BusReply]
invalid
shared
Load/--

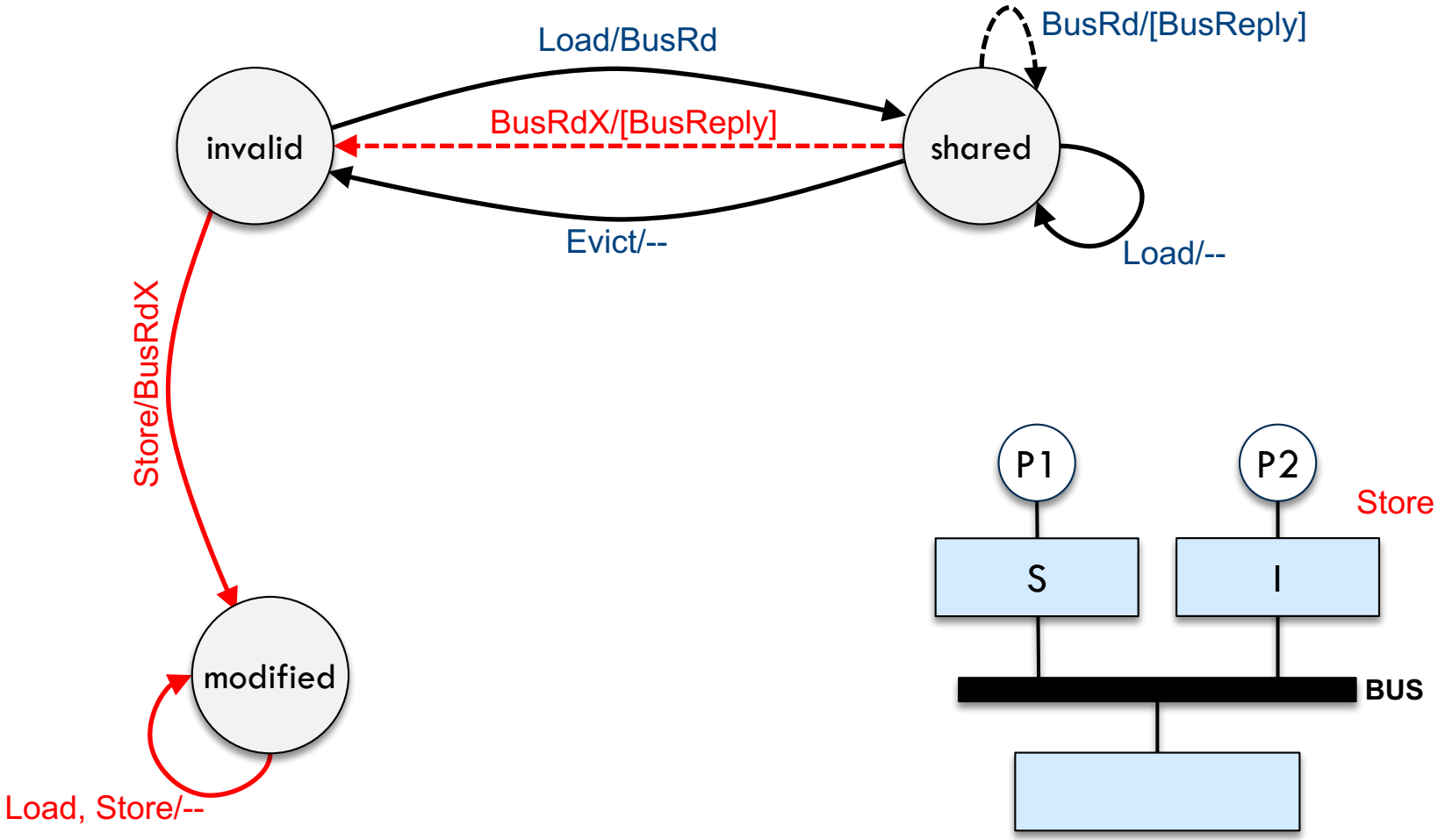

P1
P2
Load
S
I
BusRd
BUS

MSI Example

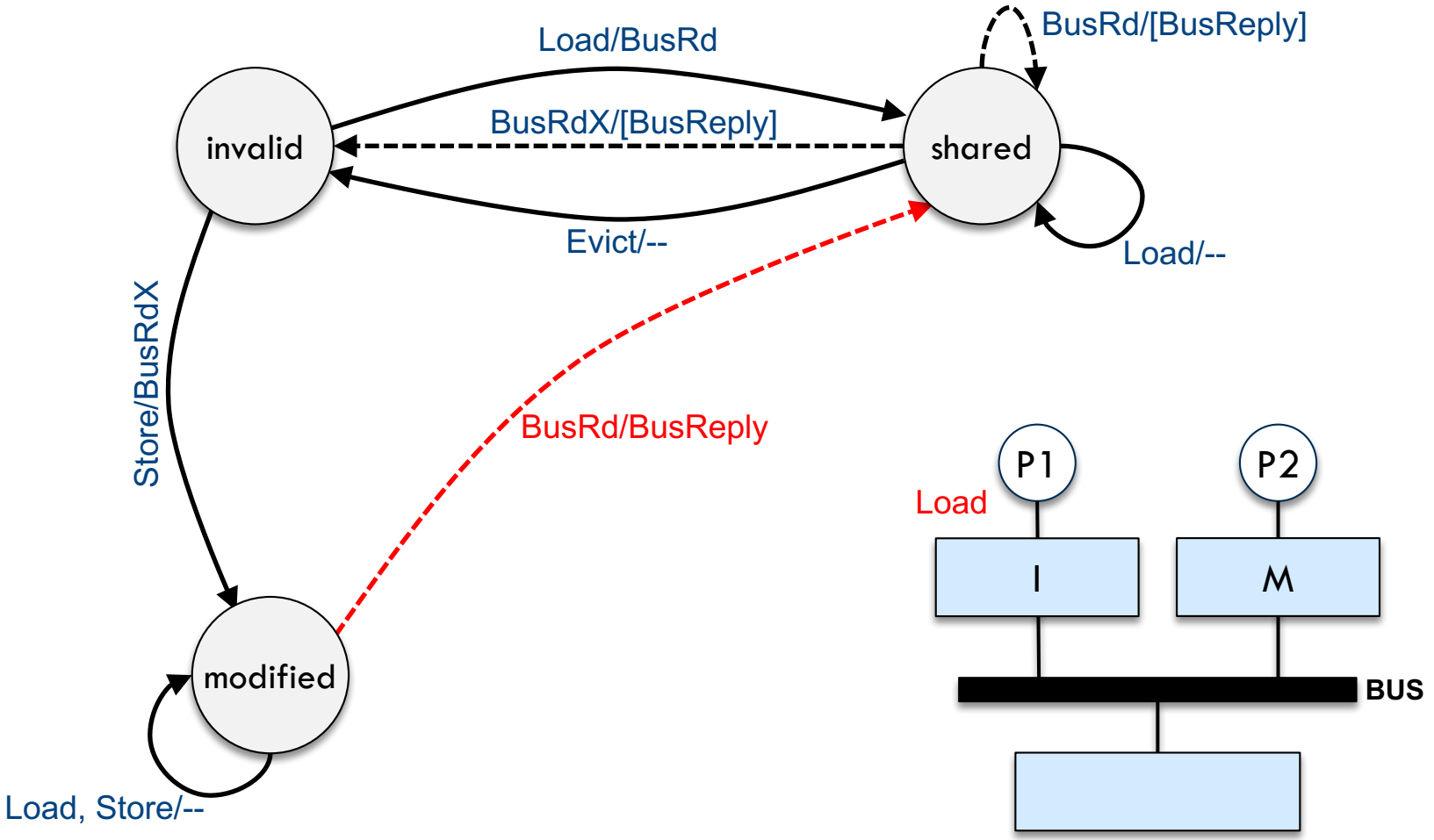

Load/BusRd
BusRd/[BusReply]
invalid
shared
Evict/--
Load/--


P1
P2
Evict
S
S
BUS

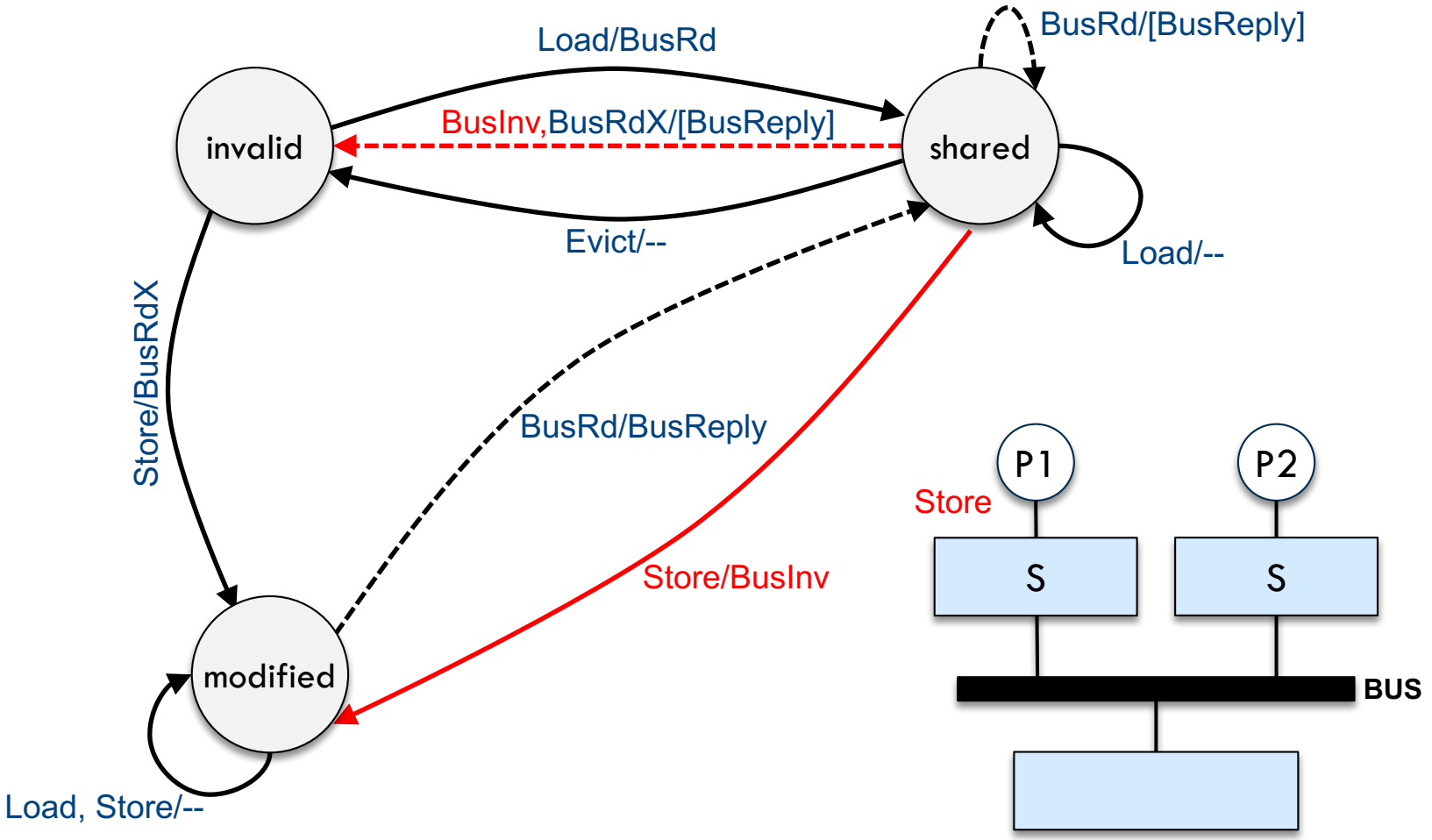
MSI Example


Load/BusRd
BusRd/[BusReply]
invalid
BusRdX/[BusReply]
shared
Evict/--
Load/--
Store/BusRdX
modified
Load, Store/--
P1
P2
Store
S
I
BUS

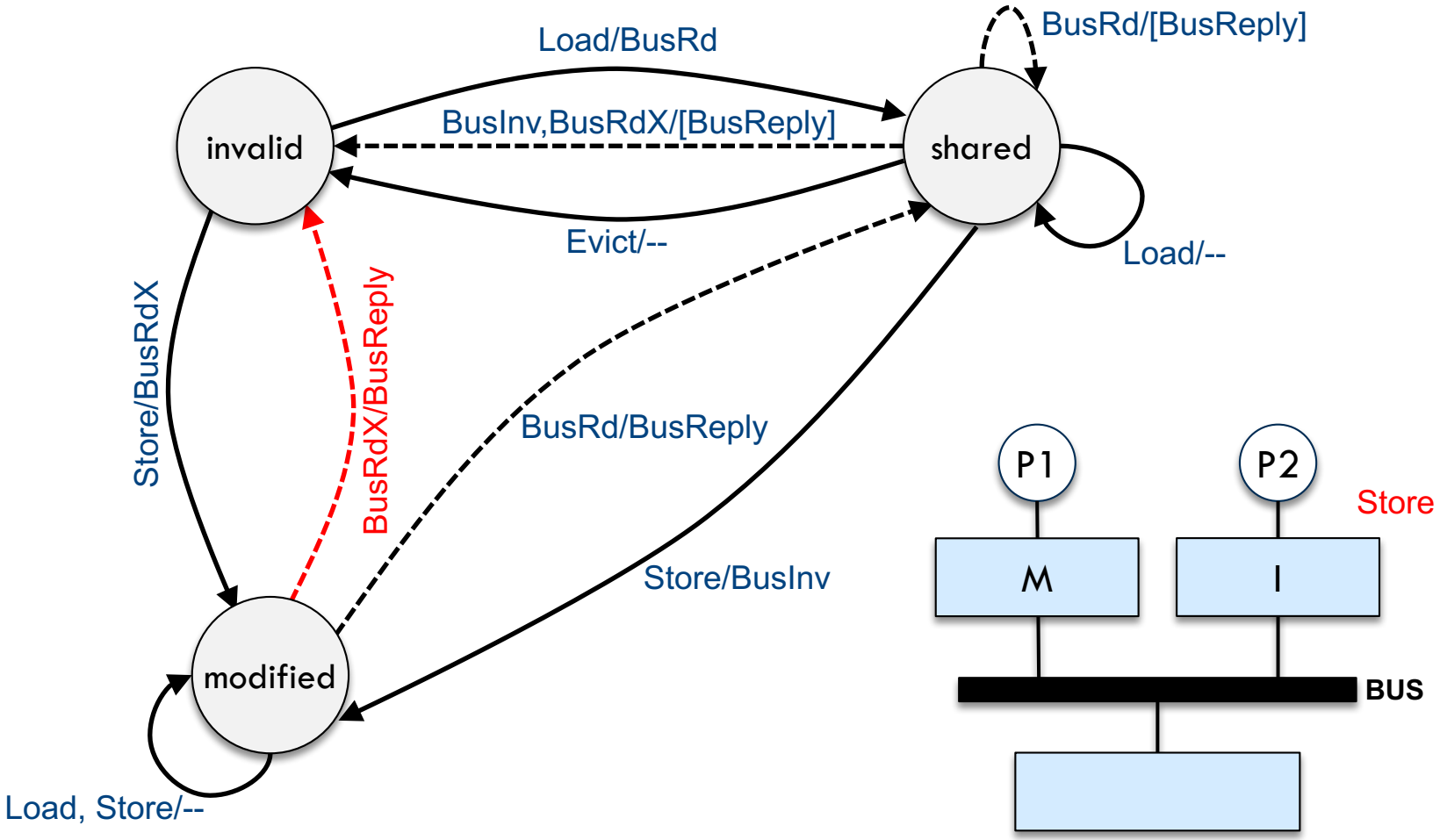
MSI Example


Load/BusRd
BusRd/[BusReply]
invalid
BusRdX/[BusReply]
shared
Evict/--
Load/--
Store/BusRdX
BusRd/BusReply
modified
Load, Store/--
P1
P2
Load
I
M
BUS

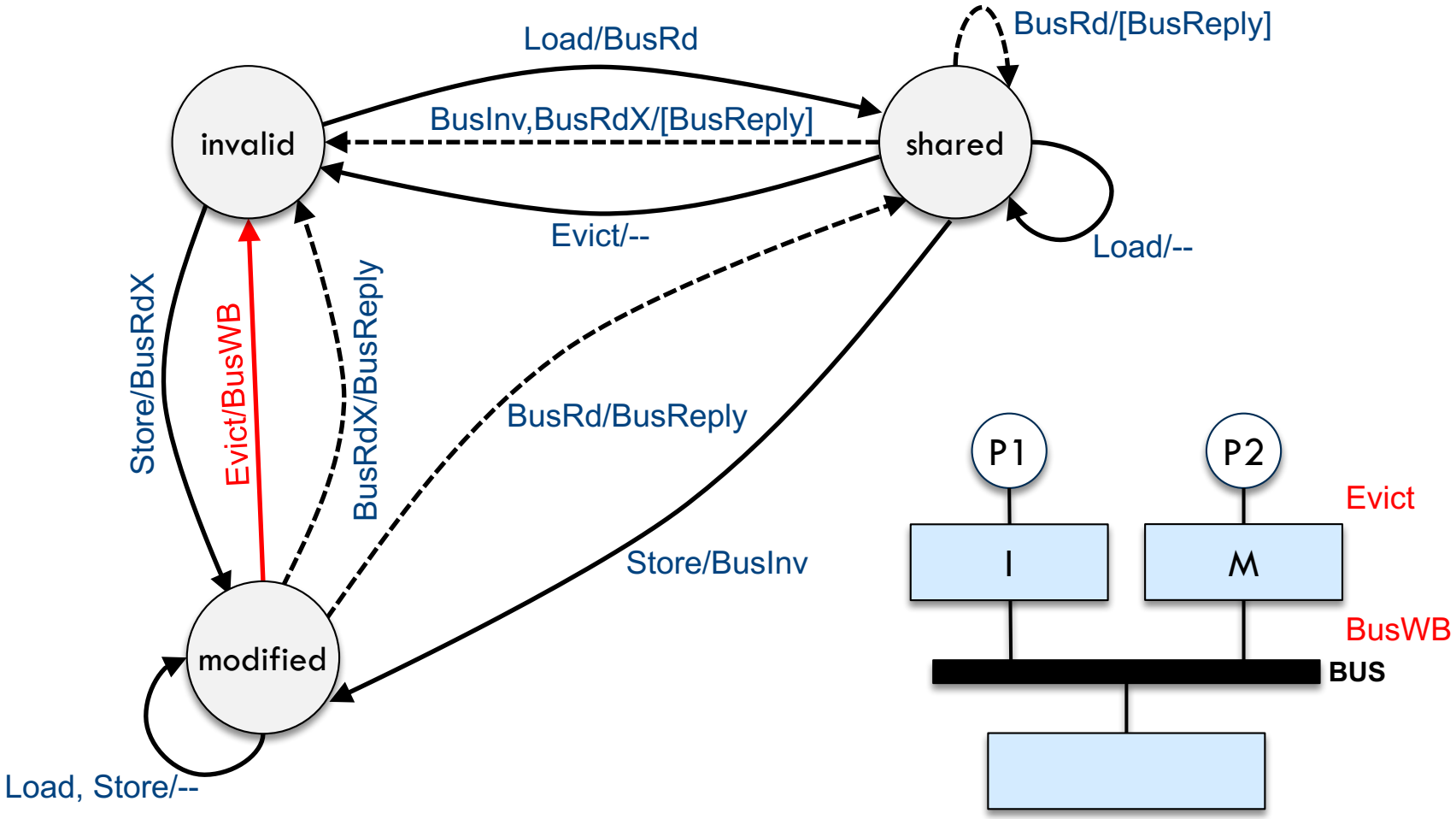
MSI Example


Load/BusRd
BusRd/[BusReply]
invalid
BusInv,BusRdX/[BusReply]
shared
Evict/--
Load/--
Store/BusRdX
BusRd/BusReply
Store/BusInv
modified
Load, Store/--
P1
P2
Store
S
S
BUS

MSI Example


Load/BusRd
BusRd/[BusReply]
invalid
BusInv,BusRdX/[BusReply]
shared
Evict/--
Load/--
Store/BusRdX
BusRdX/BusReply
BusRd/BusReply
Store/BusInv
modified
Load, Store/--
P1
P2
Store
M
I
BUS

MSI Example


Load/BusRd
BusRd/[BusReply]
BusInv,BusRdX/[BusReply]
invalid
shared
Evict/--
Load/--
Store/BusRdX
Evict/BusWB
BusRdX/BusReply
BusRd/BusReply
Store/BusInv
modified
Load, Store/--
P1
P2
Evict
I
M
BusWB
BUS

Modified, Exclusive, Shared, Invalid

- Also known as Illinois protocol
  - Employed by real processors
  - A cache may have an exclusive copy of the data
  - The exclusive copy may be copied between caches
- Pros
  - No invalidation traffic on write-hits in the E state
  - Lower overheads in sequential applications
- Cons
  - More complex protocol
  - Longer memory latency due to the protocol