Directory-Based Cache Coherence

Scalable Shared Memory System

- **Bus-based MP: good for small multiprocessors, but does not scale**
  - **Physical constraints**
    - long wires (low clock frequency), arbitration delay
  - **Protocol constraints**
    - **Snoopy/broadcast**: bandwidth getting saturated quickly
  - **Contention everywhere**:
    - bus, snooper, memory

- **How to scale to larger MP?**

Scalable Shared Memory System

		Interconnection	
		Bus	Point-to-Point
Protocol	Snoopy	Least scalable	More scalable
	Directory	/	Most scalable

Ways to Scale Bus-Based MP

- **Approach 1: replace bus with point-to-point interconnection, but keep relying on snooping/broadcast**
  - e.g., AMD Opteron (mesh), IBM Cell (ring), Intel Larrabee (ring)
  - Still cannot scale to hundreds of processors

Ways to Scale Bus-Based MP

- **Approach 2: replace broadcast with directory protocol**
  - e.g. SGI Altix system
  - Leads to distributed shared memory (DSM) multiprocessors
  - Scaling to tens to hundreds of processors

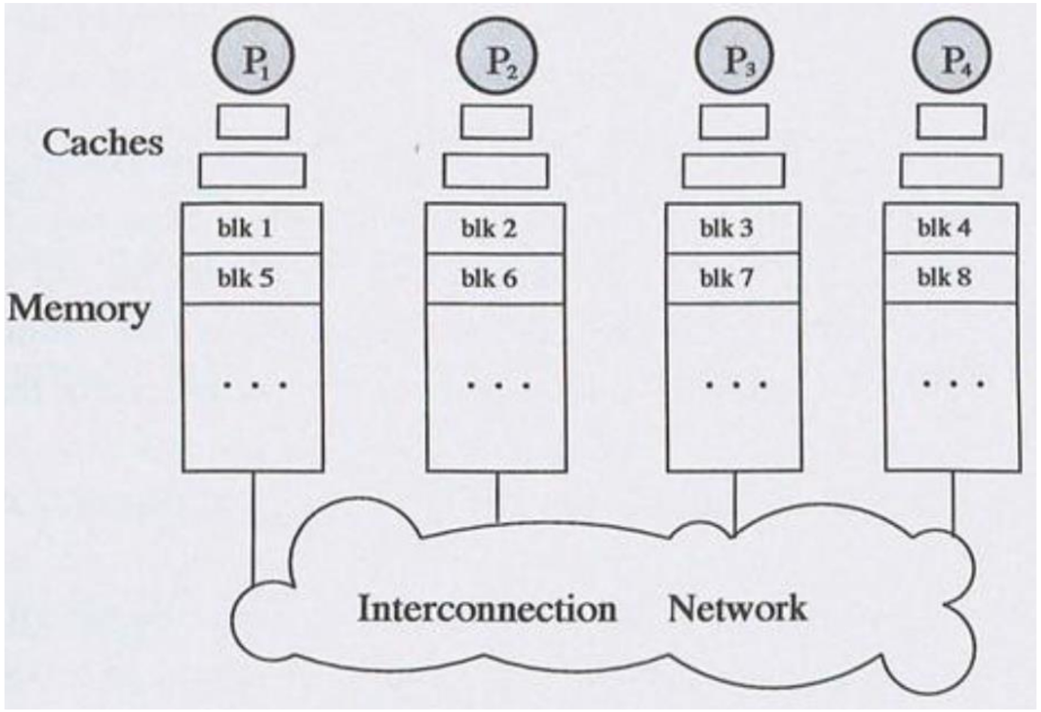
Directory Protocols

Questions to be considered

- **Questions:**
  - Which memory a block must be placed in
  - Where the directory of a particular block can be found
  - How the directory information is organized
  - How to handle races in the protocol
  - How to ensure write propagation and write serialization in a DSM
  - …….

Question 1

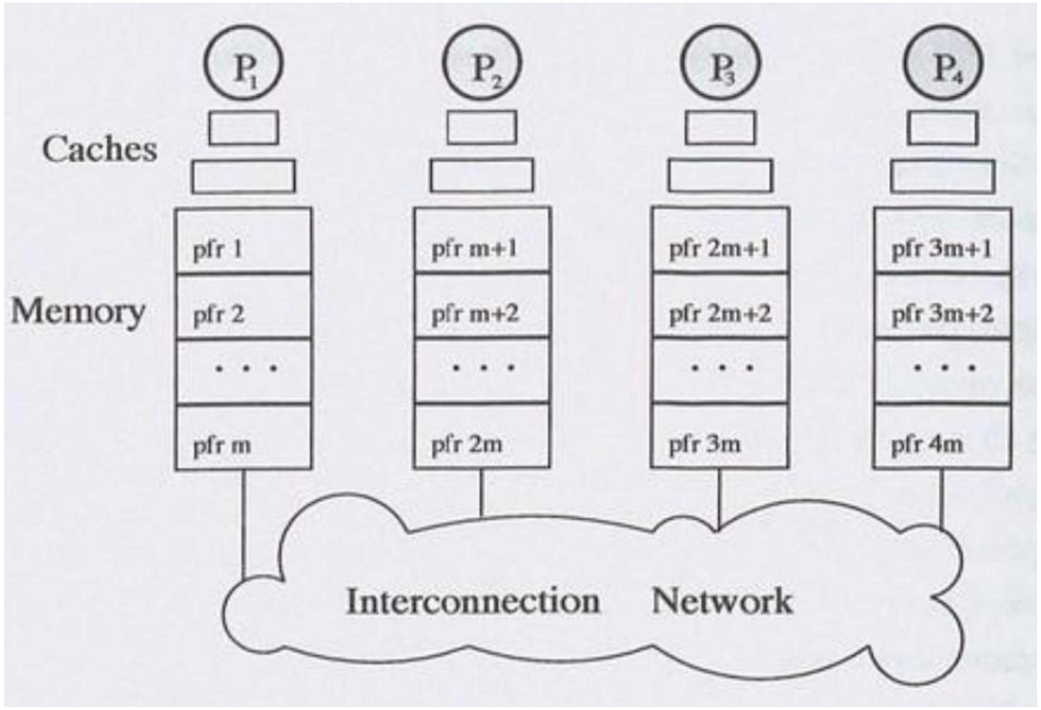
- **Q1: which memory to place a particular block?**
- **One solution: block interleaving**



- Works against spatial locality of accesses
- In most DSMs, block interleaving is not used.

Question 1

- **Another solution: mapping without interleaving**
  - Consecutive physical page frames (pfr) are mapped to a memory.



- OS must be involved in deciding where to allocate
  - First touch
    - Processor that first accesses it.
  - round robin

Finding a Block

- **A block can only map to a single memory**
  - This memory is referred to as the "**home**" of the block

- **The home memory keeps track of which caches may have the copy of a block**
  - in a structure called the "**directory**"

Question 2

- **Q2: where is directory stored?**
- **Solution:**
  - **Cache-based directory**
    - Kept as a doubly linked list among multiple caches
    - Maintaining the list is *too complicated!*
  - **Memory-based directory**
    - Keep directory in the main memory.
    - Mainstream approach

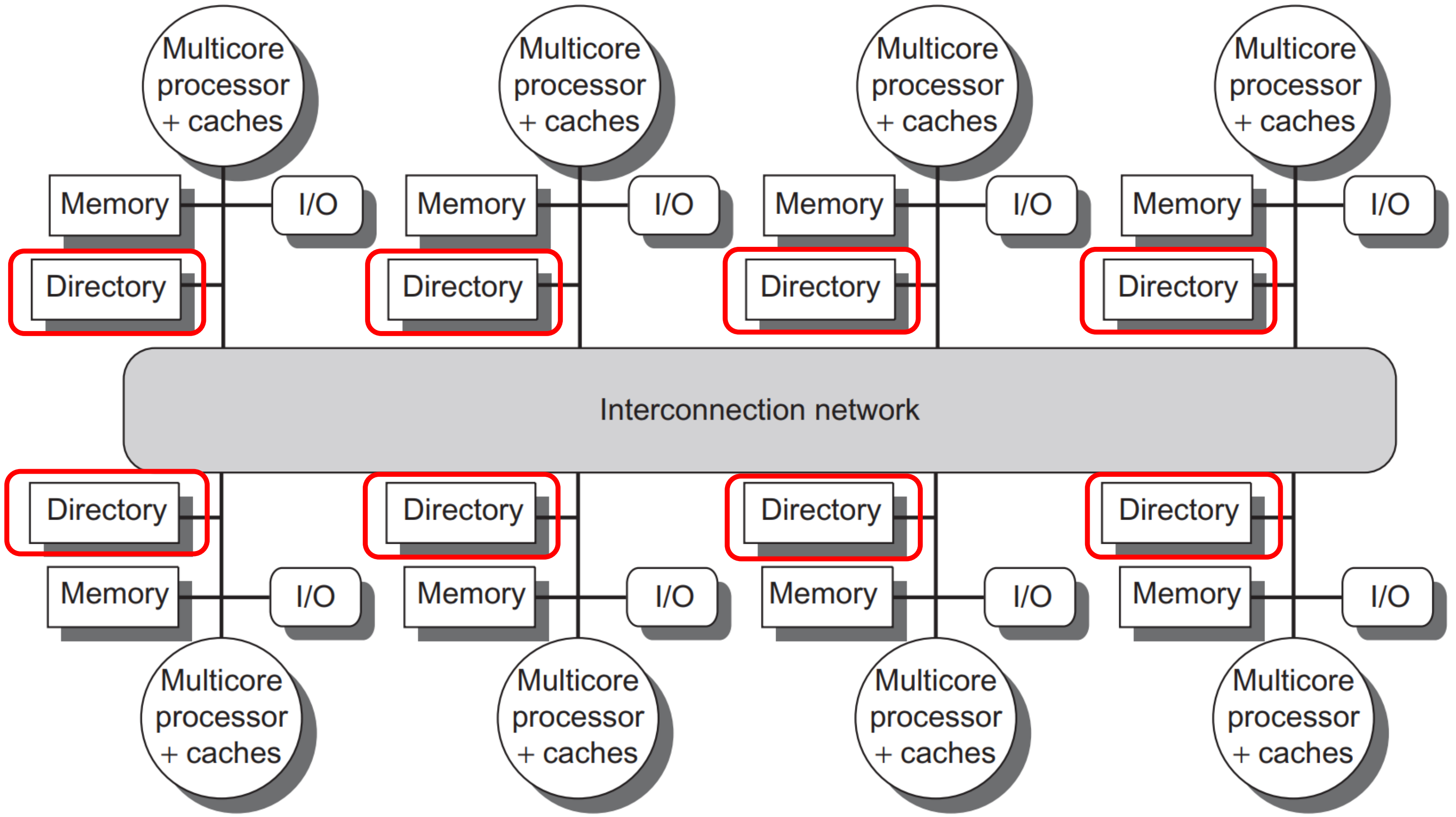
Centralized Directory

- Use a **centralized directory** to keep track of every block
  - Which caches have each block
  - Dirty status of each block

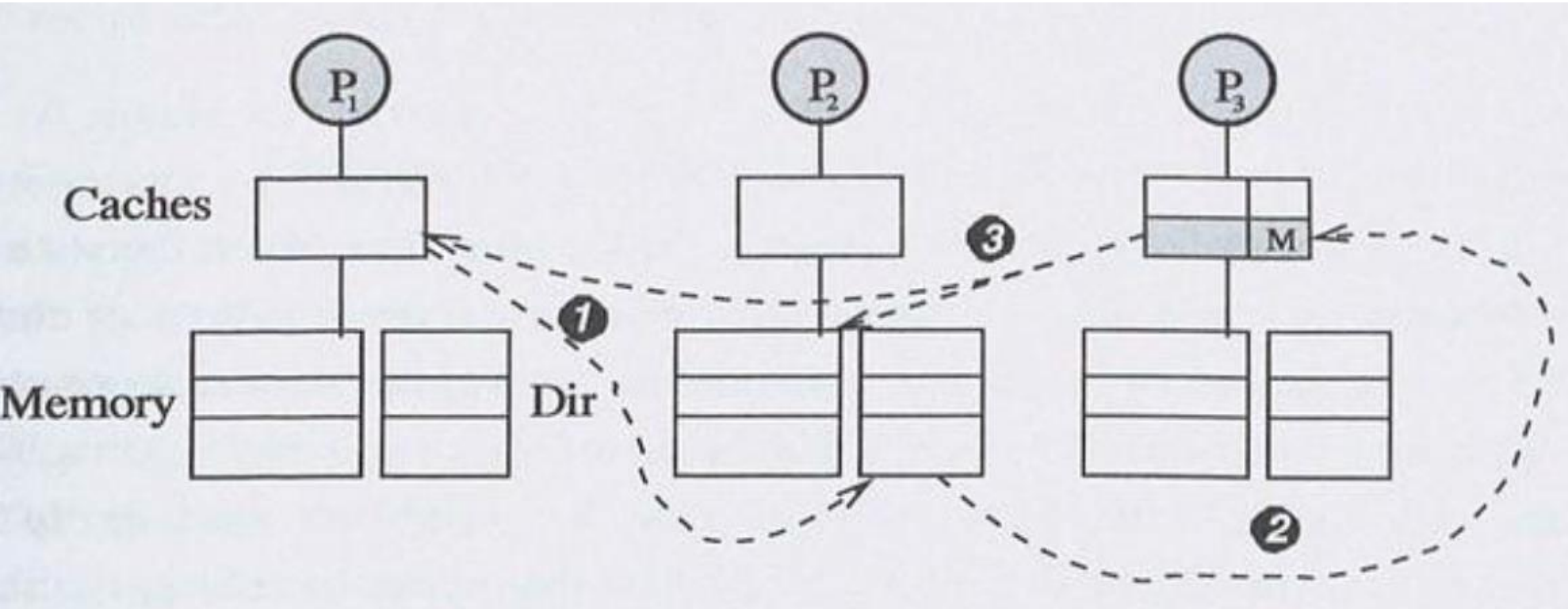
- ***Not scalable***
  - The directory receives requests from all processors
  - Its central location is far away from most processors

Distributed Directory

- **The directory is scattered at different nodes.**
  - Keep directory info of a block in home node

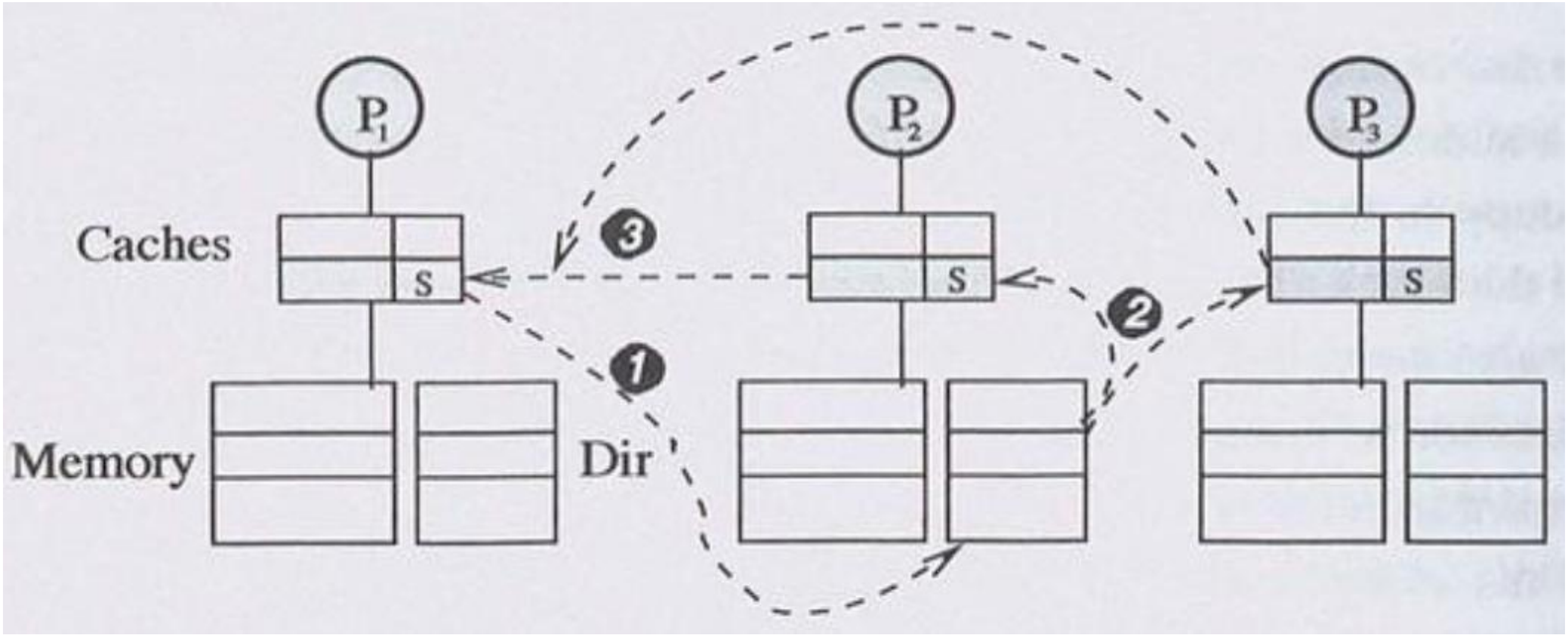


Basic Directory Handling of Load Request



1. Requestor (P1) sends a Read request to the Home (P2) of the block
2. The home looks up the directory to find out who has the block and in what state. Then it sends an intervention request to the current owner (P3)
3. The owner (P3) supplies the block to the requestor (P1) and may update the home as well

Basic Directory Handling of a Write Req



1. Requestor (P1) sends a Write request to Home (P2)
2. Home finds out that P2 and P3 are currently sharers, then it sends invalidation messages to them
3. Sharers (P2 and P3) reply with Invalidation Acknowledgement to P1

What Info in the Directory?

- **Information that must be kept at the directory**
  - Which caches are currently sharers or owner
  - Which state the block is currently cached

- **Ideally, if the caches support MESI states, then the directory keeps MESI states as well, but:**
  - Transition from Exclusive to Modified in a cache is silent
  - Hence, the directory cannot tell if a block is cached in Exclusive or Modified state

What Info in the Directory

- **Hence, the directory may keep 3 states**
  - **Shared (S)**
    - The block is cached cleanly, possibly by multiple caches
  - **Uncached (U)**
    - The block is uncached or cached in invalid state.
  - **Exclusive or Modified (EM)**
    - The block is either cached in Exclusive or Modified state in only one cache.

Basic DSM Cache Coherence Protocol

Example

- **Cache State: MESI**
- **Directory: Full-bit Vector**
- **3 States:**
  - EM (Exclusive or Modified)
  - S (Shared)
  - U (Uncached)

Coherence Messages

- **Issued by requesting processor:**
  - **Read:**
    - read request from a processor
  - **ReadX:**
    - read exclusive (write) request from a processor that does not have the block
  - **Upgr:**
    - upgrade request (from Shared to Modified), made by a processor that already has the block

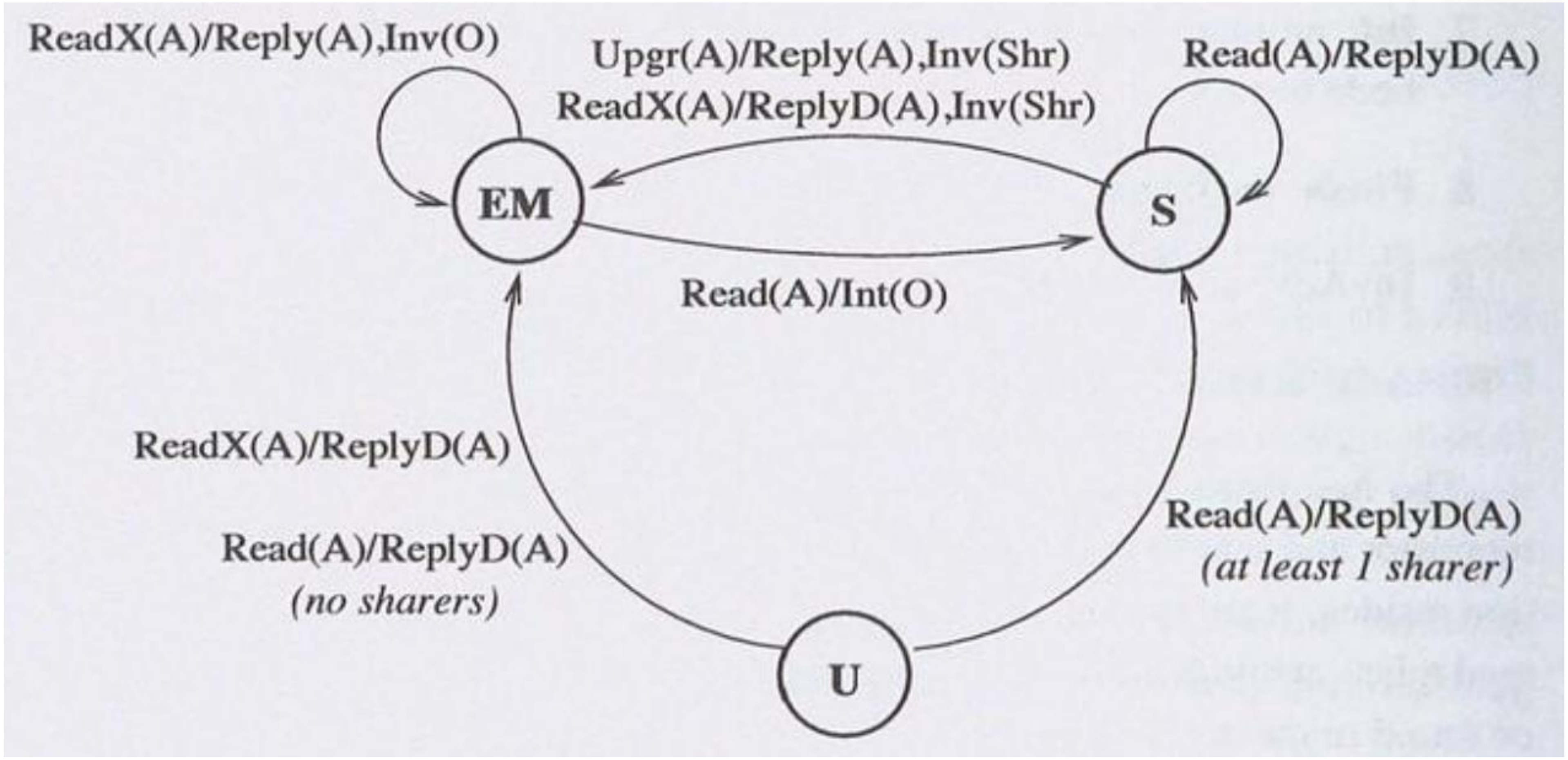
Coherence Messages

- **Issued by home node:**
  - **ReplyD:**
    - home replies with data to requestor
  - **Reply:**
    - home replies not containing data
  - **Inv (Invalidation):**
    - Invalidation request sent by home node to sharers
  - **Int (Intervention):**
    - Intervention request sent by home to owner, asking owner to flush and change to Shared state.

Coherence Messages

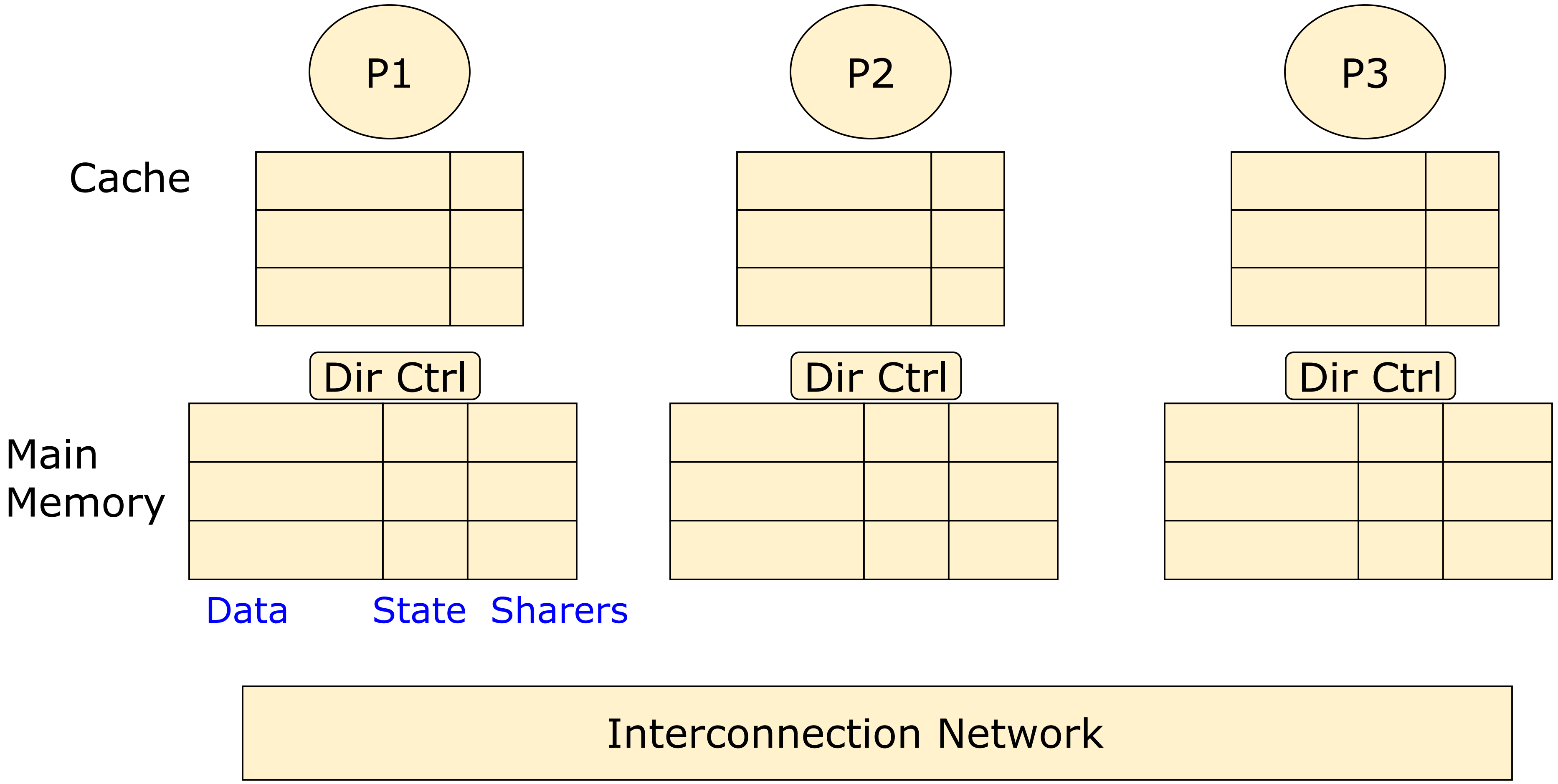
- **Issued by owner of a cache block:**
  - **Flush:**
    - owner flushes data to home + requestor
  - **InvAck:**
    - sharer/owner ack Invalidation msg
  - **Ack:**
    - acknowledgement of receipt of non-invalidation msg

State Transition at the Directory

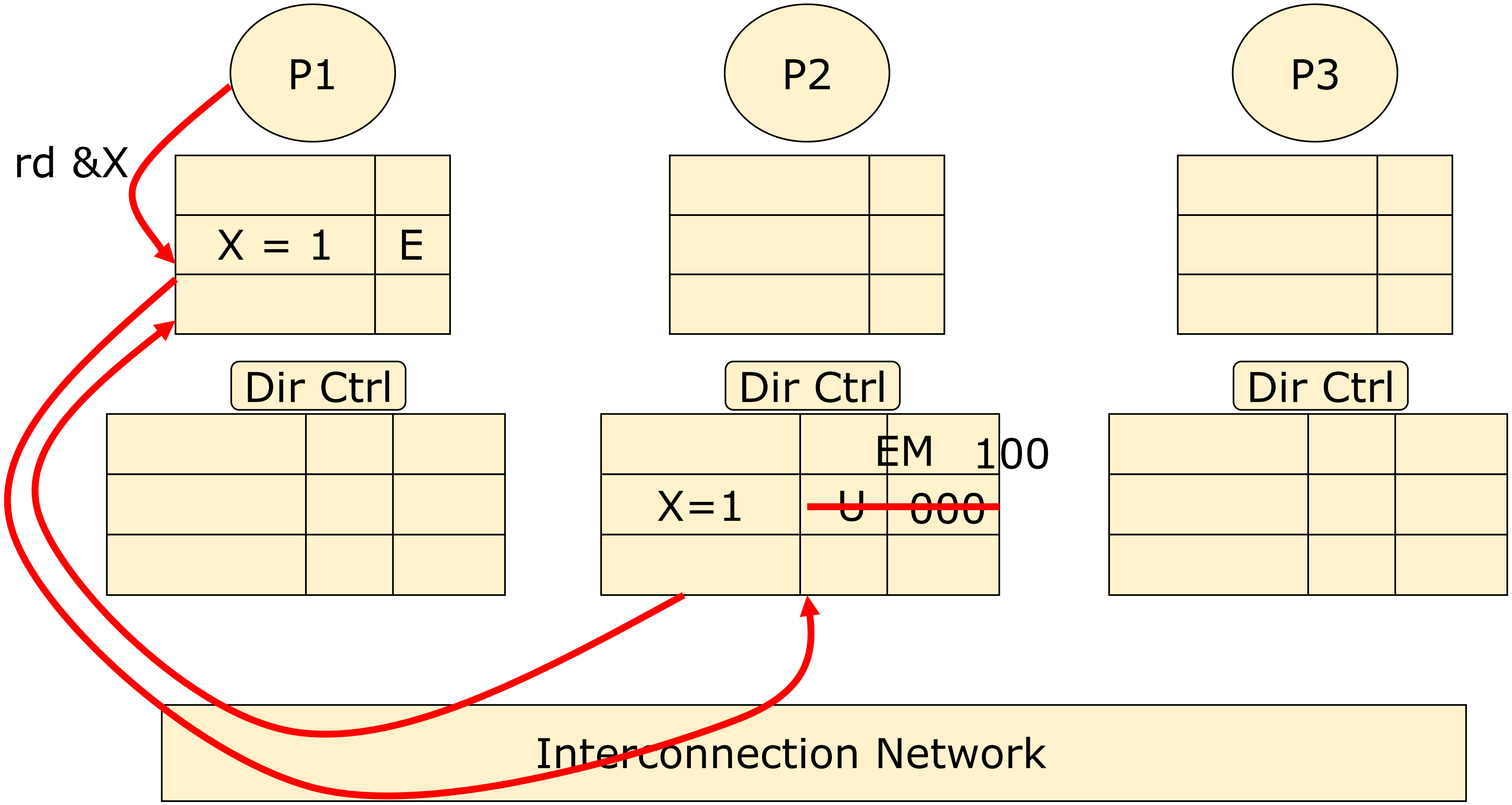


- Legend:
  - <Request-type>(<requestor>) / <Reply-type>(<destination>)
  - O = owner, A = requestor, Shr = sharers

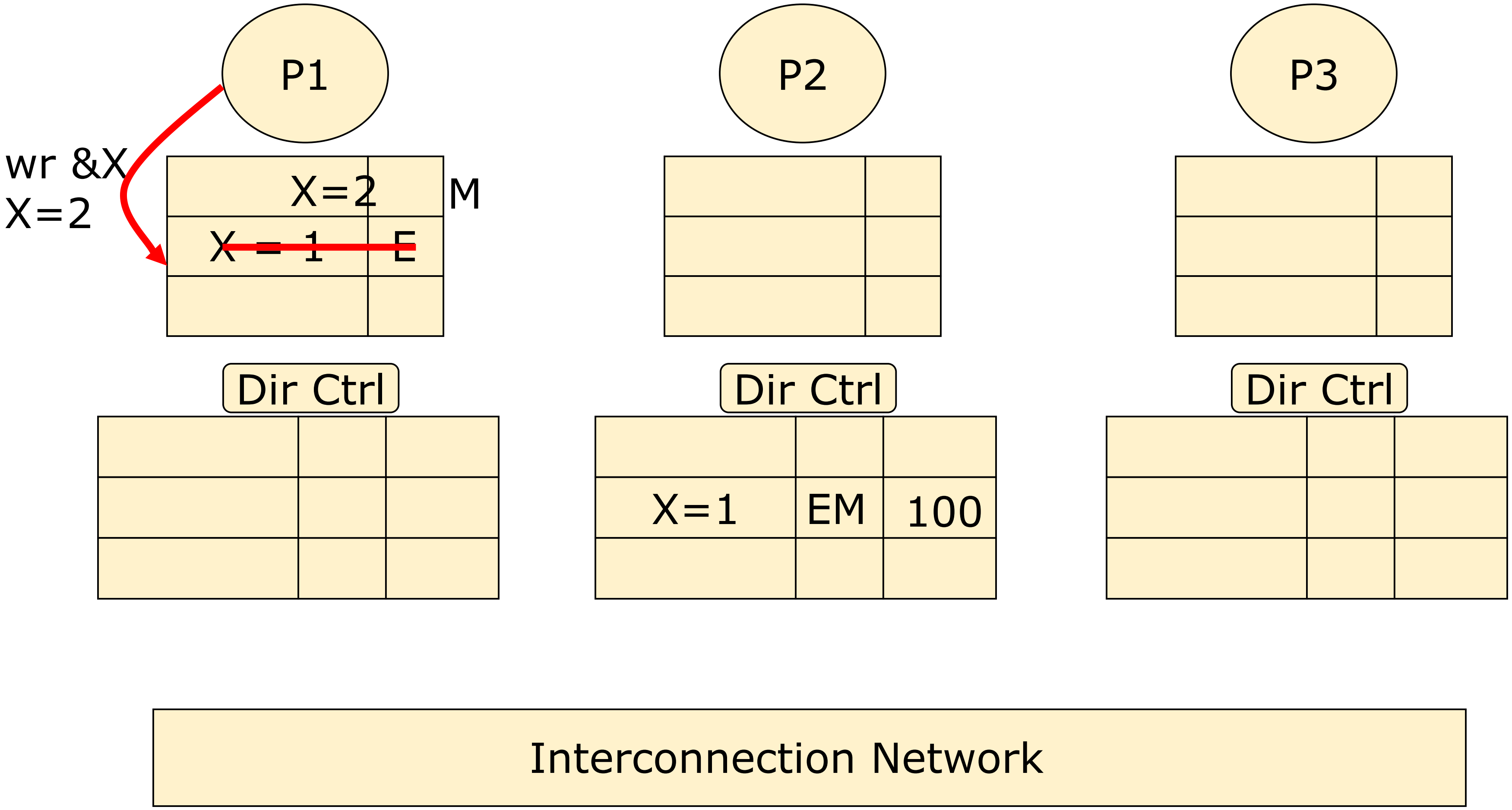
Full-bit Vector Protocol Visualization



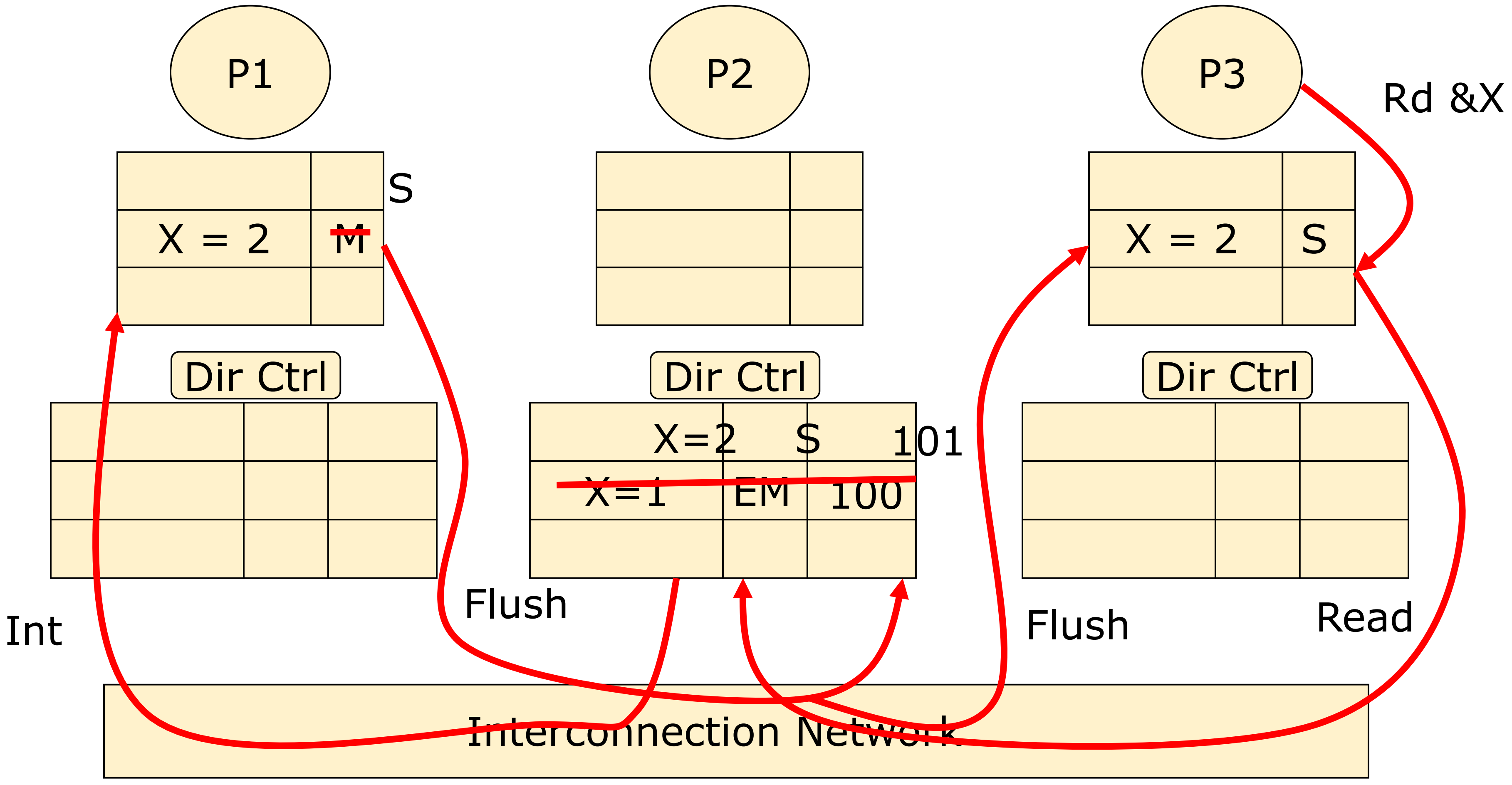
Proc Action	State P1	State P2	State P3	Dir State @Home	Network Msg	Hops
R1	E	-	-	EM, 100	Read (P1-> H), ReplyD (H->P1)	2



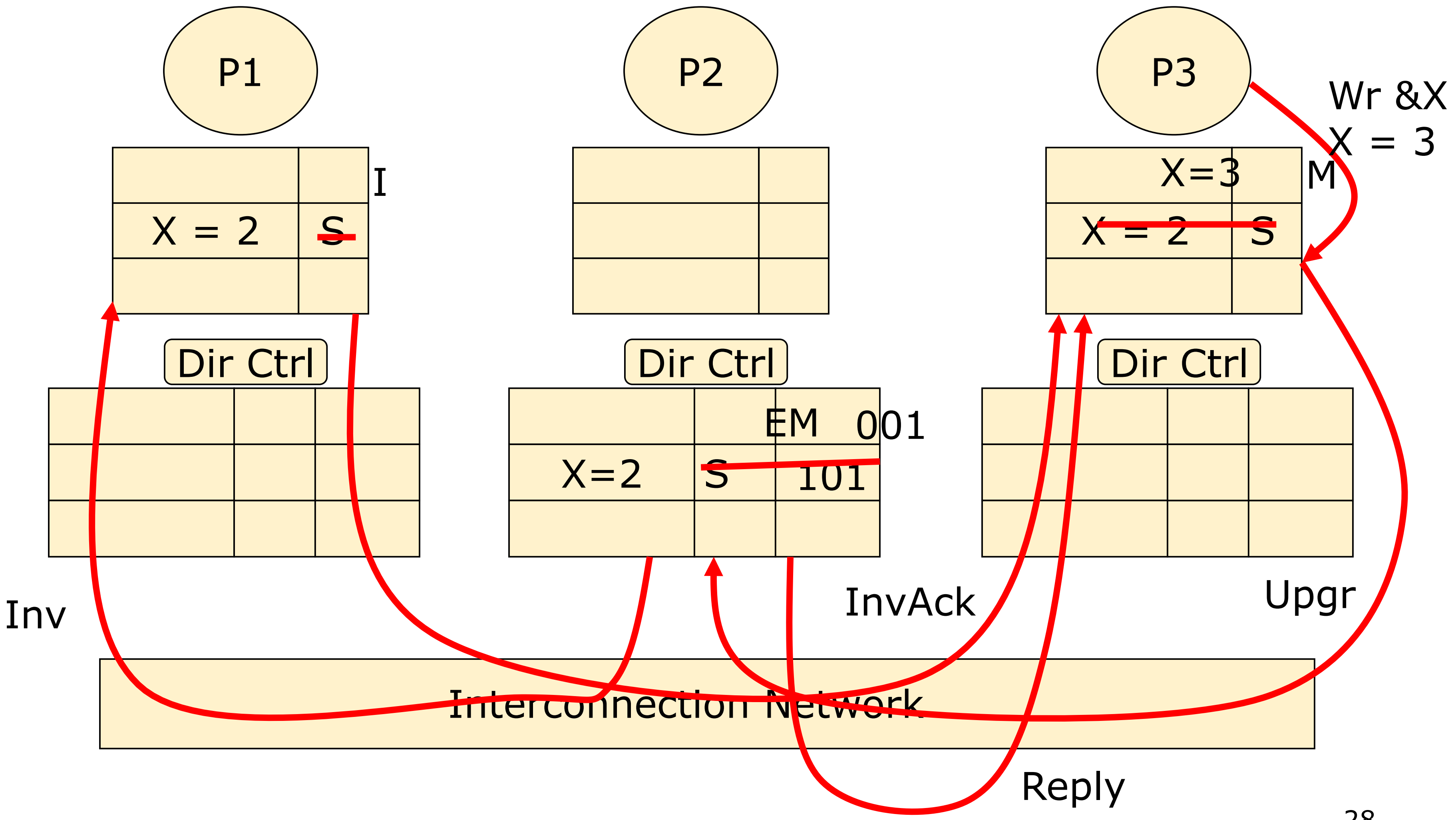
Proc Action	State P1	State P2	State P3	Dir State @Home	Network Msg	Hops
W1	M	-	-	EM, 100	-	0



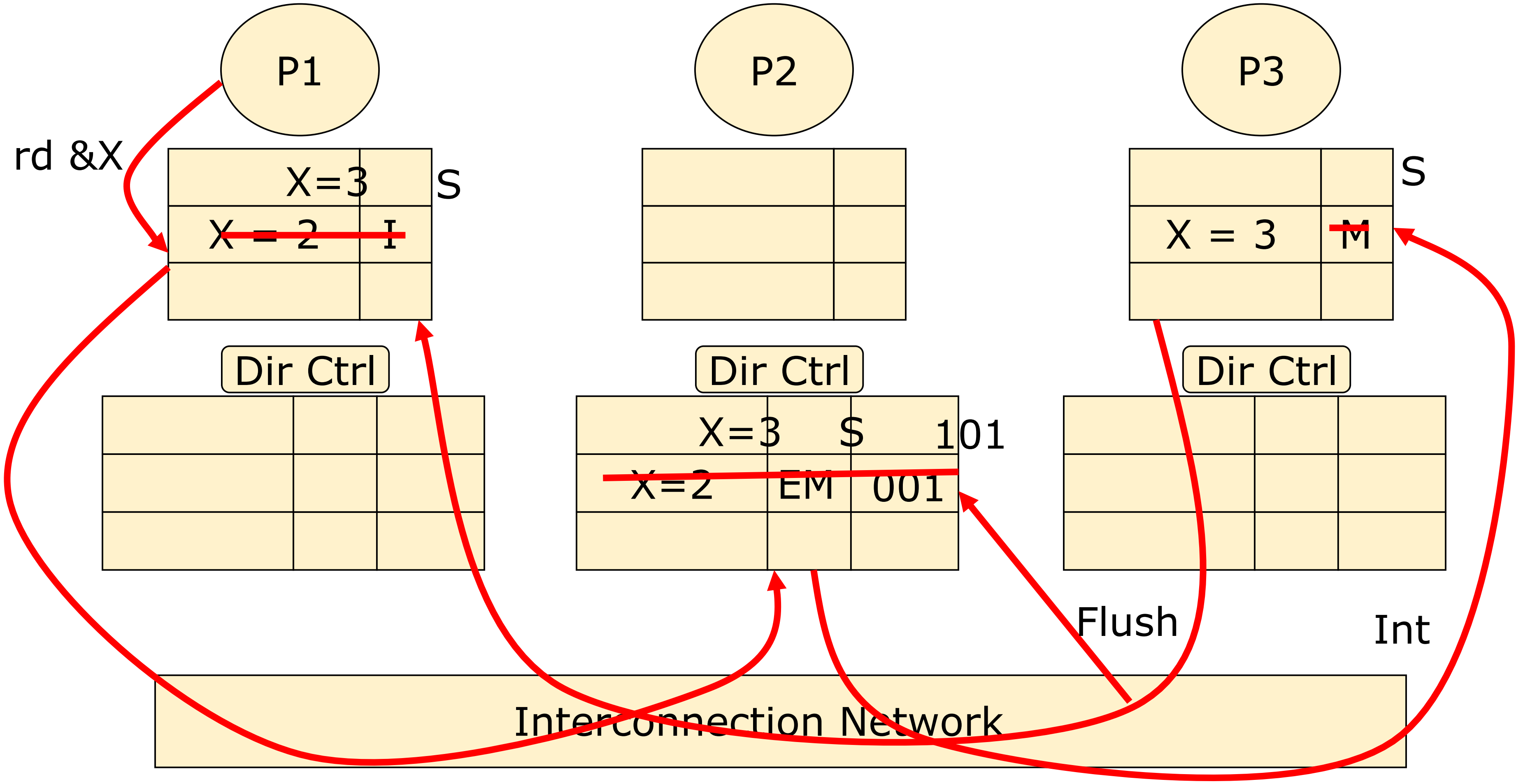
Proc Action	State P1	State P2	State P3	Dir State @Home	Network Msg	Hops
R3	S	-	S	S, 101	Read (P3->H), Int (H->P1), Flush (P1->H, P3)	3



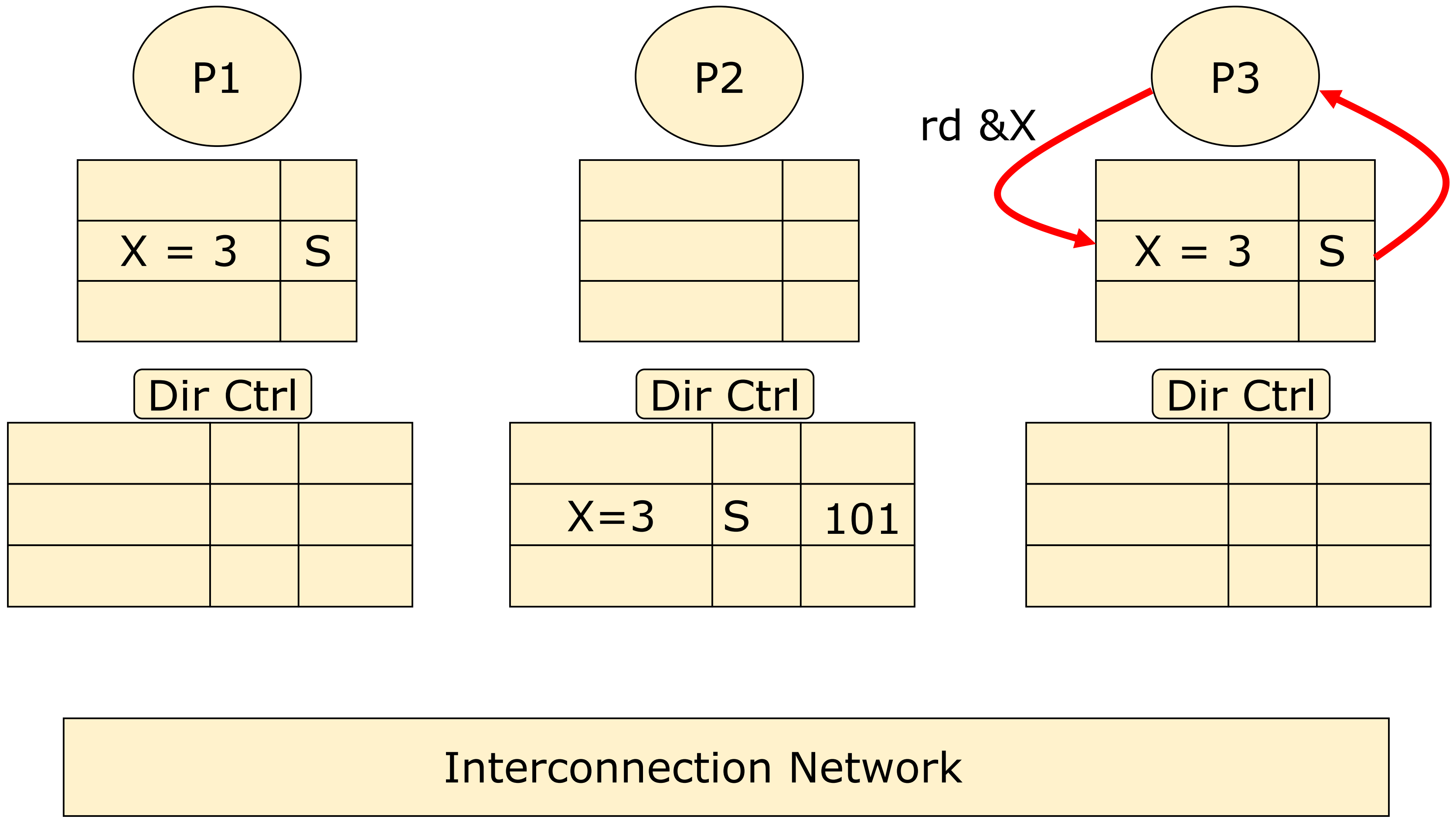
Proc Action	State P1	State P2	State P3	Dir State @Home	Network Msg	Hops
W3	I	-	M	EM, 001	Upgr (P3->H), Reply (H->P3) // Inv (H->P1), InvAck(P1->P3)	3



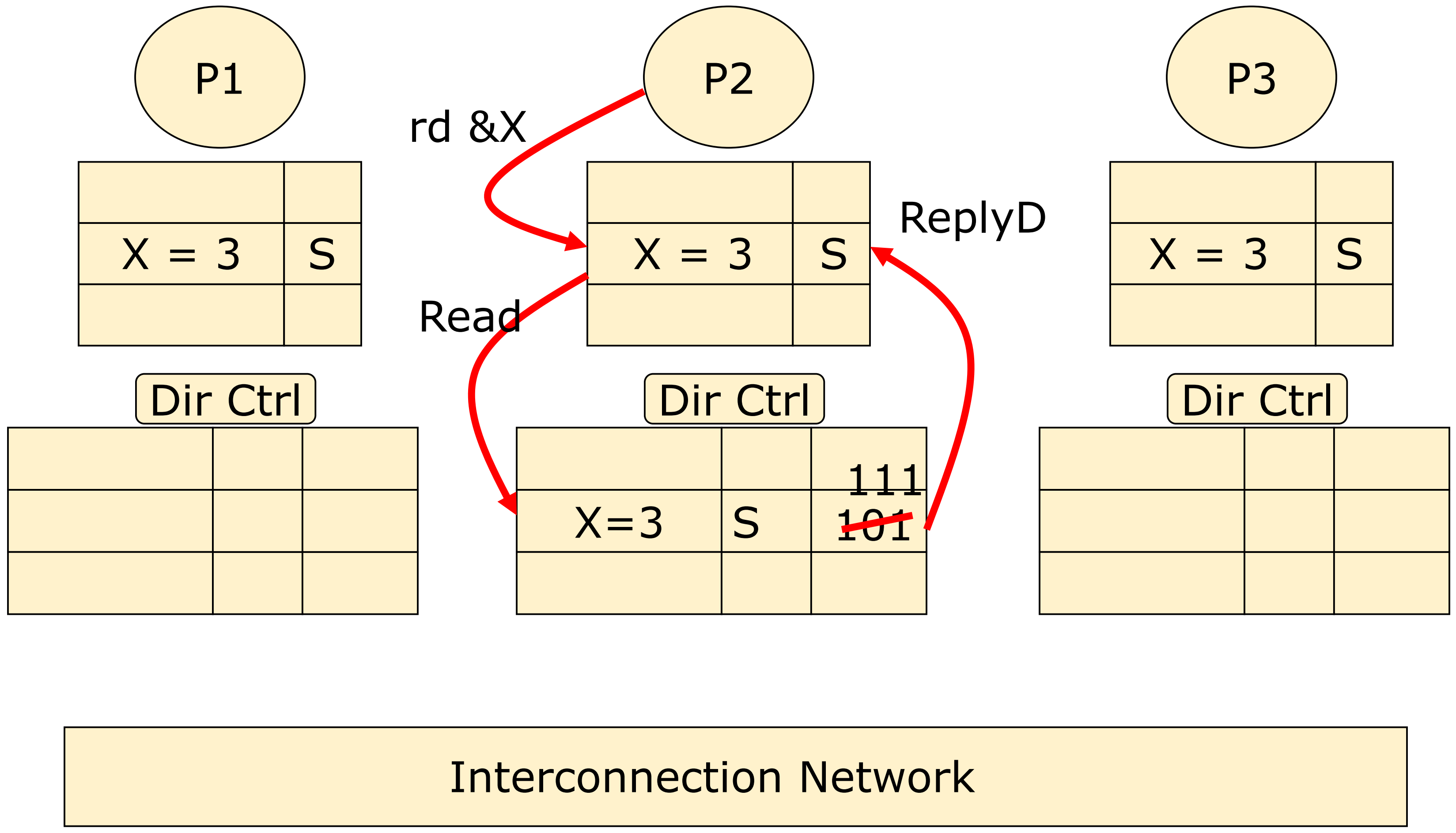
Proc Action	State P1	State P2	State P3	Dir State @Home	Network Msg	Hops
R1	S	-	S	S, 101	Read (P1->H), Int (H->P3), Flush (P3->H, P1)	3



Proc Action	State P1	State P2	State P3	Dir State @Home	Network Msg	Hops
R3	S	-	S	S, 101	-	0



Proc Action	State P1	State P2	State P3	Dir State @Home	Network Msg	Hops
R2	S	S	S	S, 111	Read(P2->H), ReplyD(H->P2)	2



Summary of State Changes

Proc Action	State P1	State P2	State P3	Dir State @Home	Network Msg	Hops
R1	E	-	-	EM, 100	Read (P1-> H), ReplyD (H->P1)	2
W1	M	-	-	EM, 100	-	0
R3	S	-	S	S, 101	Read (P3->H), Int (H->P1), Flush (P1->H, P3)	3

W3	I	-	M	EM, 001	Upgr (P3->H),
					Reply (H->P3) // Inv (H->P1),
					InvAck(P1->P3)
						3

R1	S	-	S	S, 101	Read (P1->H), Int (H->P3), Flush (P3->H, P1)	3
R3	S	-	S	S, 101	-	0
R2	S	S	S	S, 111	Read(P2->H), ReplyD(H->P2)	2

Performance Criteria of Directory Schemes

- **Main criteria:**
  - Traffic on invalidations
  - Latency of invalidations
  - Storage overheads
- **Less important criteria:**
  - Traffic on interventions
  - Latency of interventions
- **Invalidation involves multiple sharers vs. intervention involves a single owner**

Note:
Invalidation:
- Any → Invalid

Intervention:
- Exclusive/Modified → Shared

课程内容

- **参考材料：**
  - Fundamentals of Parallel Computer Architecture. Chapter 11 “Distributed Shared Memory Multiprocessors”.