

Arquitectura de Computadores



TEMA 6 Multiprocesadores: Coherencia, sincronización y consistencia

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVIA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP: 689 45 44 70

Curso 2015-2016

Contenidos

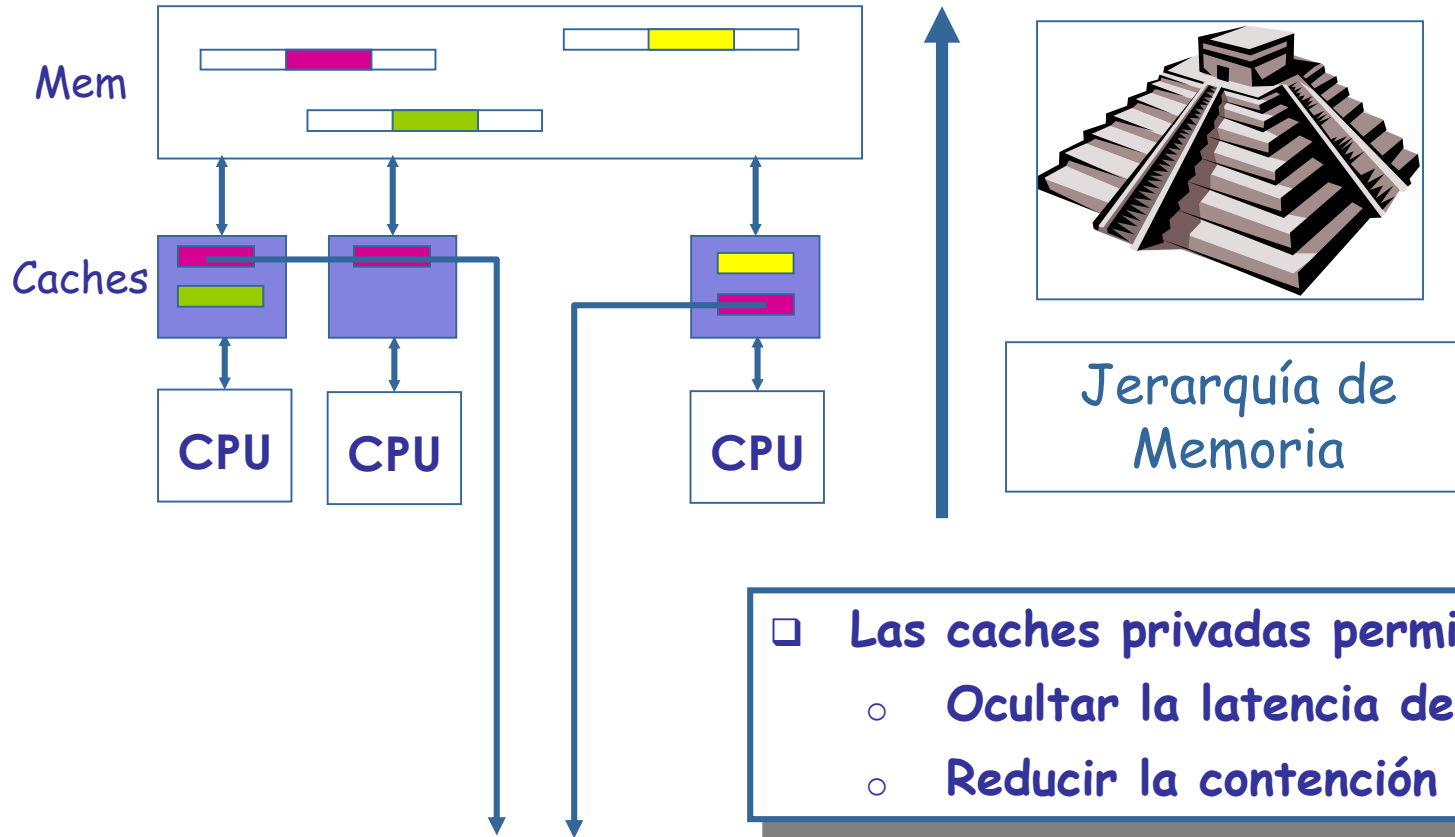
- ❑ Jerarquía de memoria extendida
- ❑ El problema de la coherencia cache
- ❑ Protocolos de coherencia "Snoopy"
 - o Protocolo Snoopy de dos estados
 - o Protocolo MSI: invalidación de tres estados
 - o Protocolo MESI: invalidación de cuatro estados
- ❑ Coherencia cache basada en directorio
 - o Directorio plano basado en memoria
 - o Directorio plano basado en cache
- ❑ Sincronización
- ❑ Consistencia de memoria
- ❑ Bibliografía
 - o Cap 5 y 8 de [CuSi99]

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Jerarquía de memoria extendida



- ❑ Las caches privadas permiten:
 - Ocultar la latencia de los accesos
 - Reducir la contención

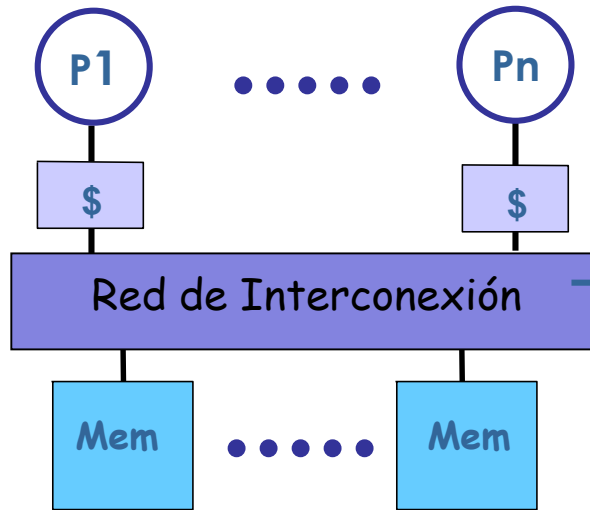
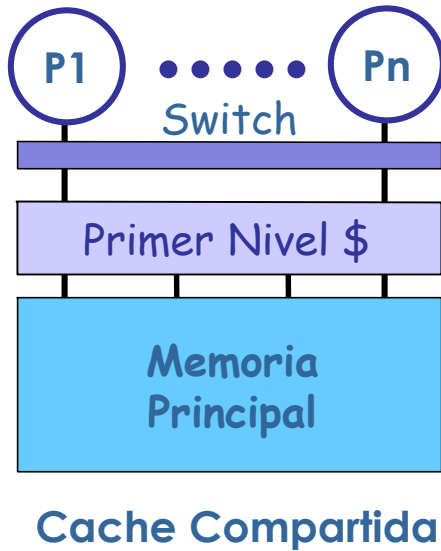
CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Cartagena99

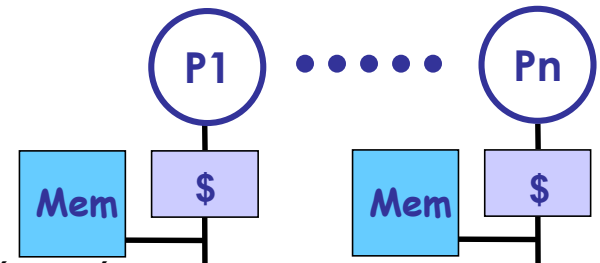
Alternativas de implementación

Escalabilidad



- Bus Compartido
- Red punto a punto

UMA, SMP



CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

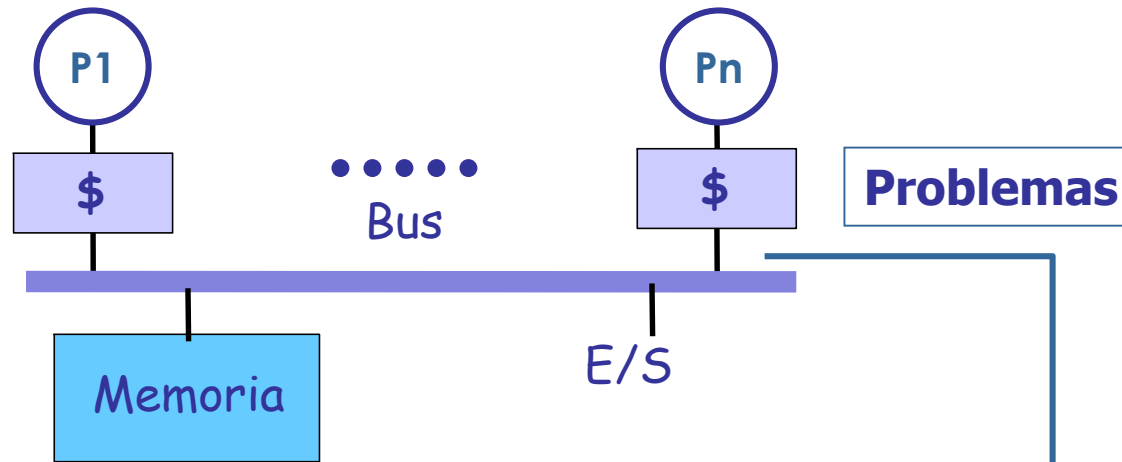
ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Cartagena99

NUMA: Memoria Distribuida

Alternativas de implementación

- ❑ Una alternativa ampliamente usada: SMP basado en Bus Compartido



- ❑ Servidores departamentales
- ❑ Estaciones de trabajo
- ❑ Bloques de construcción básicos

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVIA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Modelo intuitivo de memoria

❑ Programa secuencial

o Las posiciones de memoria se utilizan para "comunicar valores" entre distintos puntos del programa. Cuando se lee una posición se devuelve el último valor escrito en ella

❑ Espacio de direcciones compartido en sistemas con un único procesador (Multiprogramación)

o Cuando se lee una posición se devuelve el último valor escrito en ella, independientemente del proceso (thread) que realizó la última escritura sobre dicha posición

o Las caches no interfieren con el uso de múltiples procesos (threads) en un procesador, ya que todos ellos ven la memoria a través de la misma jerarquía

❑ Multiprocesadores de Memoria Compartida

o Objetivo: Nos gustaría que el resultado de ejecutar un programa que usa varios procesos (threads) sea el mismo independientemente de si los procesos se ejecutan en paralelo (multiprocesadores) o de forma

Cartagena99

CLASES PARTICULARES, TUTORIAS TECNICAS ONLINE
LLAMA O ENVIA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

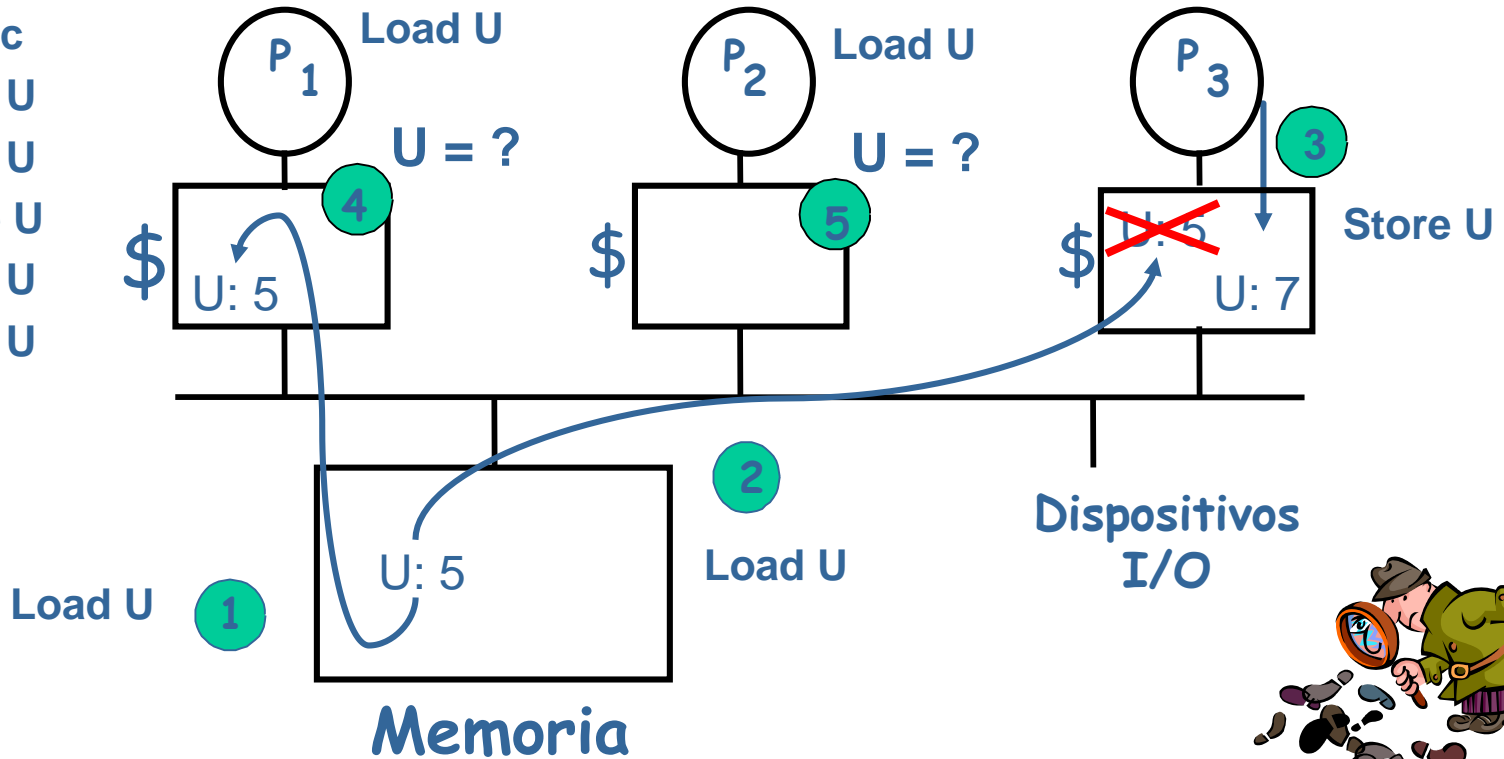
INCORRISTENTES

Problema de la coherencia cache

Transacciones

(Proc) Operac

1. (P1) Load U
2. (P3) Load U
3. (P3) Store U
4. (P1) Load U
5. (P2) Load U



Cartagena99

CLASES PARTICULARES, TUTORIAS TECNICAS ONLINE
LLAMA O ENVIA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Soluciones de "Grano Grueso"

- ❑ Problemas de coherencia cache en uniprosesadores:
operaciones de E/S a través de dispositivos DMA
 - o Dispositivo DMA escribe en memoria: El procesador puede seguir viendo valores antiguos en la cache.
 - o Dispositivo DMA lee de memoria: El DMA puede leer un valor antiguo en el caso de que se utilice write-back

- ❑ Alternativas (dado que las operaciones de E/S son mucho menos frecuentes que las operaciones de acceso a memoria)
 - o Evitar usar la cache:
 - Los segmentos de memoria involucrados en operaciones de E/S se marcan como **No-Cacheables** o bien se utilizan operaciones load/store no-cacheables
 - o Sacar de la cache antes de E/S (**soporte del SO**):
 - Las páginas de memoria involucradas en cualquier operación de E/S son eliminadas de la cache (**flush**) previamente por el SO
 - o Usar la cache para E/S:
 - El tráfico de E/S pasa por todos los niveles de la jerarquía de memoria

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Soluciones de "Grano Grueso"

- ❑ En multiprocesadores
 - o La escritura o lectura de variables compartidas es un evento frecuente.
 - o No es práctico:
 - Deshabilitar la cache para datos compartidos
 - Invocar al SO en cada referencia a una variable compartida
- ❑ Todos los μ Procesadores actuales proporcionan mecanismos para soportar la coherencia
 - o Transparente al software

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVIA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

□ ¿Qué significa que un sistema de memoria es coherente?

- o Una operación de lectura retorna siempre el **último** valor que fue escrito en la posición de memoria correspondiente, independientemente del procesador que efectúa la lectura o escritura
- o Dos aspectos clave:
 - 1) Qué valor debe ser devuelto en una lectura **Coherencia**.
 - 2) Cuándo un valor escrito debe ser devuelto por una operación de lectura **Consistencia**

□ ¿Cuándo un sistema de memoria es coherente?

- o Una lectura del procesador P a la posición X de memoria, que sigue a una escritura de P en X, sin ninguna otra escritura debe retornar el valor escrito por P
- o Una lectura por un procesador a la posición X que sigue a una escritura de otro procesador en X, retorna el valor escrito si no hay ninguna otra escritura y están suficientemente separadas en el tiempo.

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

□ Serialización de escrituras: ejemplo

Si P1 ejecuta:

- o LD R1, X

- y se obtiene en R1 el valor V1 escrito por el procesador P2 en la posición de memoria X

Y después P1 ejecuta:

- o LD R2, X

- y se obtiene en R2 el valor V2 escrito por el procesador P3 en la posición de memoria X

...entonces es imposible que cualquier otro procesador, Pi, que lea la posición de memoria X obtenga primero el

Cartagena99

CLASES PARTICULARES, TUTORIAS TÉCNICAS ONLINE
LLAMA O ENVIA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Políticas (o protocolos) para mantener la coherencia

❑ Invalidación en Escritura / Coherencia Dinámica

- o Al escribir en un bloque se invalidan todas las otras copias (Múltiples lectores , un solo escritor)
- o Escrituras consecutivas al mismo bloque (no necesariamente a la misma palabra) efectuadas desde el mismo procesador se realizan localmente (No hay copias). Sólo es necesario una transacción en el medio de comunicaciones en múltiples escrituras al mismo dato
- o ¿Qué ocurre en un fallo de lectura?
 - Con write-through: la memoria esta siempre actualizada
 - Con write-back: es necesario búsqueda (snoop) en caches remotas para encontrar el último valor.
 - La cache dueña del último valor lo proporciona al solicitante (y a la Mp)

❑ Actualización en Escritura

- o Al escribir en un bloque se actualizan todas las copias
- o Típicamente con write-through, pocos procesadores
- o Escrituras consecutivas a la misma palabra requiere

Cartagena99

CLASES PARTICULARES, TUTORIAS TECNICAS ONLINE
LLAMA O ENVIA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Políticas (o protocolos) para mantener la coherencia

- ❑ Los protocolos usados para la invalidación o actualización dependen de la red de interconexión utilizada
- ❑ Si la red de interconexión **permite broadcast** eficiente las operaciones de invalidación o actualización se pueden enviar de forma simultánea a todos los controladores.
 - o **Protocolo Snoopy (buses): Observación del bus**
 - Cada controlador de cache está observando los eventos que suceden en el bus y toma decisiones en consecuencia.
 - Un controlador de cache envía eventos al bus en función de las peticiones de acceso a memoria que recibe de su procesador → Envía órdenes de **invalidación** o **actualización** de forma simultánea a todos los demás controladores.
- ❑ Si la red de interconexión **no permite broadcast** (o el broadcast no es eficiente), la invalidación o la actualización se envía únicamente a aquellas caches que tienen una copia del bloque.

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Protocolos *Snoopy*

- ❑ El Arbitraje del bus Impone un Orden.
 - o Las Transacciones del Bus son visibles en el mismo orden por todos los controladores de cache.
- ❑ Observar/Espiar el bus



Sólo es necesario extender la funcionalidad del controlador de cache

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVIA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

U) Transacciones relevantes observadas en el bus

Protocolos *Snoopy*

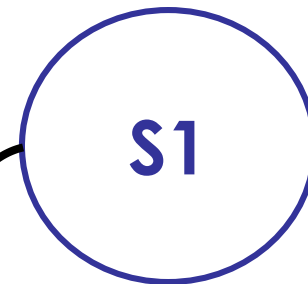
- Diagrama de estados de un bloque de cache: convenio de representación
 - o Cada transición es disparada por la observación de un evento en el procesador o el bus
 - o Una transición puede implicar que el controlador ejecute alguna acción sobre su cache, u ordene una cierta acción al bus

Notación:

Op del Procesador/ **Acción a realizar**

o bien

Evento en BUS / **Acción a realizar**



Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

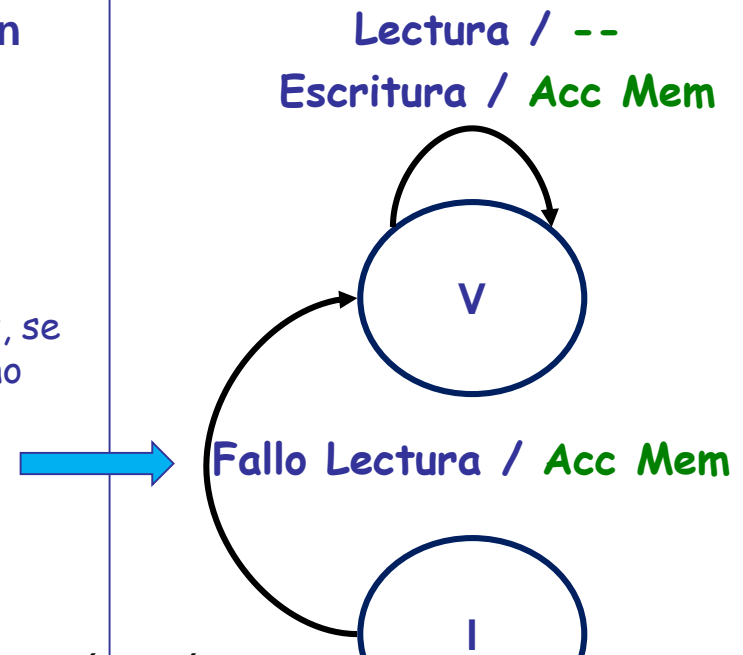
ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Protocolo *Snoopy* de 2 Estados

- Aplicado a: Política de invalidación en escritura + Escritura directa (write through).
 - o Sin Asignación en Escritura (*no write allocate*)

Ejemplo previo: Diagrama de Estados en un Uniprocador

- Cada bloque tiene un estado (I,V)
 - Nota: Aunque solamente los bloques residentes en \$ tienen un bit de validez, se considera que el estado de los bloques no residentes es Inválido
- Estado inicial : bloques inválidos
- Fallo Lectura : Inválido → Válido
 - Puede generar reemplazamiento
- Escritura: no cambia el estado



Cartagena99

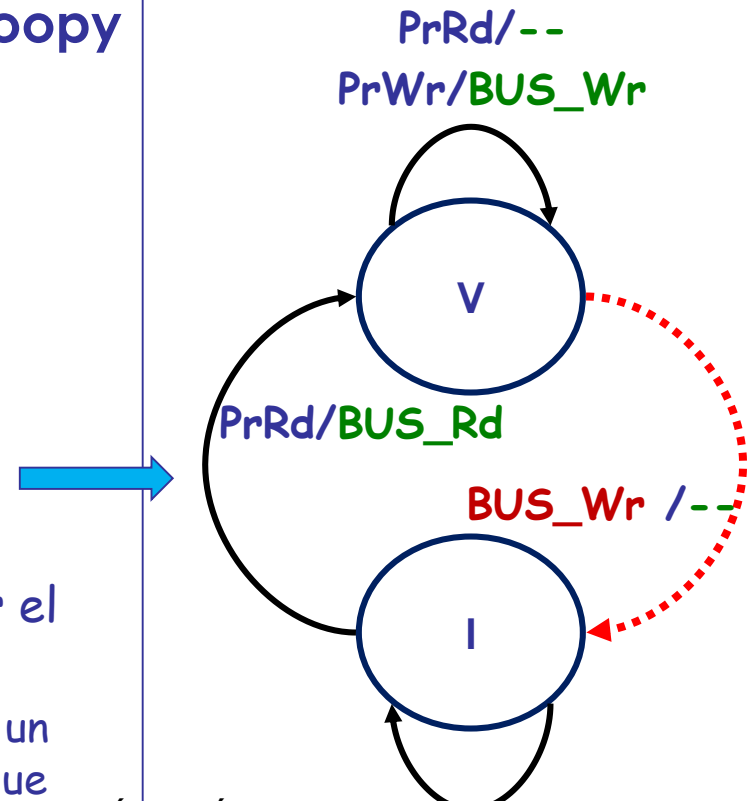
CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Protocolo *Snoopy* de 2 Estados

Diagrama de Estados en Protocolo Snoopy

- Operaciones del Procesador
 - Lecturas: PrRd
 - Escrituras: PrWr
- Transacciones del BUS ordenadas por el controlador de \$
 - BUS_Rd (fallo de lectura)
 - BUS_Wr (relevante: actualiz. Mp)
- Transacciones en el BUS detectadas por el controlador de \$ (línea - - - - -)
 - BUS_Wr (Si otro procesador ha escrito un bloque que está en esta \$: **Invaldar** bloque



Cartagena99

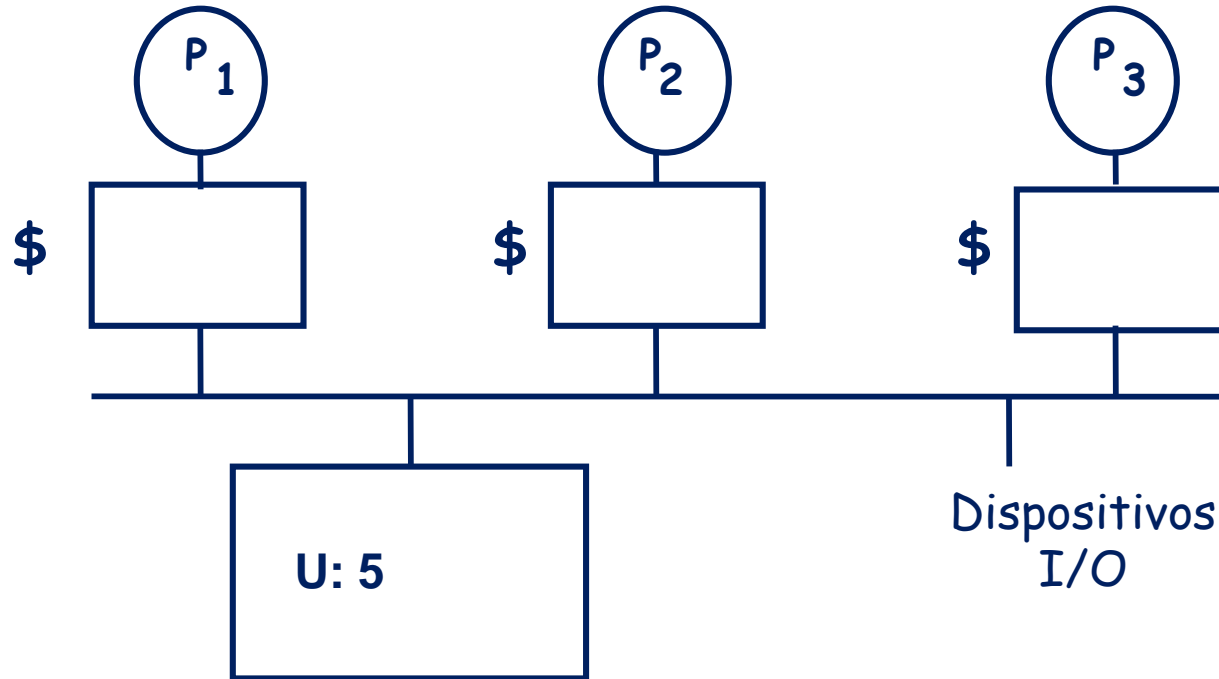
CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

- - -

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Protocolo *Snoopy* de 2 Estados

□ Ejemplo: Escritura directa + Invalidación



(P1) Load U

(P3) Load U

Memoria

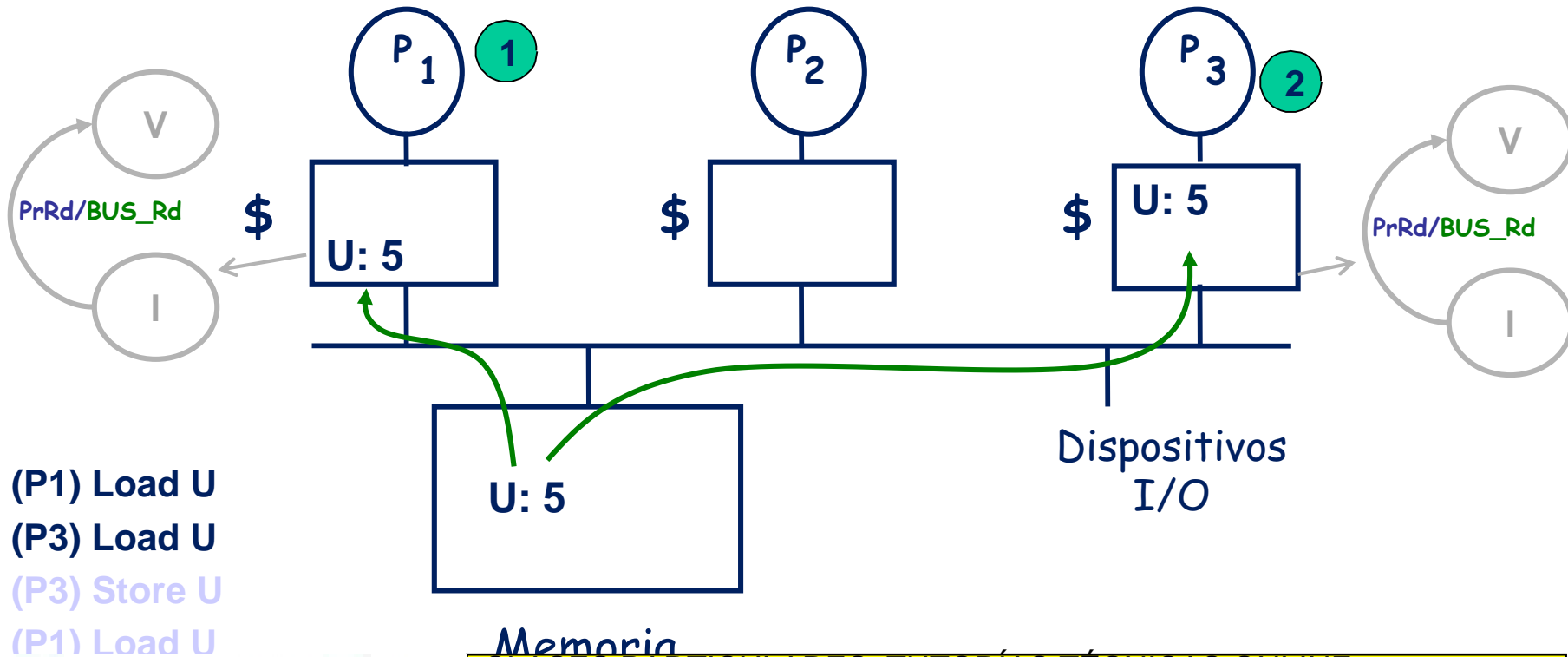
CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVIA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Cartagena99

Protocolo *Snoopy* de 2 Estados

□ Ejemplo: Escritura directa + Invalidación



- (P1) Load U
- (P3) Load U
- (P3) Store U
- (P1) Load U

Memoria

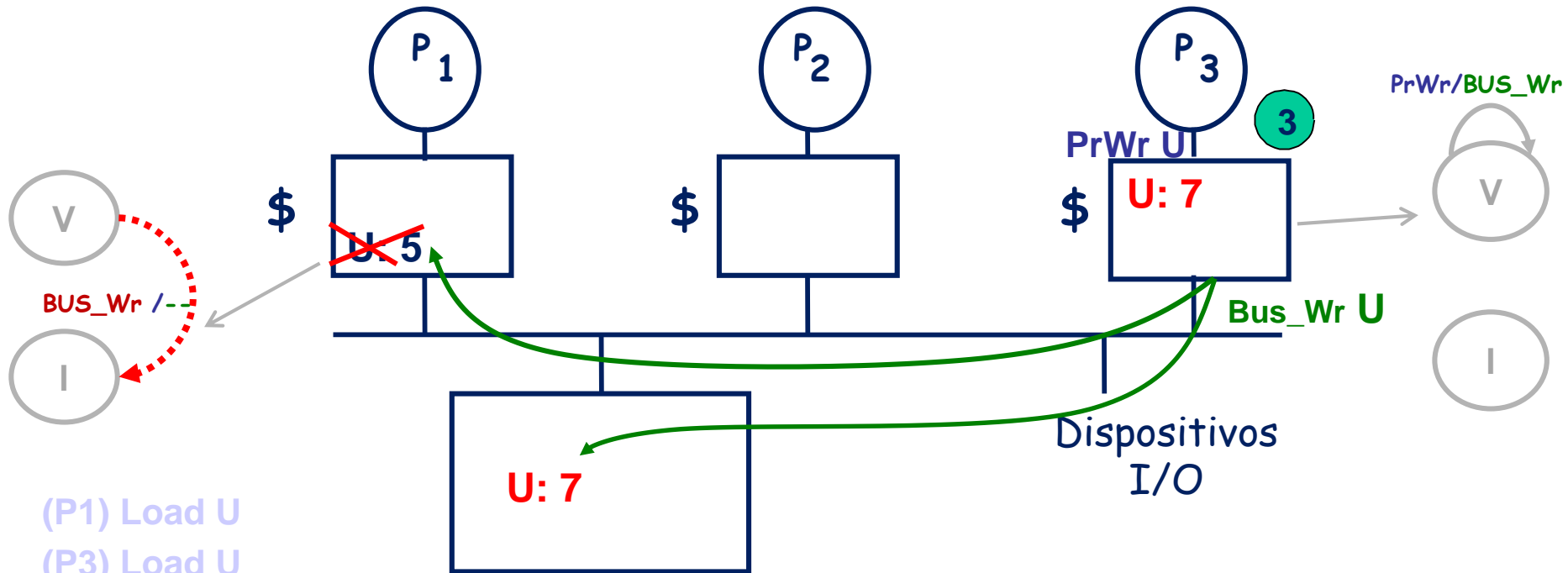
CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVIA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Cartagena99

Protocolo *Snoopy* de 2 Estados

□ Ejemplo: Escritura directa + Invalidación



- (P1) Load U
- (P3) Load U
- (P3) Store U

Memoria

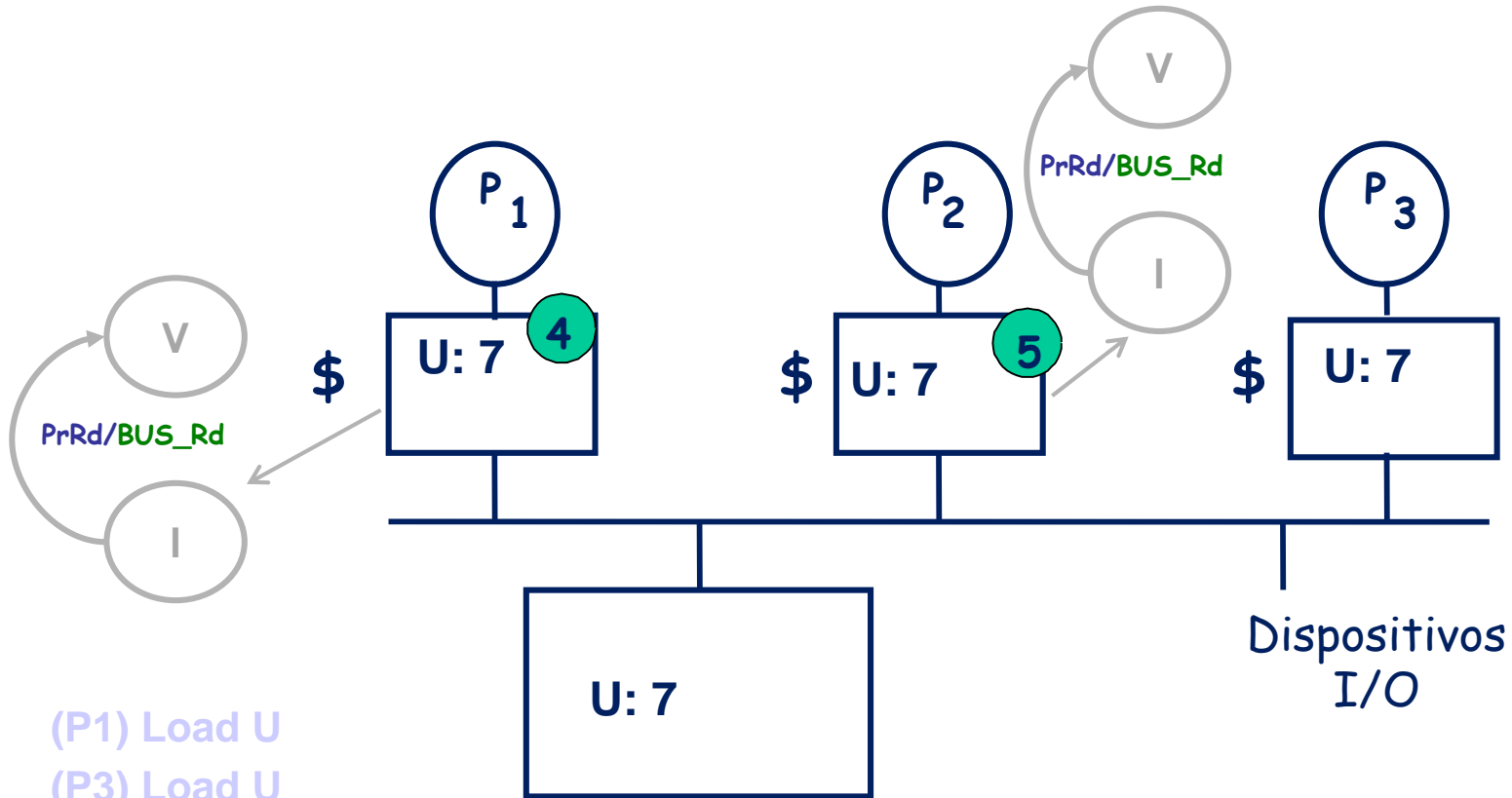
CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP: 689 45 44 70

Cartagena99

Protocolo *Snoopy* de 2 Estados

□ Ejemplo: Escritura directa + Invalidación



(P1) Load U
(P3) Load U
(P3) Store U

Memoria

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Cartagena99

Protocolo *Snoopy* de 2 Estados

- ❑ ¿Es coherente el Protocolo Snoopy de 2 estados?
 - o ¿Existe una Ordenación Secuencial de las operaciones de Memoria?

- o Escritura directa y 1 solo nivel de cache
- o Bus atómico
- o Operaciones atómicas

Orden parcial de escrituras = Orden del bus

- o Propagación de escrituras
 - Las escrituras son visibles a todos los controladores
- o Serialización de Escrituras

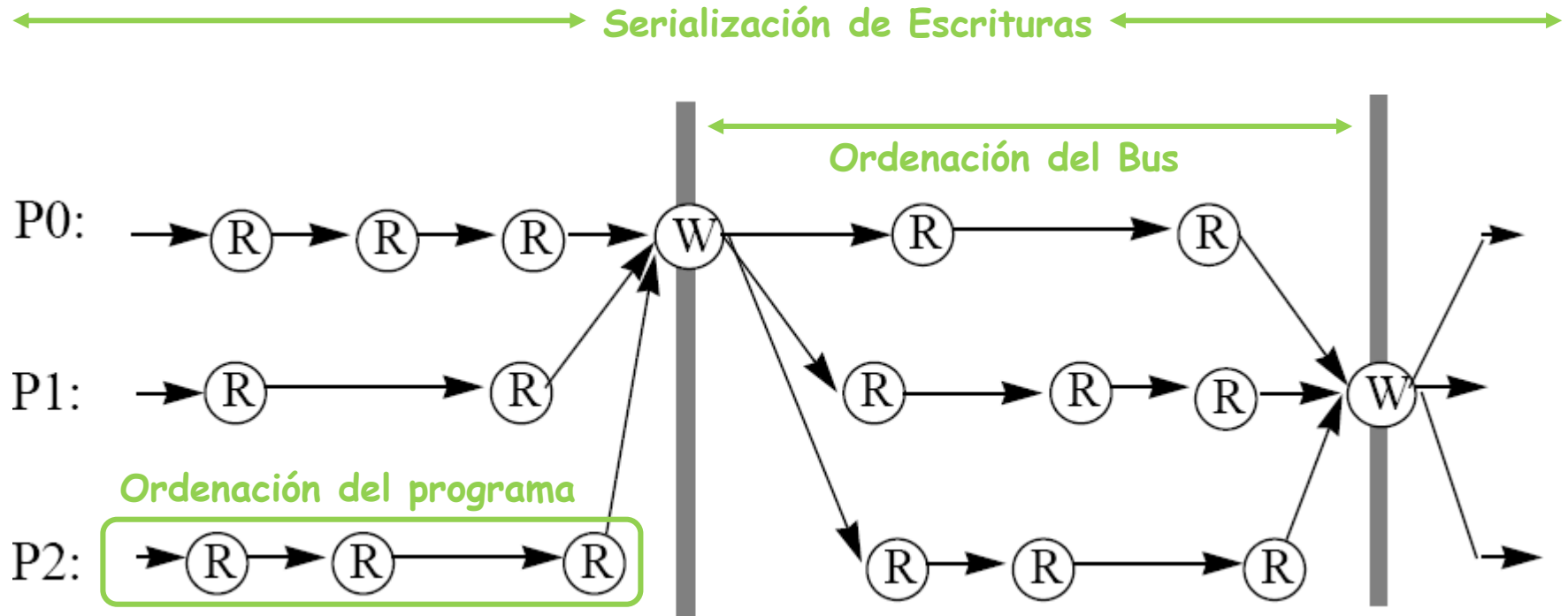
Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVIA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Protocolo *Snoopy* de 2 Estados

□ ¿Es coherente el Protocolo Snoopy de 2 estados?



□ Las transacciones del bus asociadas con escrituras segmentan los flujos individuales de cada programa.

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Protocolo *Snoopy* de 2 Estados

- ❑ Problema: Cada escritura hecha por cada procesador implica una transacción en el Bus → Consumo de Ancho de Banda

- ❑ Ejemplo:
 - o Procesador a 2000 MHz
 - o CPI = 1
 - o 15% stores de 8 bytes

 - o Se ejecutan 300 Millones de stores/s por procesador
 - $(0,15 \text{ stores/instr} \times 1 \text{ instr/ciclo} \times 2000 \times 10^6 \text{ ciclos/seg})$
 - o 2400 MB/s por procesador
 - o Un bus con un ancho de banda de 10GB/s sólo puede soportar 4 procesadores sin saturarse

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Recordatorio: caches write-back en un uniprocador

- ❑ Clave para conseguir un uso eficiente del ancho de banda (limitado) que proporciona un bus compartido:
 - o Post-Escritura (*write-back*)
 - o Asignación en Escritura (*write-allocate*)

- ❑ Diagrama de transición de estados de una cache con Post-Escritura (uniprocador)
 - o **Tres** estados para un bloque de memoria:
 - Invalido (o no presente)
 - Válido (*clean*)
 - **Modificado** (*dirty*)
 - o Dos tipos de accesos a memoria que implican transacciones de Bus
 - Lecturas (*BusRd*): provocadas por

Cartagena99

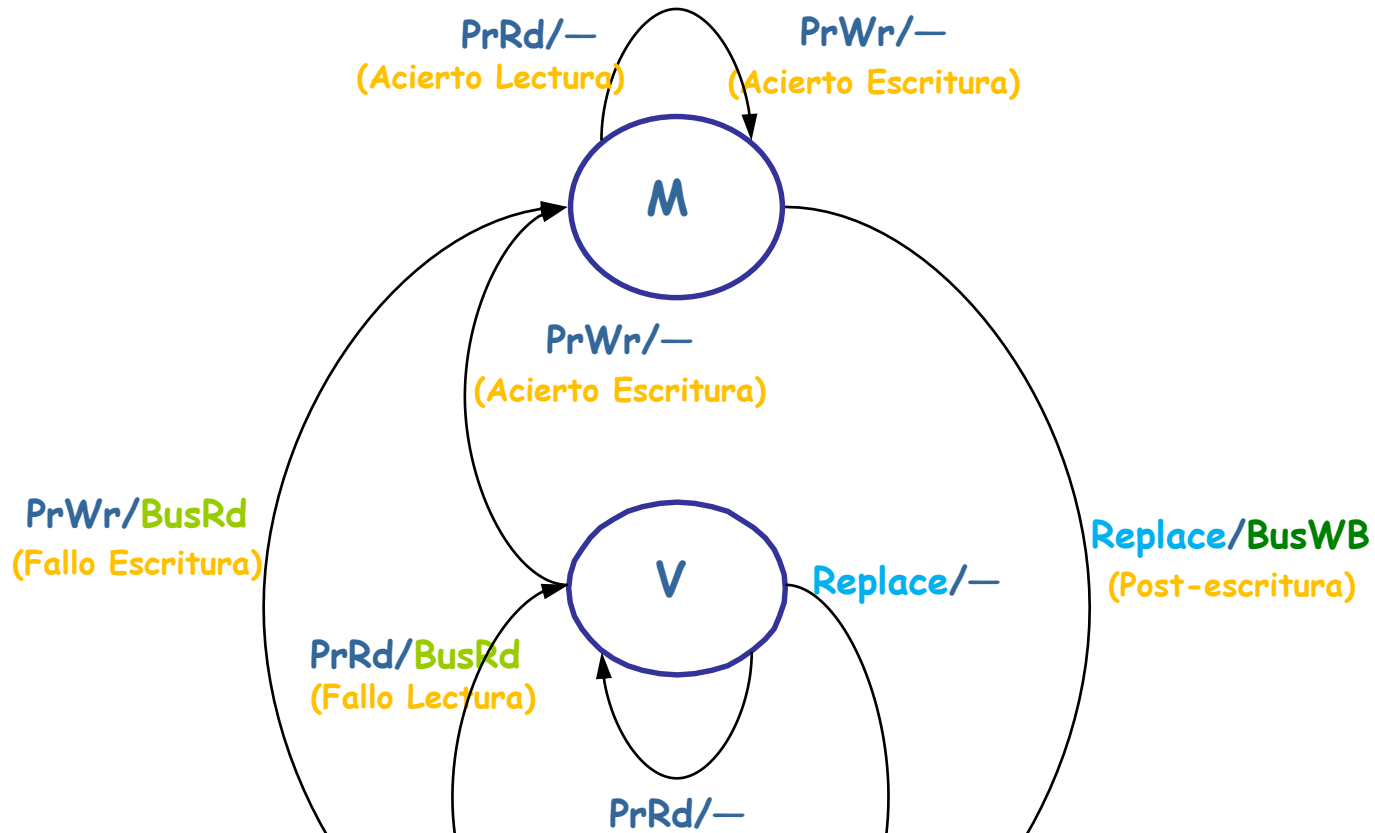
CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

- Reemplazamiento

Recordatorio: caches write-back en un uniprocador

- Diagrama de transición de estados de una cache con Post-Escritura



Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVIA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Protocolo MSI: invalidación de 3 estados (multiprocesador)

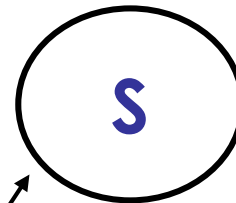
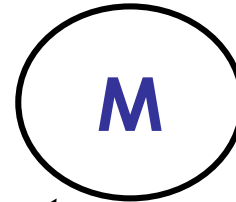
- ❑ Multiprocesador: Tres estados para un bloque de memoria
 - **M**odificado: Exclusivo
 - **S**hared (Compartido): Válido
 - **I**nválido
- ❑ Nueva transacción ordenada al Bus
 - BusRdx (Lectura con Exclusividad)
- ❑ Nueva respuesta del controlador de \$
 - **Flush** (volcar bloque de \$ a bus)

Estado M:

- Sólo hay 1 cache con copia válida (la copia de la memoria principal está anticuada)
- La exclusividad implica que la cache puede modificar el bloque sin notificárselo a nadie

Estado S:

- El bloque está presente en la cache y



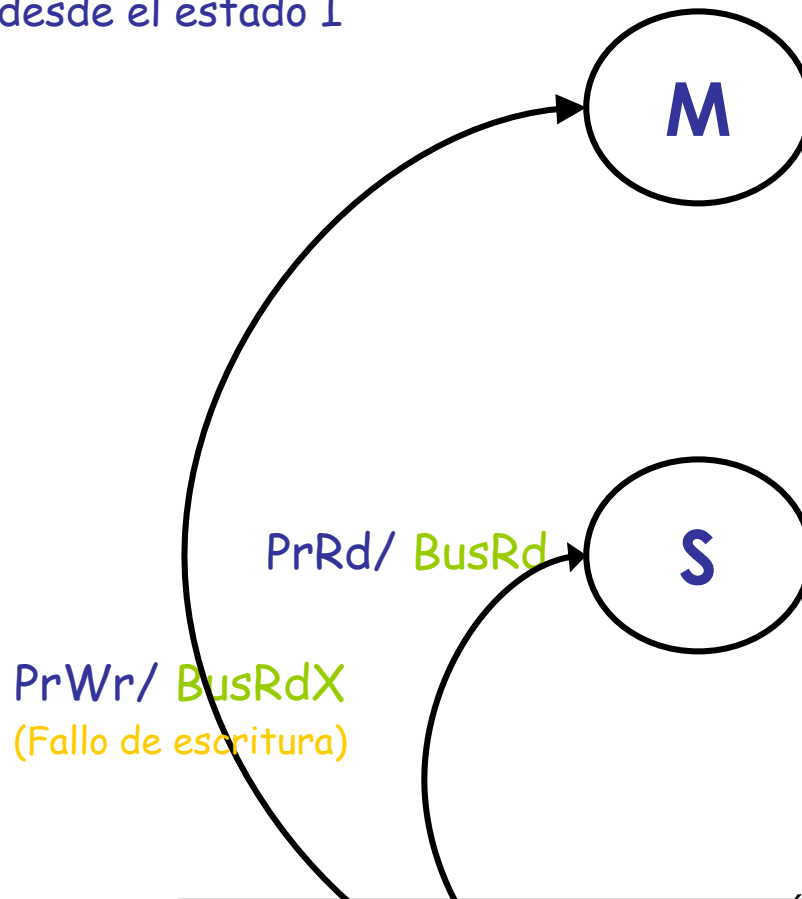
CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Cartagena99

Protocolo MSI: invalidación de 3 estados

□ Transiciones desde el estado I



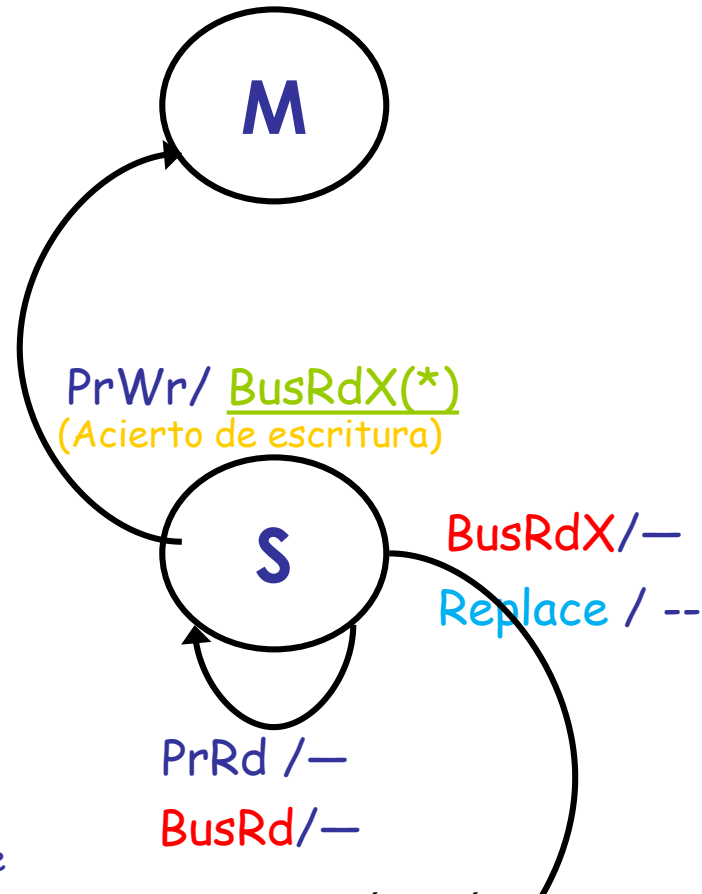
Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Protocolo MSI: invalidación de 3 estados

- Transiciones desde el estado S



(*) Alternativa; Como la \$ ya tiene el bloque de

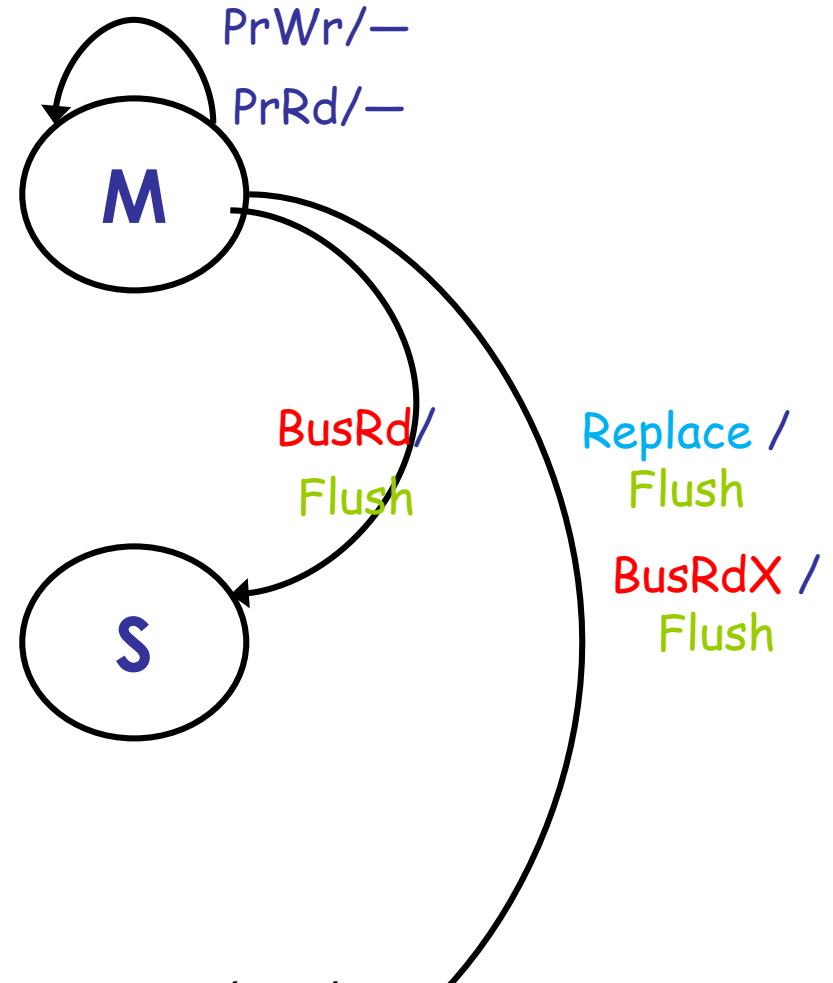
Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVIA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Protocolo MSI: invalidación de 3 estados

- Transiciones desde el estado M



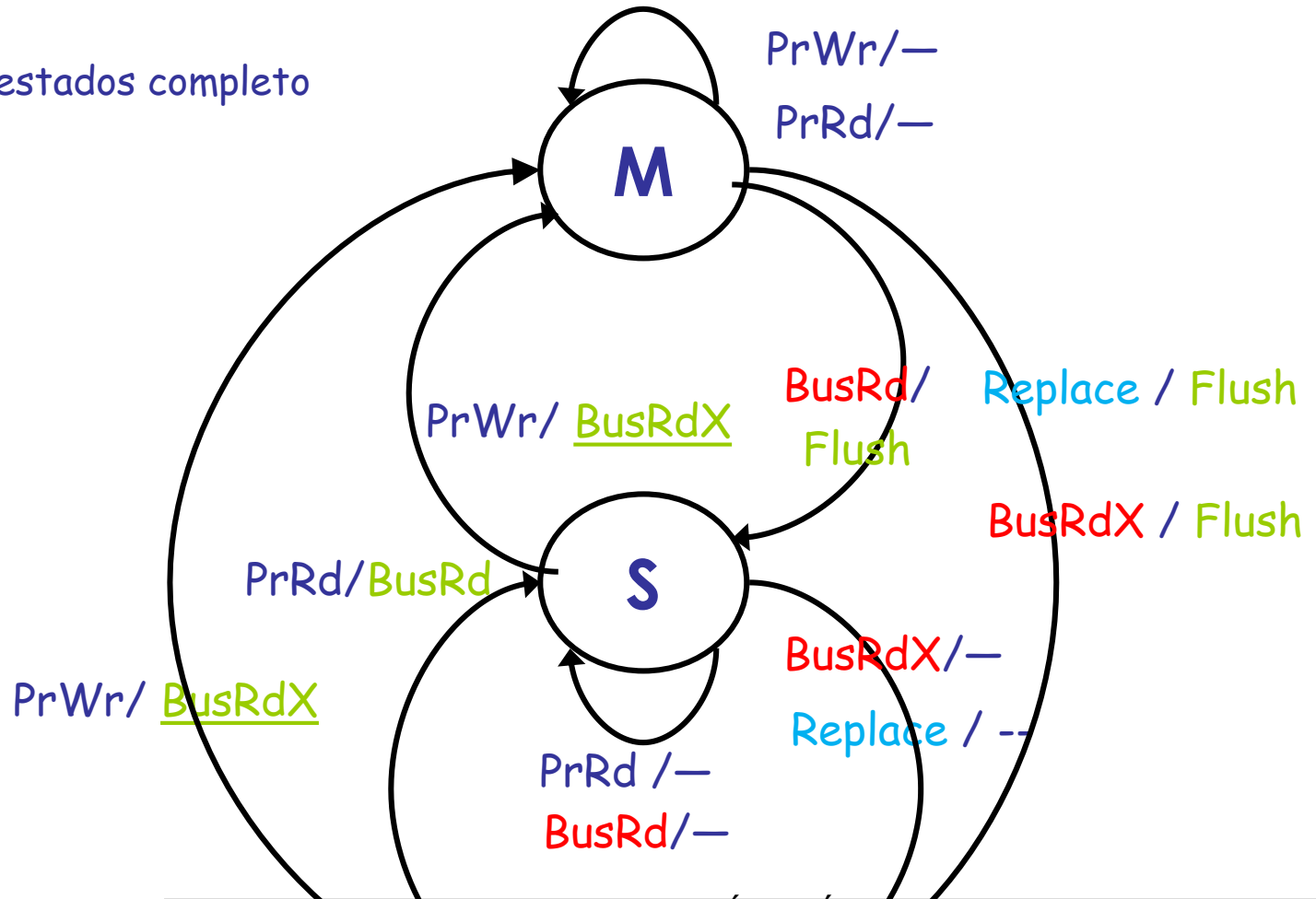
Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVIA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Protocolo MSI: invalidación de 3 estados

- Diagrama de estados completo

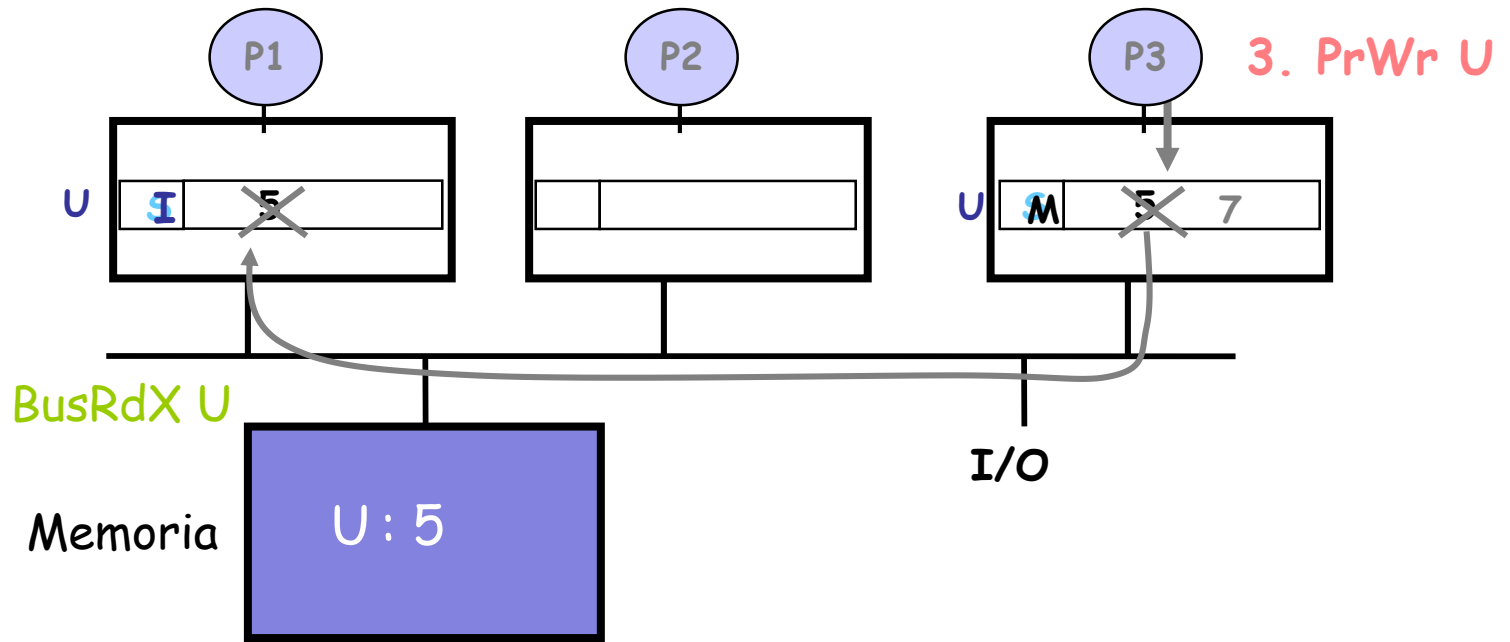


Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP: 689 45 44 70

Protocolo MSI: invalidación de 3 estados



Operación	Estado P1	Estado P2	Estado P3	Transacción Bus	Datos Suministrados
P1 Lee U	S	—	—	BusRd	Memoria
P3 Lee U	S	—	S	BusRd	Memoria

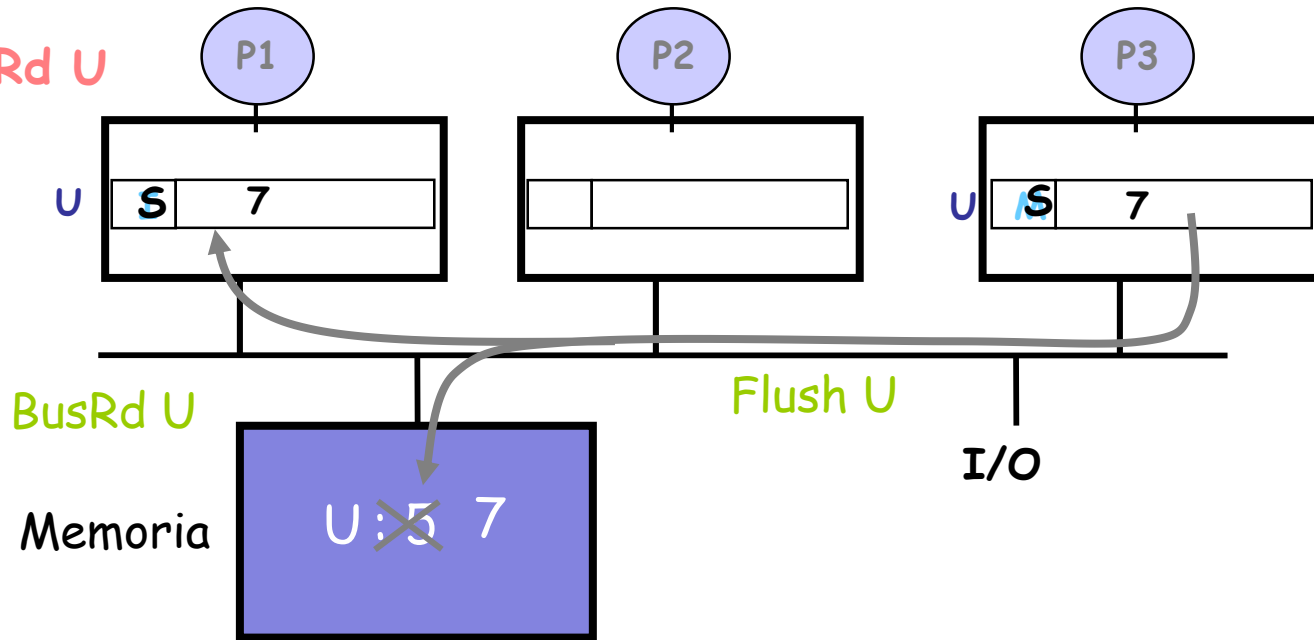
CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
 LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
 CALL OR WHATSAPP: 689 45 44 70

Cartagena99

Protocolo MSI: invalidación de 3 estados

4. PrRd U



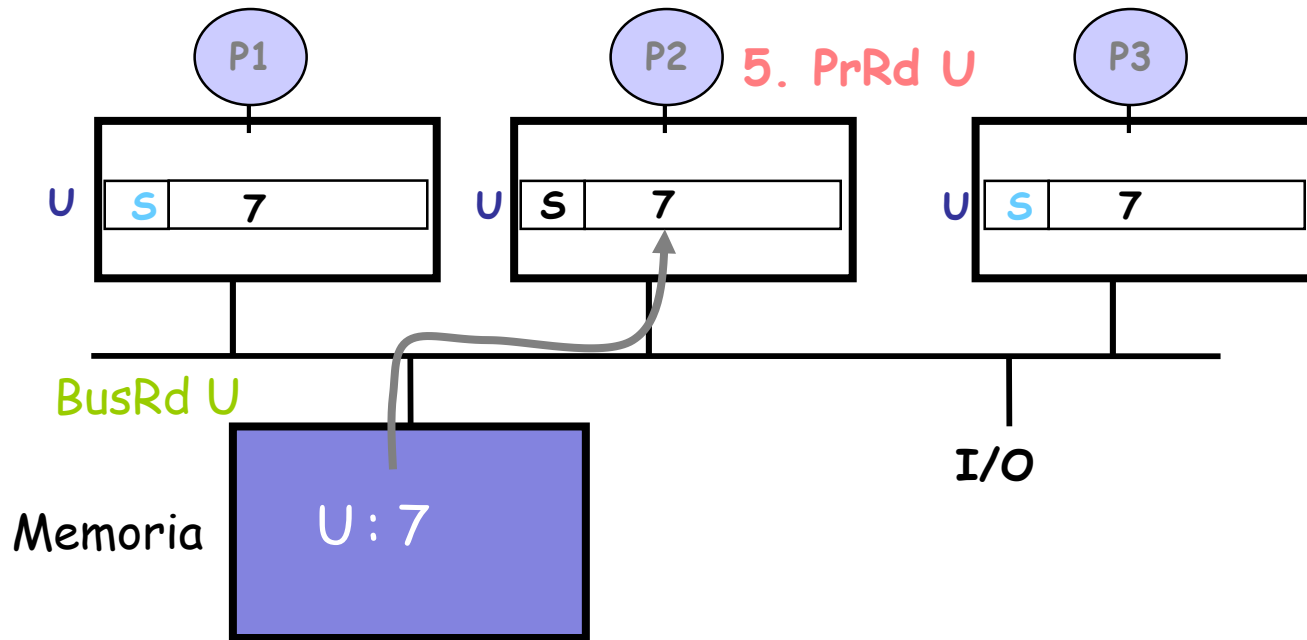
Operación	Estado P1	Estado P2	Estado P3	Transacción Bus	Datos Suministrados
P1 Lee U	S	—	—	BusRd	Memoria
P3 Lee U	S	—	S	BusRd	Memoria

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Protocolo MSI: invalidación de 3 estados



Operación	Estado P1	Estado P2	Estado P3	Transacción Bus	Datos Suministrados
P1 Lee U	S	—	—	BusRd	Memoria
P3 Lee U	S	—	S	BusRd	Memoria

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Implementación de protocolos *Snoopy*

- ❑ Carreras (races) entre escrituras
 - No puede escribirse en la cache hasta que se tiene el bus
 - Otro procesador podría tomar el bus antes y escribir el mismo bloque en su cache
 - Dos pasos:
 - Arbitrar el bus
 - Poner el fallo sobre el bus y completar la operación
 - Bus de transacciones partidas
 - Las transacciones sobre el bus no son atómicas: puede haber varias transacciones en marcha al mismo bloque
- ❑ Los diferentes procesadores deben acceder al bus: datos y direcciones
- ❑ Los procesadores "espían" el bus:
 - Si la dirección coincide con algún "tag" invalidar o actualizar
- ❑ Consulta de tag interfiere con uso de la cache por la CPU
 - Solución : duplicar los tag
- ❑ El bus serializa las escrituras. Solo un procesador realiza

Cartagena99

CLASES PARTICULARES, TUTORIAS TECNICAS ONLINE
LLAMA O ENVIA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Protocolo MESI: invalidación de 4 estados

□ Problema Protocolo MSI

- o Multiprogramación: carga de trabajo típica multiprocesadores pequeña escala
- o El protocolo MSI no se comporta bien con **Aplicaciones Secuenciales**
 - **Lectura-Modificación de un dato:** El protocolo MSI debe generar 2 transacciones de bus aunque no exista compartición
 - Al leer (I→S) seguida de al modificar (S→M)

□ Protocolo MESI

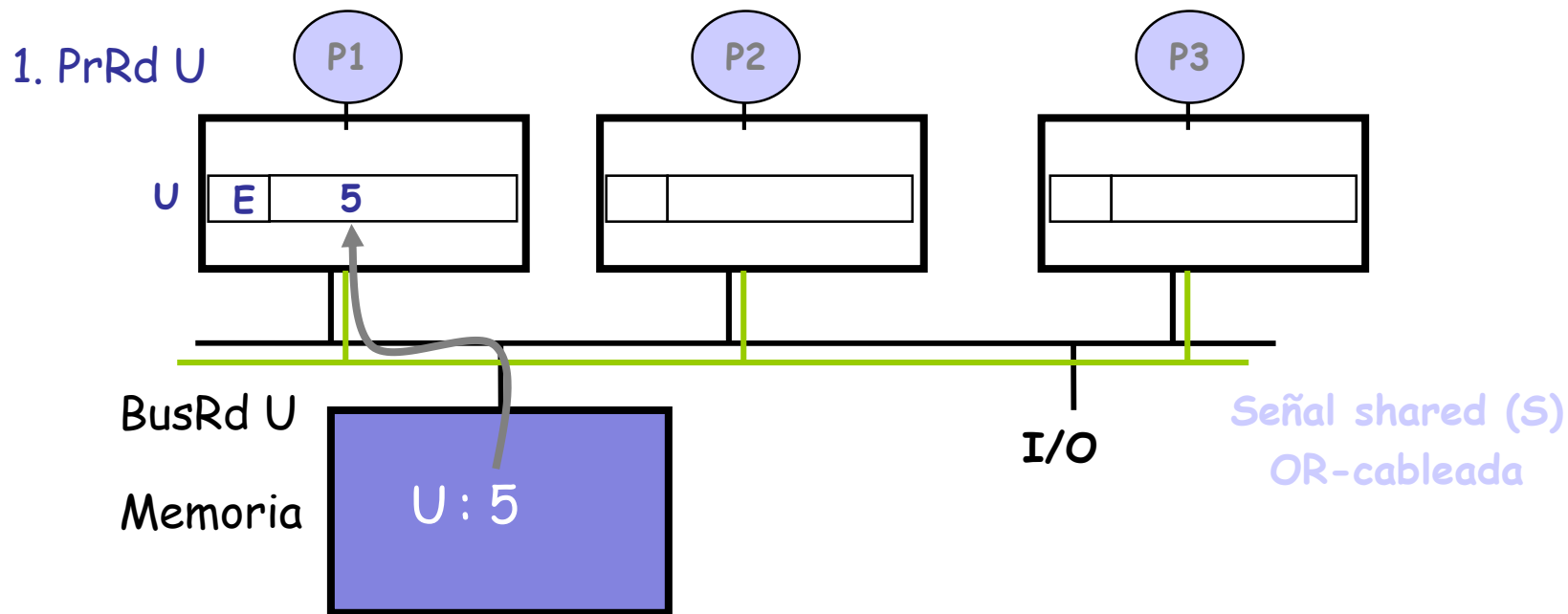
- o 4 Estados
 - (M) Modificado, (E) Exclusivo, (S) Compartido e (I) Inválido
 - Semántica similar a la del protocolo MSI
- o Nuevo estado (**E: Exclusive Clean o simplemente Exclusive**) indica que el bloque es la única copia pero no ha sido modificada: reduce tráfico en el bus
 - Nivel intermedio entre compartido y modificado
 - **Implica Exclusividad:** Puede pasarse a modificado sin realizar ninguna transacción (al contrario que Compartido)
 - **No implica pertenencia:** La memoria tiene una copia válida. El controlador de cache no debe proporcionar el bloque en respuesta a

Cartagena99

CLASES PARTICULARES, TUTORIAS TECNICAS ONLINE
LLAMA O ENVIA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Protocolo MESI: invalidación de 4 estados



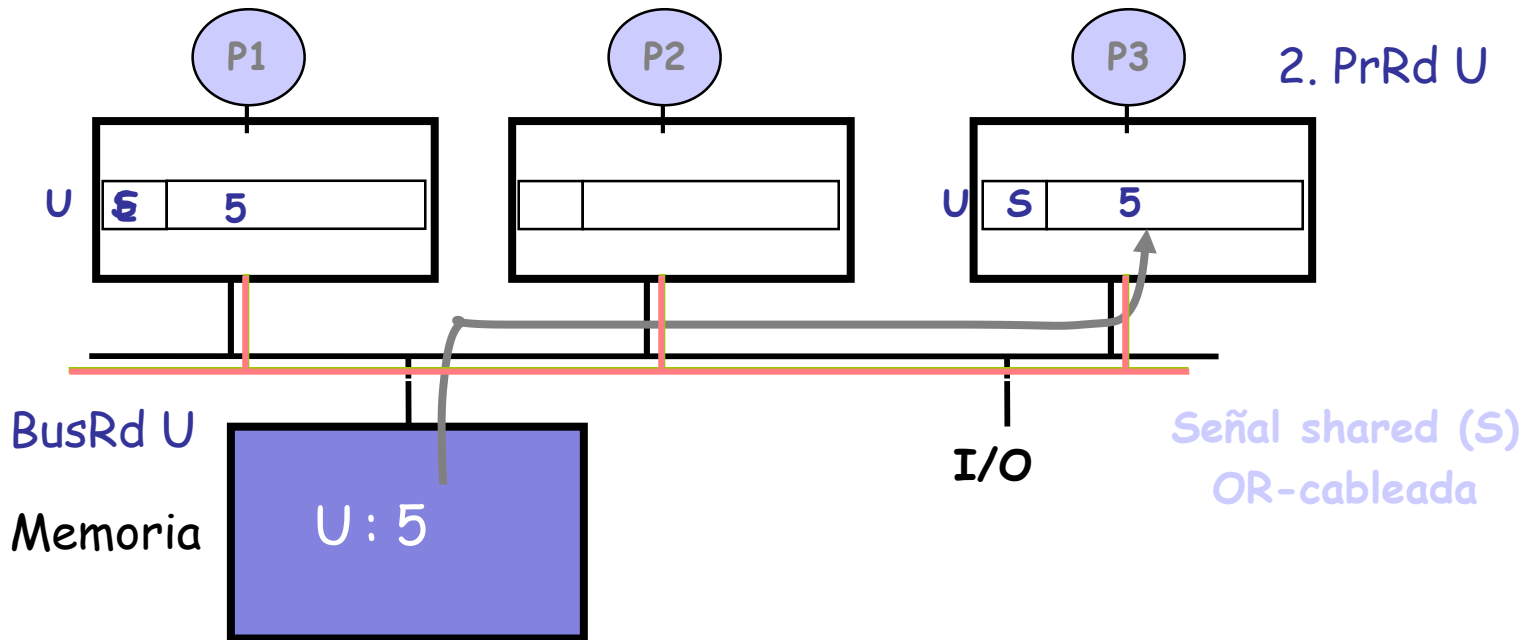
Operación	Estado P1	Estado P2	Estado P3	Transacción Bus	Datos Suministrados
P1 Lee U	E	—	—	BusRd	Memoria
P3 Lee U					

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Cartagena99

Protocolo MESI: invalidación de 4 estados



Operación	Estado P1	Estado P2	Estado P3	Transacción Bus	Datos Suministrados
P1 Lee U	E	—	—	BusRd	Memoria
P3 Lee U	S	—	S	BusRd	Memoria

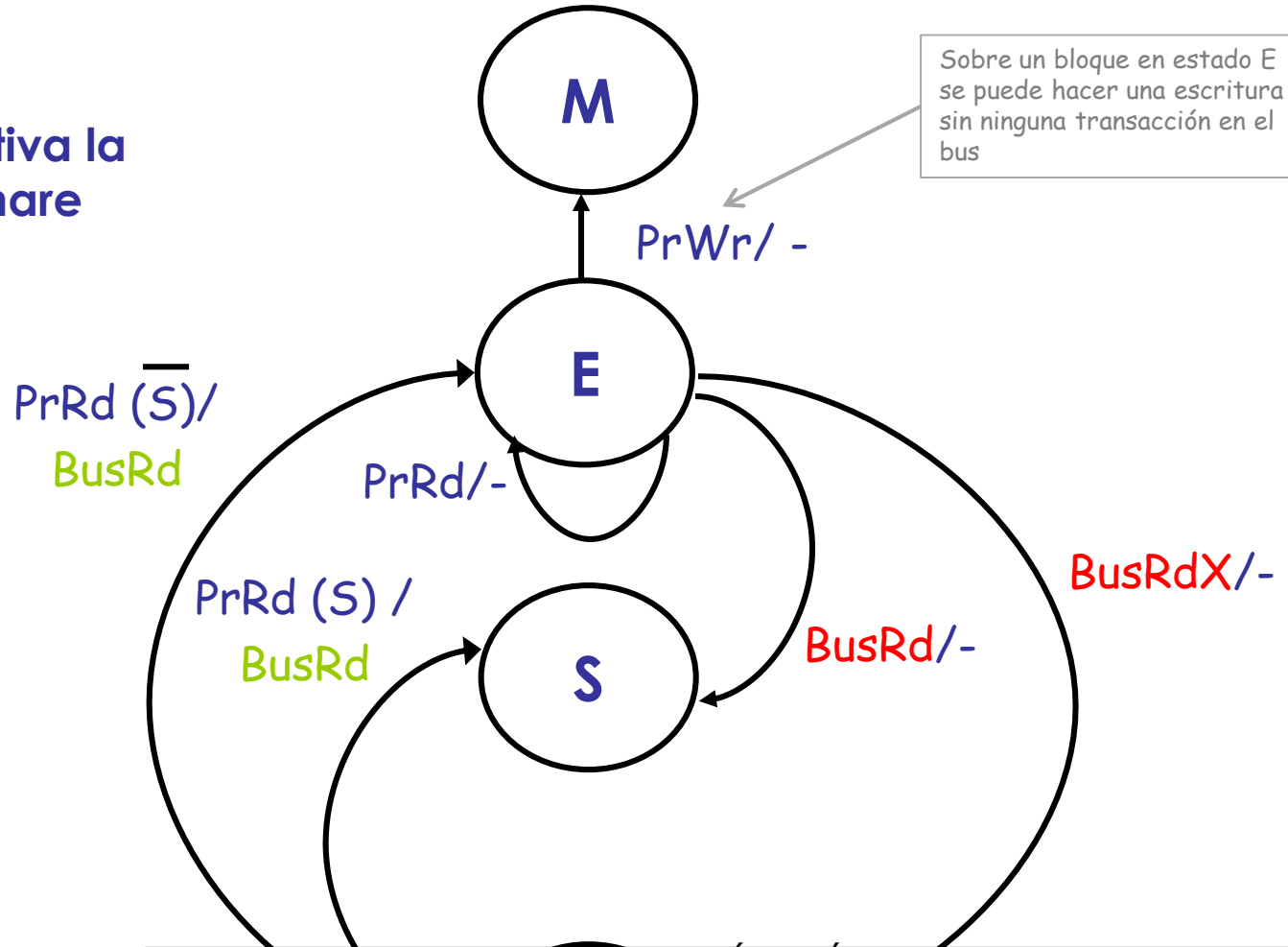
Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Protocolo MESI: invalidación de 4 estados

(S): se activa la señal share



Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVIA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP: 689 45 44 70

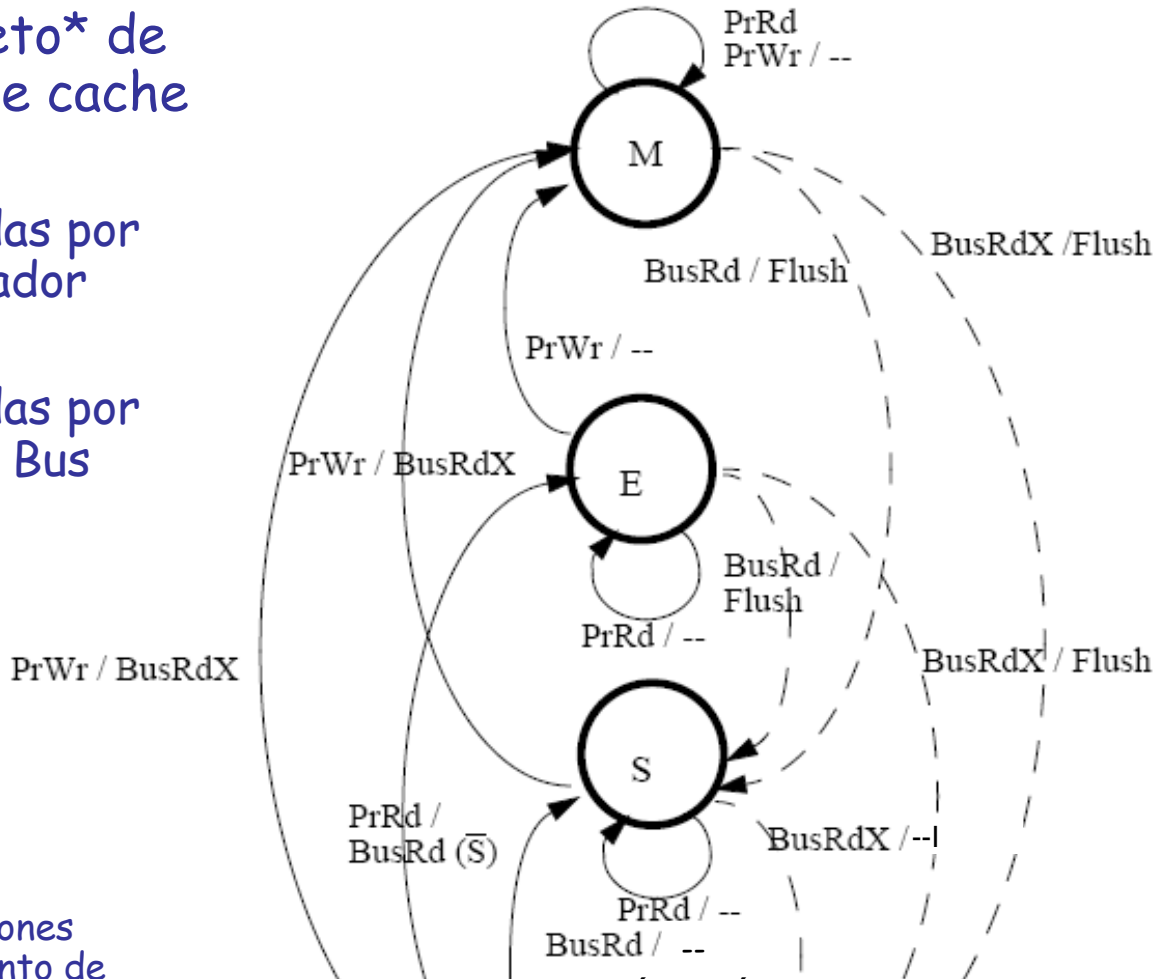
Protocolo MESI: invalidación de 4 estados

- Diagrama completo* de un controlador de cache

Transiciones producidas por peticiones del procesador



Transiciones producidas por eventos detectado en Bus



(*) No se incluyen las transiciones provocadas por reemplazamiento de

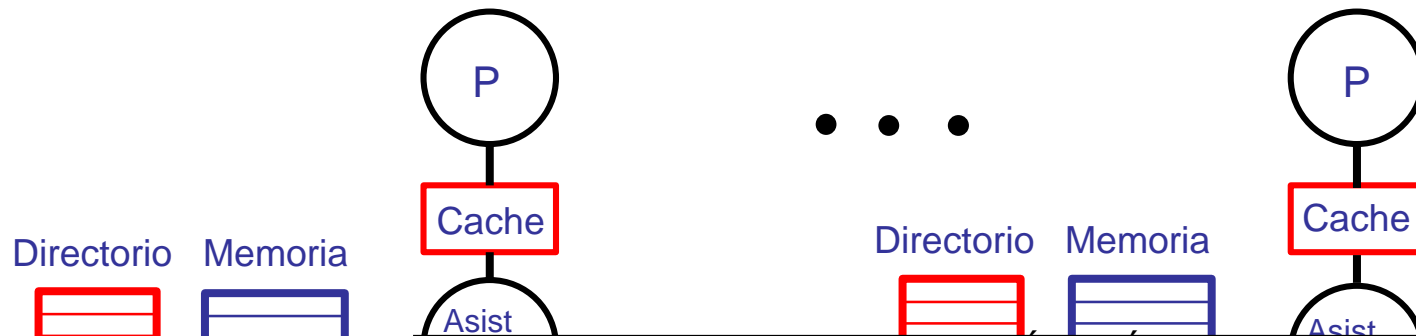
Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Coherencia basada en directorio

- ❑ Sistemas de memoria distribuida (arquitectura NUMA)
 - o Una zona de la memoria asociada a cada procesador
 - Pero... Cada procesador ve un espacio global de direcciones físicas compartido
 - o Los accesos locales o remotos los maneja el Asistente de Comunicaciones del nodo.
 - El procesador y el controlador de cache, no perciben si un fallo de cache se resuelve en su memoria local o es preciso acudir a un nodo remoto de la red
 - o Problema: coherencia de caches (igual que en snoopy).
 - Pero... **No hay un mecanismo para observar** las transacciones de los otros nodos
 - Una solución: llevar a la cache solo datos locales (ejemplo:T3E)
 - o Alternativa: Un directorío que mantiene información sobre el estado y localización en caches de cada bloque de memoria
 - El directorío podría ser un "cuello de botella": distribuir directorío.



CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Red de interconexión Escalable

Cartagena99

Coherencia basada en directorio

- ❑ Protocolo de directorio: se pueden utilizar un esquema similar a snoop (p. ej. con 3 estados: **MSI**)...
 - ... pero los cambios de estado son provocados por mensajes a través de la red.
- ❑ Estados de un bloque en la cache de un procesador P
 - **M**: Exclusivo (o Modificado).
 - La cache de P tiene la única copia válida del bloque. La Mp no está actualizada
 - **S**: Compartido (Shared).
 - La cache de P tiene una copia del bloque. Puede haber más copias en otras caches. La Mp está actualizada.
 - **I**: Inválido.
 - El bloque no está en la cache de P.

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Coherencia basada en directorio

- El directorio permite determinar para cada bloque de memoria:
 - o Si el bloque no se encuentra copiado en la cache de ningún procesador
 - o Si existe una o varias copias del bloque sin modificar (clean, Mp actualizada) en los procesadores (**sharers**). Registra la lista procesadores que tienen esas copias.
 - o Si un procesador (y uno solo) tiene una copia modificada del bloque (dirty, Mp no actualizada) y cuál es ese procesador (**owner**).
 - o Posible implementación: Para cada bloque de Mp
 - 1 bit por procesador (indica presencia/ausencia del bloque en el procesador)
 - 1 bit clean/dirty

- Tres tipos de procesadores implicados en una transacción
 - o **Peticionario** (local): El procesador que inicia la petición
 - o **Home**: El procesador en donde reside la zona de memoria que

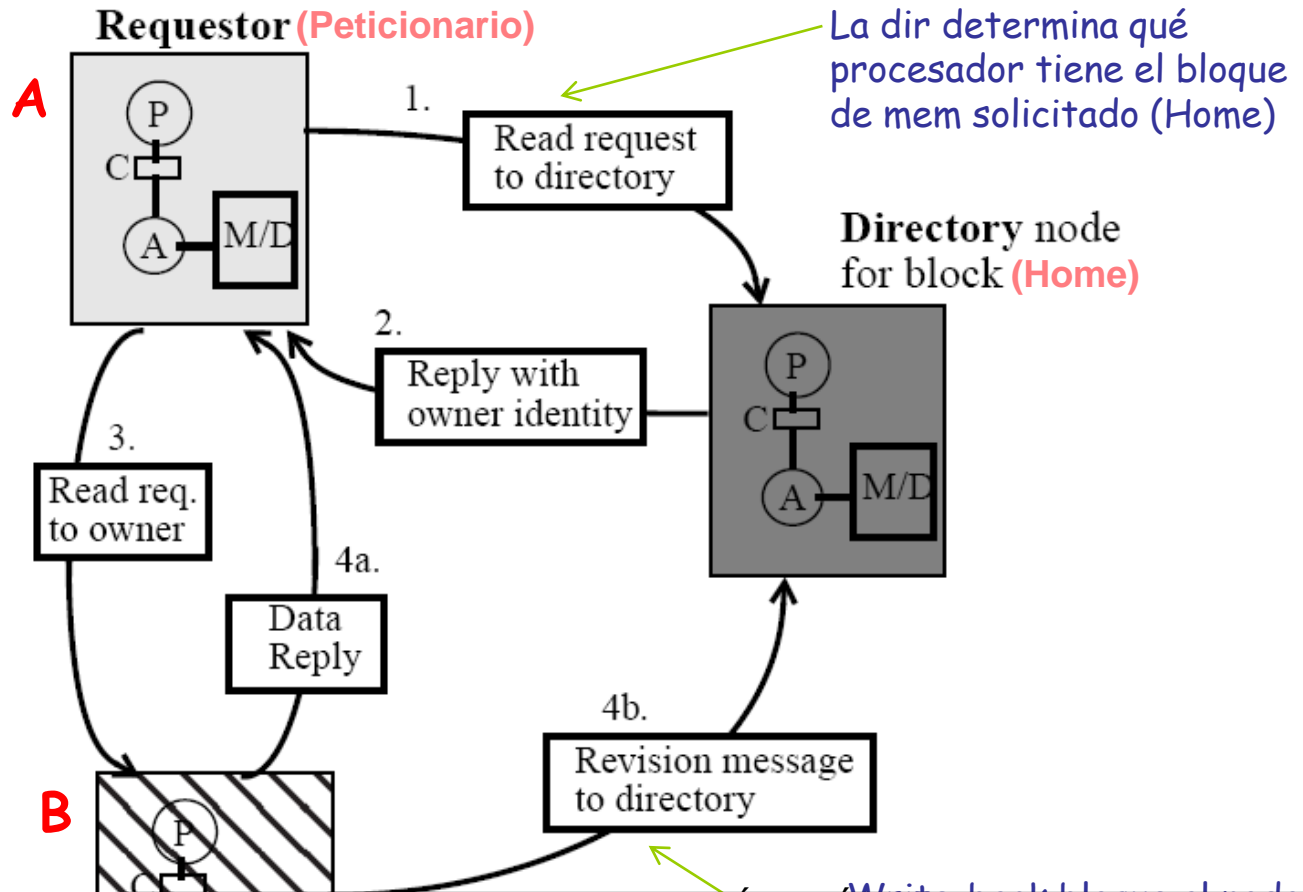
Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Ejemplo de funcionamiento de un directorio

❑ Fallo de lectura sobre un bloque sucio



CLASES PARTICULARES, TUTORIAS TÉCNICAS ONLINE
LLAMA O ENVIA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP: 689 45 44 70

dirty copy (Remoto, Owner)

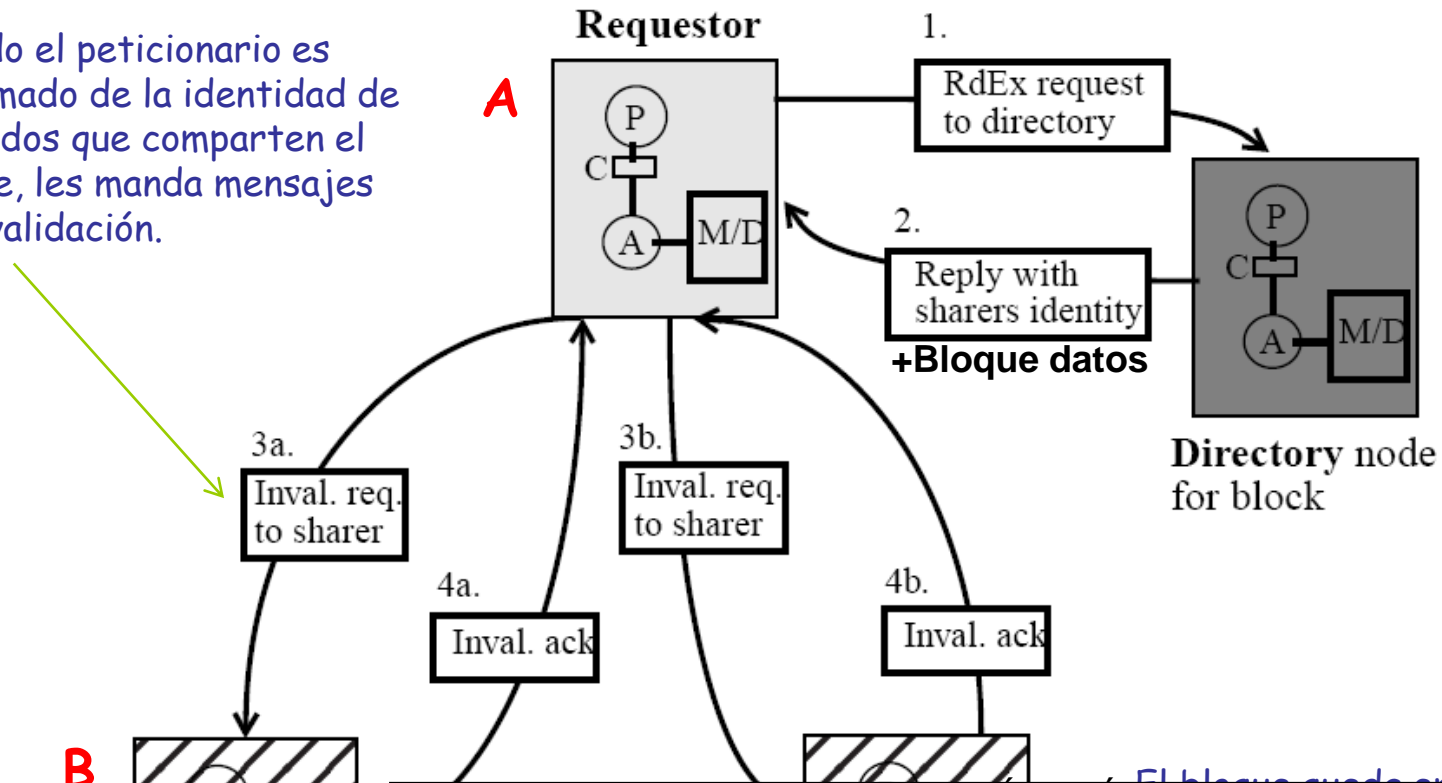
limpio"

Cartagena99

Ejemplo de funcionamiento de un directorio

- ❑ Fallo de escritura sobre un bloque compartido por dos procesadores (bloque limpio)

Cuando el peticionario es informado de la identidad de los nodos que comparten el bloque, les manda mensajes de invalidación.



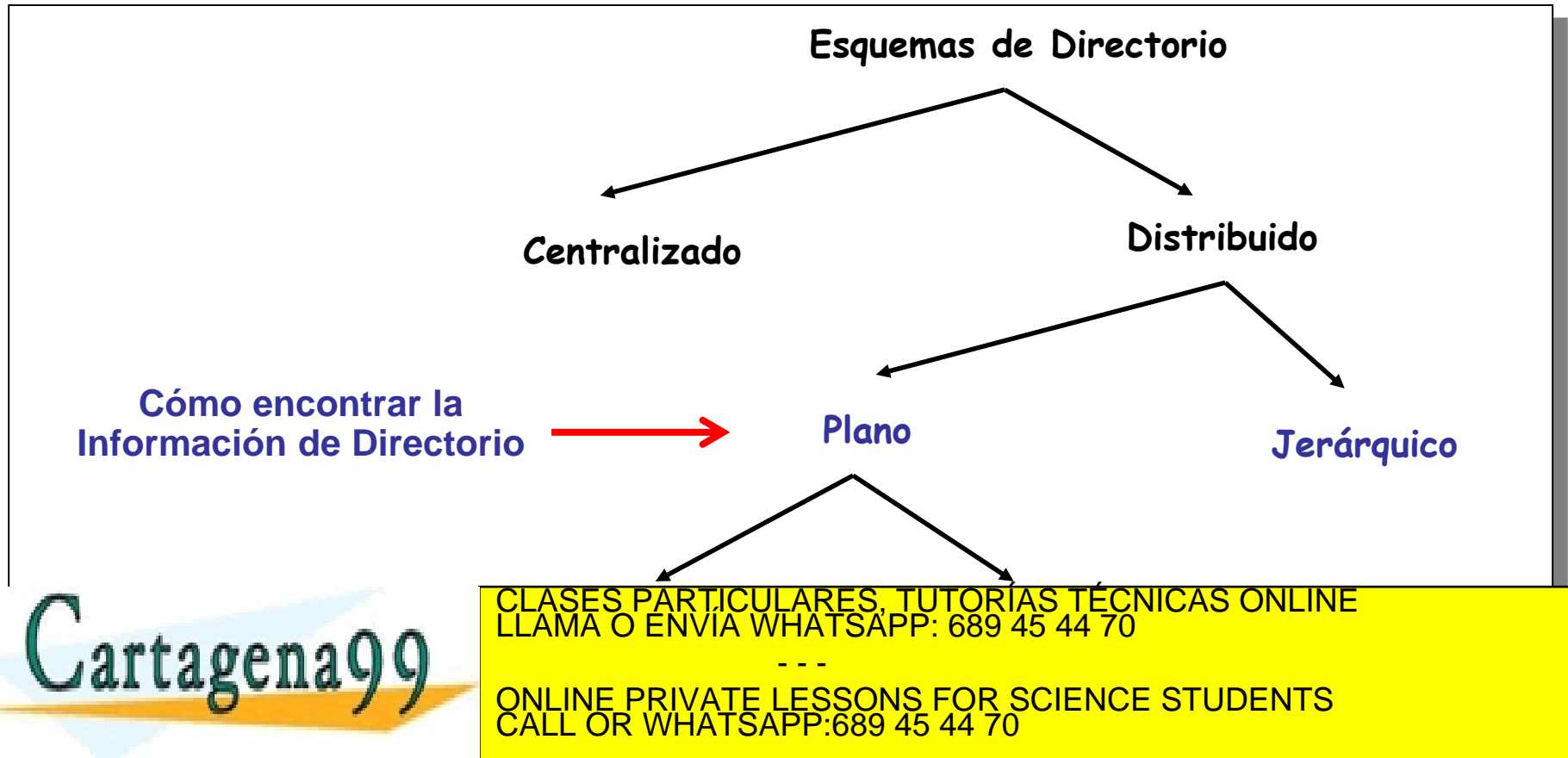
CLASES PARTICULARES, TUTORIAS TÉCNICAS ONLINE
LLAMA O ENVIA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Cartagena99

Organización del directorio

- Primeros Esquemas de Directorio aparecieron en Mainframes de IBM. La memoria (centralizada) mantenía copia de las etiquetas (tags) almacenadas en cada una de las caches (Directorio Centralizado)



Cartagena99

Organización del directorio

□ Esquema Plano

- o El origen de la información de directorio para un bloque se encuentra en un lugar fijo (nodo home/origen), que viene determinado directamente por la dirección del bloque (hashing)
- o Ante un fallo de cache se envía un transacción de red directamente al nodo origen para consultar el directorio

□ Esquema Plano Basado en Memoria

- o La información de directorio acerca de un determinado bloque esta almacenada en el nodo origen (home) de dicho bloque (Stanford DASH/FLASH, SGI Origin, MIT Alewife, HAL)

□ Esquema Plano Basado en Cache

- o La información de directorio acerca de un determinado bloque ~~no está totalmente almacenada en el nodo origen sino~~

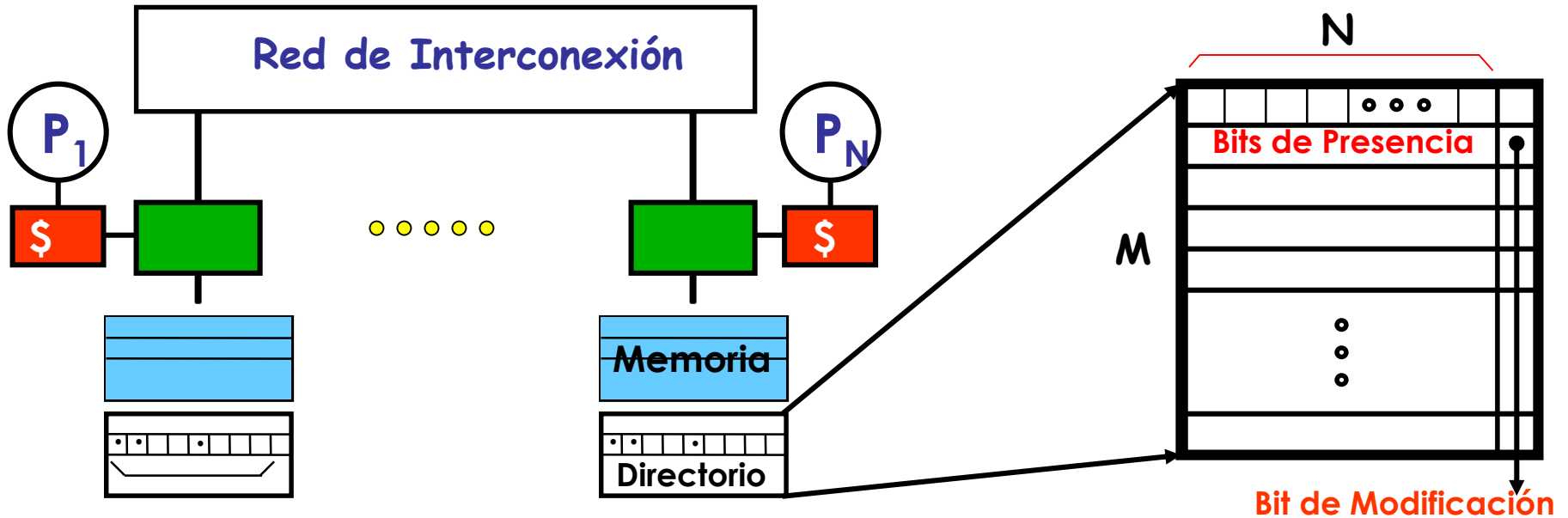
Cartagena99

CLASES PARTICULARES, TUTORIAS TÉCNICAS ONLINE
LLAMA O ENVIA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Directorio plano basado en memoria

- N nodos, M bloques de memoria por nodo.



Organización más natural: Full Bit Vector

- 1 bit de presencia por nodo (indican en qué nodos está copiado cada

Cartagena99

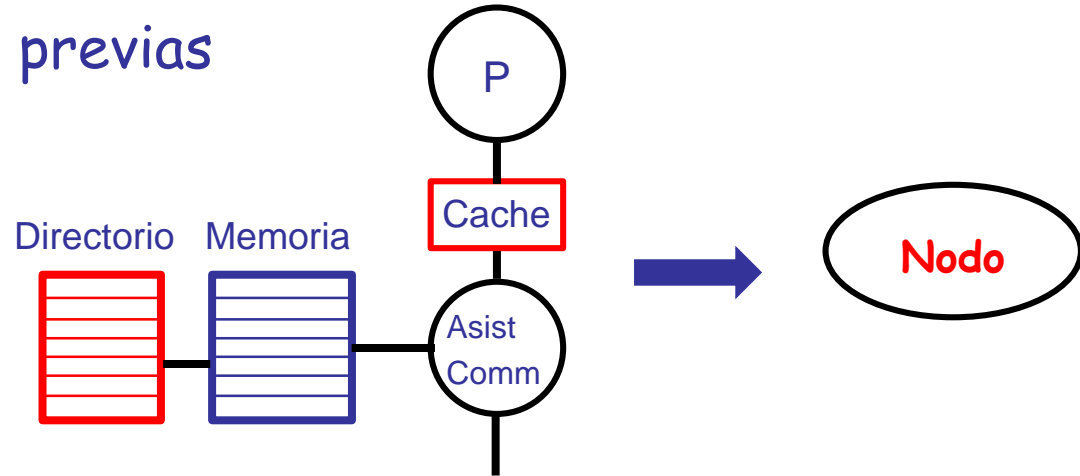
CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP: 689 45 44 70

Directorio plano basado en memoria: funcionamiento

□ Definiciones previas

o Nodo



- o **Nodo Peticionario (local)**: contiene el procesador que emite una petición del bloque
- o **Nodo Origen (home)**: nodo en cuya porción de memoria principal se encuentra el bloque (contiene el directorio)
- o **Nodo Sucio (dirty)**: tiene una copia del bloque en su cache en estado modificado
- o **Nodo Exclusivo (exclusive)**: tiene una copia del bloque en estado exclusivo (única copia de cache válida y el bloque ~~está actualizado en memoria~~)

Cartagena99

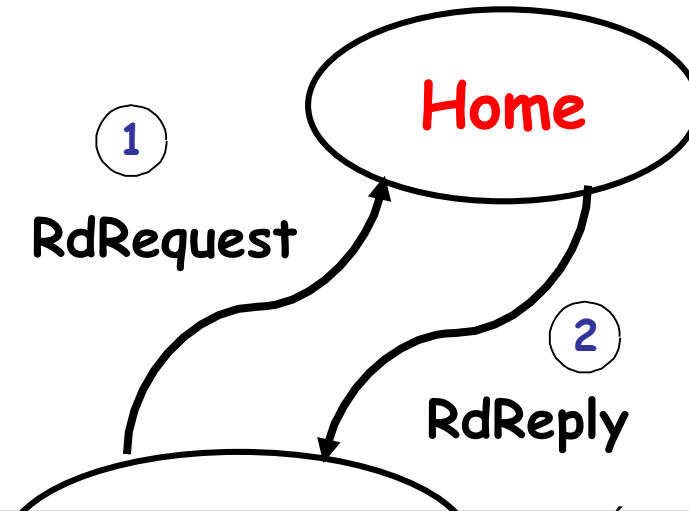
CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVIA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Directorio plano basado en memoria: funcionamiento

□ Obtener bloque para lectura

- o Caso 1: Dirty bit OFF (el bloque está actualizado en la memoria del nodo Home)
 - 1. El nodo local pide el bloque
 - 2. El Home lo proporciona (y actualiza el directorio)
 - Sólo son necesarias dos transacciones en la red



Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

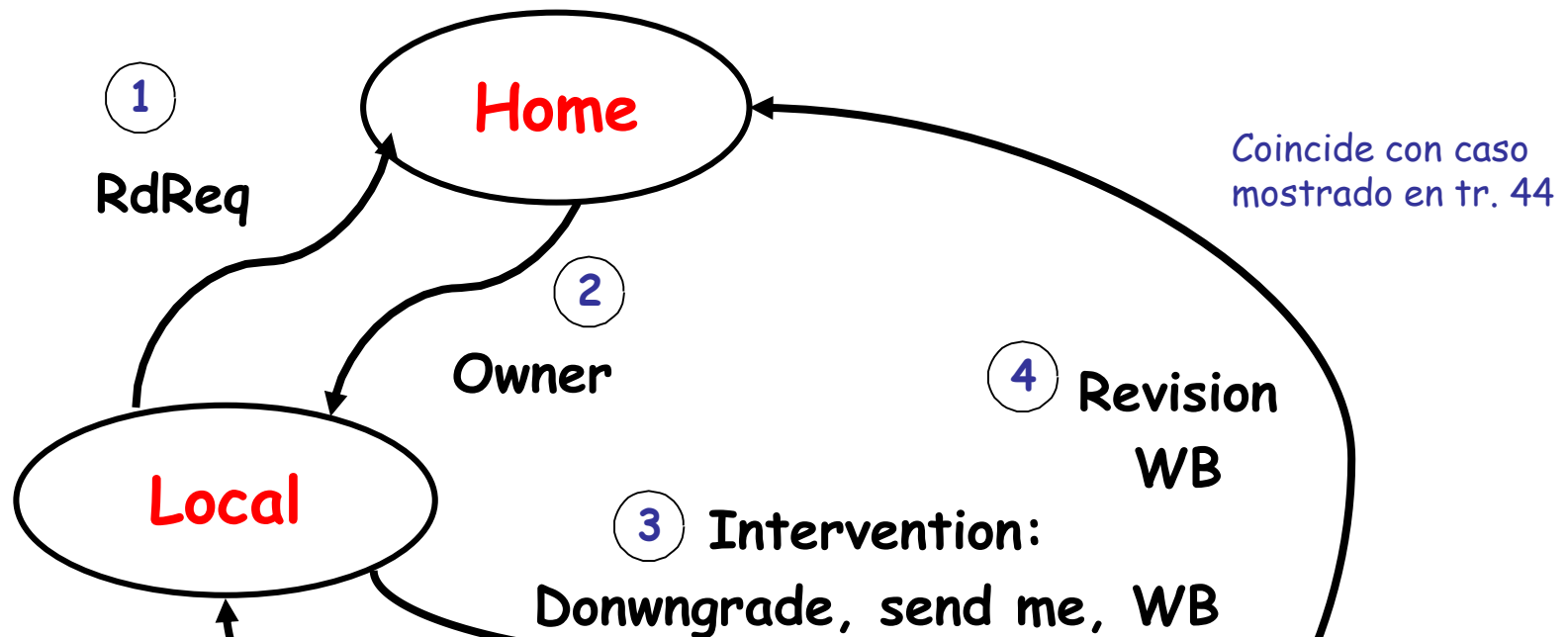
ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Directorio plano basado en memoria: funcionamiento

❑ Obtener bloque para lectura

o Caso 2: Dirty bit ON (El bloque no está actualizado en memoria del nodo Home)

- 1. El nodo local pide el bloque
- 2. El Home proporciona la identidad del nodo Propietario
- 3. El nodo local hace una petición de lectura al propietario
- 4. El nodo propietario proporciona el bloque y hace WB al Home



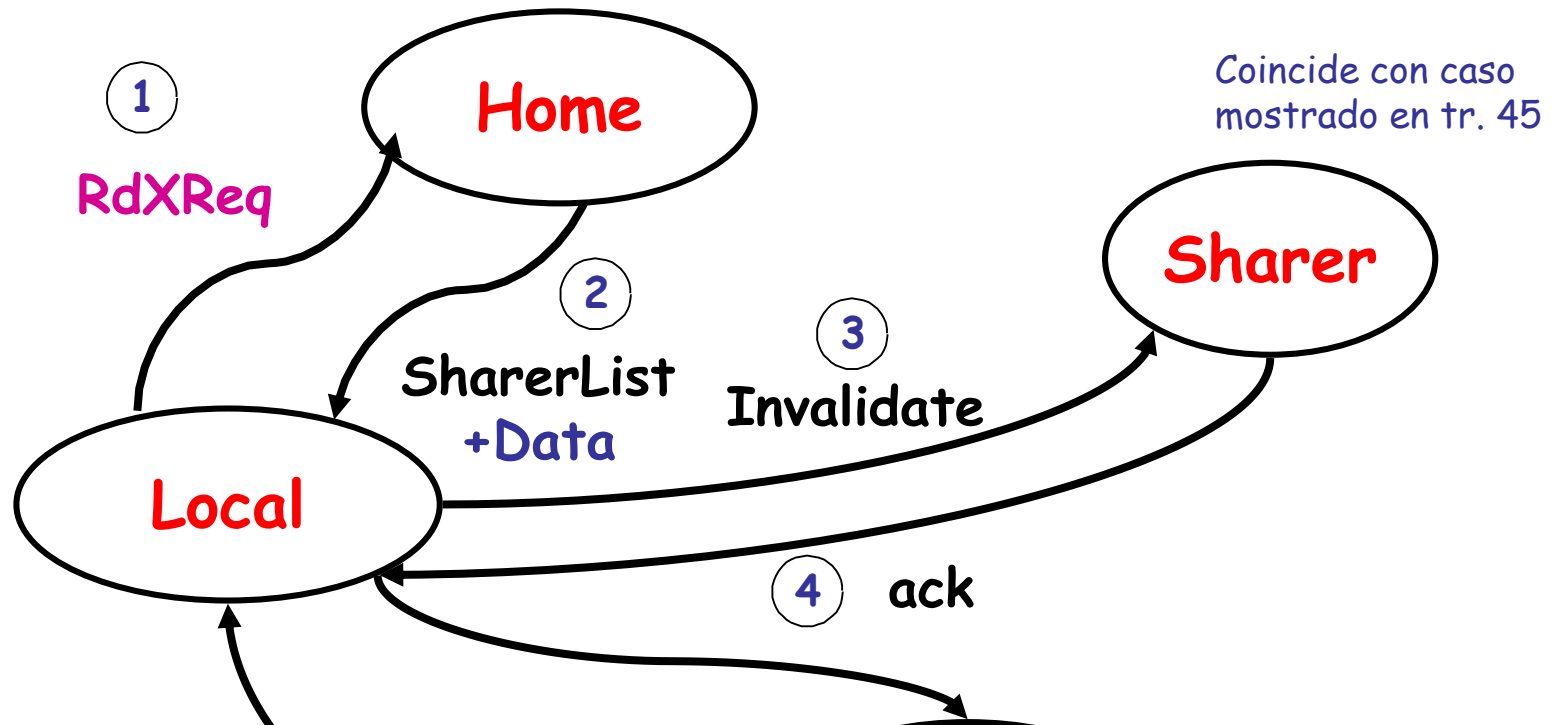
Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Directorio plano basado en memoria: funcionamiento

- ❑ Obtener bloque para escritura
 - o Caso 1: Dirty bit OFF (hay varias copias válidas. Mp actualizada)



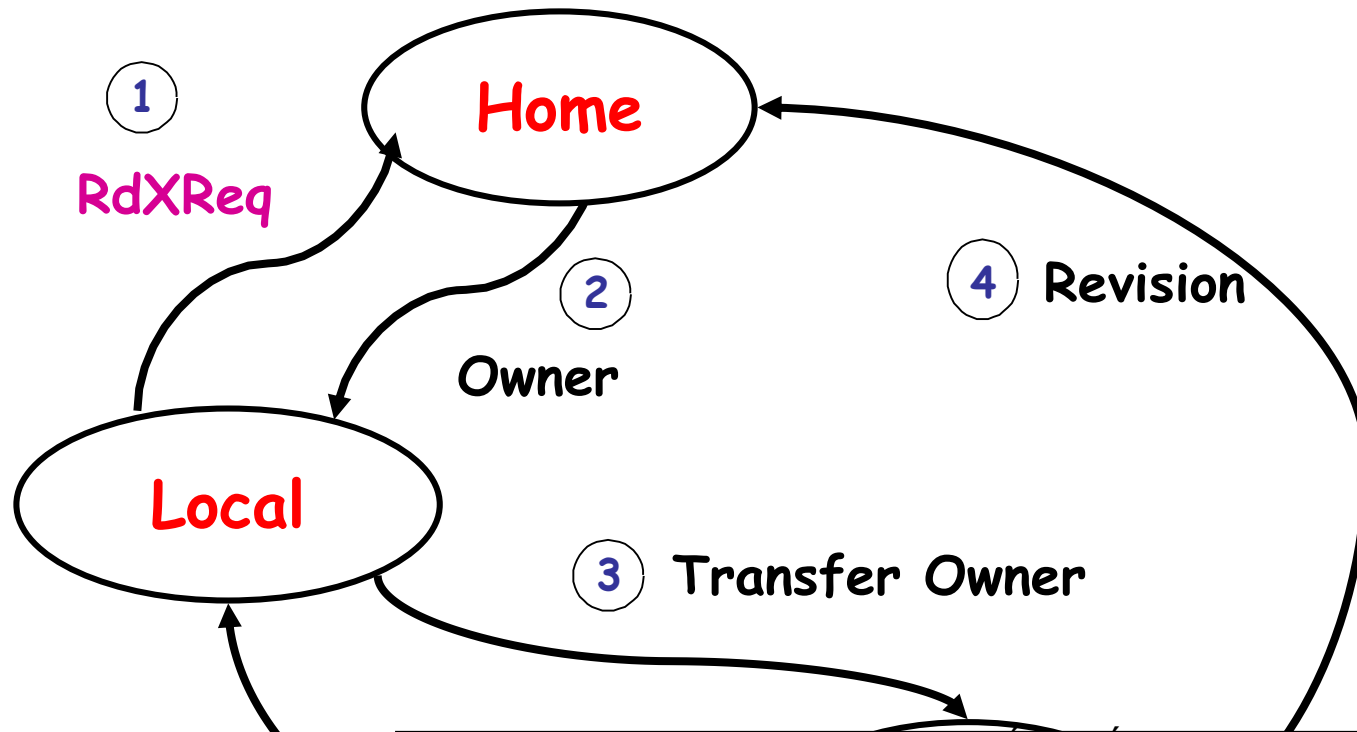
Cartagena99

CLASES PARTICULARES, TUTORIAS TÉCNICAS ONLINE
LLAMA O ENVIA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Directorio plano basado en memoria: funcionamiento

- ❑ Obtener bloque para escritura
 - o Caso 2: Dirty bit ON (hay una copia válida en el propietario. Mp no actualizada)



Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Directorio plano basado en memoria

- Rendimiento en las Escrituras (mensajes de invalidación)
 - o Número de mensajes (BW): proporcional al número de nodos que comparten el bloque
 - o Número de mensajes en el camino crítico (Latencia): La identidad de todos ellos está disponible en el nodo origen, los mensajes de invalidación se pueden enviar en paralelo
- Sobrecarga de almacenamiento:
 - o M bloques de memoria : Sobrecarga proporcional a $N_{proc} * M$
 - o Suponiendo tamaño de línea = 64 bytes
 - Con 64 nodos: sobrecarga del directorio 12.5%
 - Con 256 nodos: sobrecarga del directorio 50%
 - Con 1024 nodos: sobrecarga del directorio 200%
- Soluciones:
 - o Utilizar nodos multiprocesador (solo 1 bit por nodo) Ejemplo: 256 procesadores, 4 procesadores por nodo, línea de 128 B : Sobrecarga: 6.25%
 - o En lugar de almacenar 1 bit por nodo cada entrada del directorio,

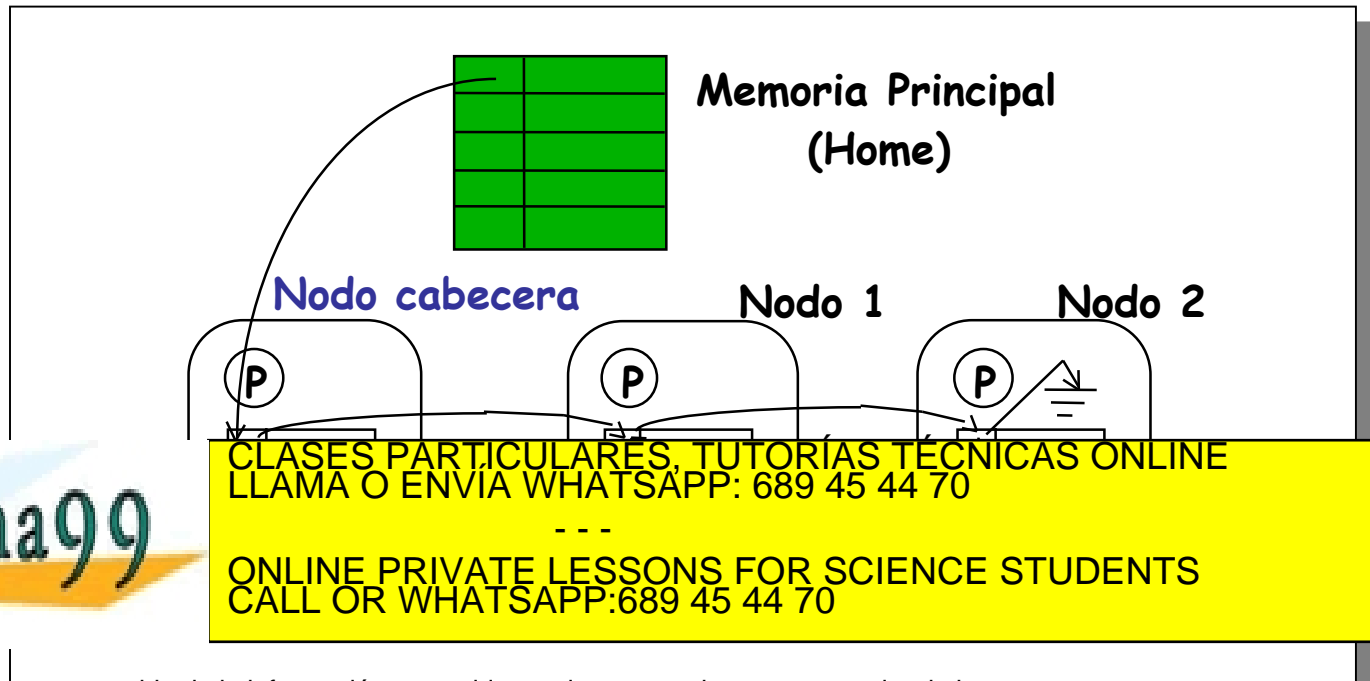
Cartagena99

CLASES PARTICULARES, TUTORIAS TECNICAS ONLINE
LLAMA O ENVIA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Directorio plano basado en cache

- Existe una memoria principal origen de cada bloque
 - o La entrada del directorio en el nodo origen tiene:
 - Un puntero al primero de los nodos (nodo cabecera) que tiene copia de dicho bloque
 - Bits de estado
 - o El resto de los nodos con copia se encuentra unidos mediante una lista enlazada distribuida (IEEE 1596-1992 Scalable Coherent Interface (SCI) : lista doblemente enlazada)



Cartagena99

Directorio plano basado en cache

❑ Fallo de lectura:

- o El nodo peticionario envía una petición al nodo origen de ese bloque para determinar la identidad del nodo cabecera
- o El nodo origen responde con la identidad del nodo cabecera
- o El nodo peticionario envía solicitud al nodo cabecera para que le inserte a la cabeza de la lista (se convierte en el nuevo nodo cabecera) : incrementa la latencia
- o Los datos son enviado por
 - Nodo origen si tiene copia
 - Nodo cabecera en caso contrario (siempre tiene la última copia)

❑ Eliminación (Reemplazamiento) e inserción en la lista

- o Añade bastante complejidad al protocolo: es necesario

~~llevar a cabo coordinación con el nodo anterior y posterior~~

CLASES PARTICULARES, TUTORIAS TECNICAS ONLINE
LLAMA O ENVIA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Cartagena99

Directorio plano basado en cache

□ Escritura:

- o El nodo peticionario puede encontrarse ya en la lista de nodos que comparten el bloque
- o Se recorre el resto de la lista para invalidar las sucesivas copias.
 - Los acuse de recibo de las invalidaciones se envían al nodo que realiza la escritura

□ Rendimiento en las Escrituras (mensajes de invalidación)

- o Número de mensajes (BW): proporcional al número de nodos que comparten el bloque (semejante a los basados en memoria), pero están distribuidos
- o Número de mensajes en el camino crítico (Latencia): también es proporcional al número de nodos que comparten

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Directorio plano basado en cache

- Ventajas respecto a los esquemas basados en memoria
 - o Menor sobrecarga de directorio.
 - Punteros a nodo cabecera:
 - Proporcional al número de bloques de memoria de la máquina
 - Punteros siguiente y anterior:
 - Proporcional al número de bloques de cache en la máquina (mucho menor que el número de bloques de memoria)
 - o El trabajo realizado por los controladores para enviar las invalidaciones no está centralizado en el nodo origen sino distribuido entre los nodos que comparten el bloque

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Sincronización: introducción

- ❑ Necesidad de garantizar acceso seguro de un proceso a variables compartidas
- ❑ Problema que aparece ya en uniprosesadores con multiprogramación (exclusión mutua, secciones críticas)
- ❑ Aspectos relevantes
 - o Primitiva hw básica: una instrucción no interrumpible (operación atómica) que **lee** y **actualiza** una cierta posición de memoria (cerrojo).
 - o Las operaciones de sincronización a nivel de usuario (rutinas sw de sincronización) se construyen usando esta primitiva hw.
 - o Para sistemas MPP las operaciones de sincronización pueden ser el cuello de botella del sistema
 - Hay un conjunto de procesos compitiendo por leer-y-modificar el cerrojo (adquisición del cerrojo).
 - **A diferencia del caso uniprosesador**, todos estos procesos pueden estar corriendo simultáneamente en los diferentes procesadores.
 - **Cada procesador puede hacer las lecturas del cerrojo sobre su cache local, sin generar tráfico en el bus.**
 - **Pero... al escribir genera tráfico en el sistema de comunicaciones**

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Sincronización: Primitivas hw básicas

- ❑ Intercambio (atomic exchange): Intercambia el valor de un registro y una posición de memoria (cerrojo).
 - o Convenio de interpretación de los valores del cerrojo
 - 0 => el cerrojo está libre
 - 1 => el cerrojo está bloqueado y no es accesible.

SI situación inicial:

Reg 1 Mem 0



Reg 0 Mem 1

al observar que Reg=0 tengo la confirmación de que el cerrojo estaba libre y lo he bloqueado

SI situación inicial:

Reg 1 Mem 1



Reg 1 Mem 1

al observar que Reg=1 tengo la confirmación de que el cerrojo estaba bloqueado por otro proceso y no lo he

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Sincronización: Primitivas hw básicas

❑ Otras primitivas

- o Test-and-set. Comprueba el valor de una variable y si está a cero, la pone a 1.
- o Fetch-and-increment. Incrementa el valor de una variable en memoria.

❑ Operaciones atómicas en multiprocesadores

- o Se requiere una lectura y una escritura en una sola instrucción no interrumpible
- o Complica la implementación de la coherencia: el hw no puede permitir otra op entre la lectura y la escritura, y además debe evitar el interbloqueo !!

❑ Sincronización con un par de operaciones

- o **Load Linked (LL):** Lee una dir de memoria, pero además guarda en un registro especial (link register) la dirección leída.
 - Si ocurre una **interrupción** (cambio de contexto), o si el bloque de cache que contiene la dir es invalidado por una **escritura**, el link register se borra
- o **Store Conditional (SC):** Almacena el contenido de un registro R en una dir de memoria, siempre que ésta coincida con la dir almacenada en el link register. En caso contrario no se hace el

Cartagena99

CLASES PARTICULARES, TUTORIAS TECNICAS ONLINE
LLAMA O ENVIA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Sincronización: Primitivas hw básicas

❑ Implementación de otras primitivas con LL+SC

❑ Exchange. Ej.: intercambio $R4 \leftrightarrow M[0+(R1)]$

```
try:    MOV    R3,R4        ;move exchange value to R3
        LL     R2,0(R1)    ;load linked
        SC     R3,0(R1)    ;store conditional
        BEQZ   R3,try      ;branch if store fails
        MOV    R4,R2      ;success: put load value in R4
```

❑ Fetch-and-increment. Ej.: Incrementa atómicamente $M[0+(R1)]$

```
try:    LL     R2,0(R1)    ;load linked
        DADDUI R3,R2,#1    ;increment (*)
        SC     R3,0(R1)    ;store conditional
        BEQZ   R3,try      ;branch if store fails
```

Cartagena99

CLASES PARTICULARES, TUTORIAS TECNICAS ONLINE
LLAMA O ENVIA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Implementación de spin locks

- ❑ Spin lock: bucle en el que un proceso permanece intentando conseguir un cerrojo hasta que lo consigue.
PROBLEMA: Si varios procesadores → Aumento de tráfico!!
- ❑ Implementación simple sin coherencia cache: mantener las variables cerrojo en memoria.
 - o Sup la variable cerrojo está en $M[0+(R1)]$

	DADDUI	R2,R0,#1
lockit:	EXCH	R2,0(R1) ;atomic exchange
	BNEZ	R2,lockit ;already locked?

- o Los accesos a mem implican tráfico en el sistema de comunicaciones → cuello de botella cuando varios procesadores compiten por el cerrojo.

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Implementación de spin locks

- Implmentación con coherencia cache.
 - o Hacer una consulta preliminar del cerrojo, antes de hacer el exchange atómico. (test...exchange)
 - o Un procesador permanece leyendo su copia local mientras detecta que el cerrojo está bloqueado.
 - o Cuando lee en su cache que el cerrojo está libre (cómo se entera?), ejecuta un EXCH y compite con los demás procesadores que estén haciendo el mismo spin lock. Uno de ellos consigue el bloqueo del cerrojo.
 - o El ganador verá un 0 en el registro de la instr EXCH, los demás verán un 1.
 - o Evita invalidaciones inútiles cuando un procesador está ejecutando repetidamente el bucle de espera.

```
lockit:   LD      R2,0(R1) ;preliminary load of lock
          BNEZ   R2,lockit ;test 1: if not available → spin
          DADDUI R2,R0,#1 ;R2 = 1
          EXCH  R2,0(R1) ;M[0+(R1)] ↔ R2
          BNEZ   R2,lockit ;test 2: if lock wasn't 0 → spin
```

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVIA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Cartagena99

Implementación de spin locks

□ Ejemplo: uso del código precedente en 3 procesadores

Paso	P0	P1	P2	Estado coherencia	Bus/actividad directorio	
1	Ha adquirido el lock (lock = 1)	No copia de lock en cache. Lectura lock → fallo. Se sirve el fallo y pasa a compartido. Chequea si lock = 0 (test1)	No copia de lock en cache. Lectura lock → fallo. Se sirve el fallo y pasa a compartido. Chequea si lock = 0 (test1)	P0: M P1: I P2: I	P0: S P1: S P2: S	Fallos de cache en P1 y P2. Estado lock pasa a shared
2	Libera lock (lock=0). Acierto de escritura. Invalida P1 y P2	Recibe invalidación	Recibe invalidación	P0: M P1: I P2: I		BusRdx de P0 para pasar a M. P1 y P2 ven RdX y pasan a invalidado
3		Fallo de cache (BusRd)	Fallo de cache (BusRd)	P0: S P1: I P2: S		Bus sirve primero fallo en P2 (flush de P0) y pasa a compartido en P0 y P2
4		Espera que se libere el bus para poder satisfacer su fallo	Chequea si lock=0 (test1), siendo el test positivo	P0: S P1: I P2: S		Fallo en P2 satisfecho, en P1 todavía no
5		Chequea si lock=0 (test1), siendo el test positivo	Como test1 positivo, ejecuta exchange	P0: S P1: S P2: S		Se satisface el fallo de P1
6		Como test1 positivo, ejecuta exchange	Resultado Exchange: 0 (éxito en test2) y pone lock =1 (adquiere el cerrojo)	P0: I P1: I P2: M		P2 escribe sobre bloque compartido (RdEx). Lock exclusivo de P2

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP: 689 45 44 70

Implementación de spin locks

- Implementación usando LL+SC (evitar el uso de lectura-modificación-escritura atómica)

lockit:	LL	R2,0(R1) ;load linked of lock (1)
	BNEZ	R2,lockit ;not available-spin (2)
	DADDUI	R2,R0,#1 ;R2=1
	SC	R2,0(R1) ;store conditional
	BEQZ	R2,lockit ;branch if store fails (3)

- (1) El LL no tiene por qué generar tráfico en el sistema de comunicaciones
- (2) Si $R2 \neq 0 \rightarrow$ el cerrojo está ocupado
- (3) Si $R2 = 0 \rightarrow$ el cerrojo estaba libre cuando se ejecutó el LL, pero el SC ha fallado. Ha habido una interrupción, ha

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVIA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Consistencia de memoria

- ❑ ¿Qué aporta la coherencia?
 - o Las escrituras sobre una posición se hacen visibles a los lectores en el mismo orden
 - o Pero... no dice nada sobre el momento en que una escritura particular se hará visible para un lector concreto

- ❑ En un programa paralelo interesa frecuentemente que una lectura devuelva el resultado de una escritura particular

- ❑ **Ejemplo:** Consideramos 2 procesos, P1 y P2, ejecutándose en diferentes procesadores. P2 debe mostrar el valor de una variable compartida, A, después de que haya sido actualizada por P1.

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Consistencia de memoria: ejemplo

❑ Sincronización mediante Flag

P1	P2
/*Valor inicial de A y de flag = 0*/	
A = 1;	while (flag == 0); /*espera activa*/
flag = 1;	print A;

- ❑ P1 almacena un nuevo valor de A, P2 debe leer el nuevo valor e imprimirlo.
 - o En teoría la variable flag forzaría este comportamiento
- ❑ El programador está asumiendo que el resultado de "print A" debe ser 1, es decir:
 - o La escritura de la variable A se hace visible al P2 antes que la escritura de flag
 - o La lectura de flag que hace que salgamos del bucle de espera se ha completado antes de la lectura de A
 - o Pero... Esta ordenación entre los accesos de P1 y P2 no está garantizada por la coherencia
- ❑ La coherencia exige que el valor de A se haga visible a P2, pero no necesariamente antes que se haga visible el nuevo valor de flag
 - o Luego, la coherencia por si sola no evita que el resultado de "print A" sea 0

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Consistencia de memoria: ejemplo

- ❑ No es preciso que exista sincronización explícita de procesos para tener comportamientos que resulten extraños desde el punto de vista del programador
- ❑ Sea el código:

P1	P2
/*Valor inicial de A y B = 0*/	
(1a) A = 1;	(2a) print B;
(1b) B = 2;	(2b) print A;

- o Los accesos a Memoria hechos por los procesos P1 y P2 son escrituras normales
- o A y B no se usan como variables de sincronización
- o La intuición nos hace esperar que si "print B" imprime un 2, entonces "print A" tiene que imprimir un 1
- o Pero... 2a y 2b leen diferentes direcciones de memoria. Y la Coherencia no dice nada sobre el orden en que las escrituras de P1 sobre esas variables se hace visible a P2

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Consistencia de memoria

- ❑ Necesitamos algo más que la coherencia para dar a un espacio de direcciones compartido una semántica clara
- ❑ Es necesario un modelo de ordenación que los programadores puedan usar para razonar acerca de la corrección de los programas
- ❑ **Modelo de Consistencia de la Memoria** para un espacio de direcciones compartido: especifica las **restricciones en el orden** en el que las **operaciones de memoria (emitidas por un único o por diferentes procesos) deben hacerse visibles** a los procesadores.
 - o Incluye operaciones a las mismas o a diferentes posiciones de memoria
→ La Consistencia incluye a la Coherencia
- ❑ **Contrato entre el programador y el diseñador del sistema**
 - o Los programadores se basan en el modelo de consistencia para razonar acerca de los posibles resultados (corrección de los programas)

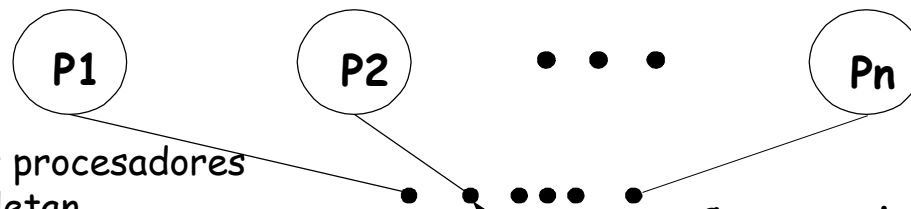
Cartagena99

CLASES PARTICULARES, TUTORIAS TECNICAS ONLINE
LLAMA O ENVIA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Consistencia secuencial

- ❑ **Objetivo:** Nos gustaría que el resultado de ejecutar un programa que usa varios procesos (threads) sea el mismo independientemente de si los procesos se ejecutan en paralelo (multiprocesadores) o de forma entrelazada (uniprocador multiprogramado)
- ❑ **Consistencia Secuencial (Lampport 1979)**
 - o Un multiprocesador es secuencialmente consistente si el resultado de la ejecución de cualquier programa es el mismo que el que se obtendría si:
 - Las operaciones de todos los procesadores se realizaran según **algún orden secuencial (orden total)** -> **cada entrelazamiento define un orden total**
 - Las operaciones de cada procesador individual ocurren en esta secuencia en el **orden especificado por el programa (orden parcial)**
- ❑ **Abstracción para el programador del subsistema de memoria bajo el modelo de consistencia secuencial**
 - o Parece que los procesadores comparten una única memoria (aunque realmente ésta se halle distribuida ente los procesadores, cada uno con su cache)



Cartagena99

CLASES PARTICULARES, TUTORIAS TECNICAS ONLINE
LLAMA O ENVIA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Consistencia secuencial

- ❑ No es realmente importante en qué orden se emitan de hecho las operaciones a la memoria o incluso cuándo se completen. **Lo que importa es que parezcan** completarse en un orden que no viole la consistencia secuencial
- ❑ Ejemplo

P1	P2
/*Valor inicial de A y B = 0*/	
(1a) A = 1;	(2a) print B;
(1b) B = 2;	(2b) print A;

- ❑ Resultados posibles bajo CS:
 - o Orden 2a, 2b, 1a, 1b produce $(A,B) = (0,0)$
 - o Orden 1a, 2a, 1b, 2b produce $(A,B) = (1,0)$
 - o Orden 1a, 1b, 2a, 2b produce $(A,B) = (1,2)$
- ❑ Es posible el resultado $(A,B)=(0,2)$ bajo consistencia secuencial?
 - o El orden del programa implica $1a \rightarrow 1b$ y $2a \rightarrow 2b$
 - o $A = 0$ implica $2b \rightarrow 1a$, lo que implica $2a \rightarrow 1b$
 - o $B = 2$ implica $1b \rightarrow 2a$ **contradicción**

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Consistencia secuencial

- ❑ Para satisfacer la Consistencia Secuencial el sistema (hardware, software) debe cumplir **dos restricciones**:
 - **Orden del programa**:
 - Las operaciones de memoria de un proceso deben hacerse visibles (tanto a sí mismo como a los otros procesos) en el orden del programa
 - **Atomicidad**:
 - Antes de emitirse la siguiente operación de memoria de acuerdo con el hipotético orden total (independientemente del proceso que lance dicha operación), es necesario que la operación en curso se haya completado (hecho visible) respecto a todos los procesos
 - **Dificultad: Atomicidad de escrituras**

- ❑ **Coherencia: propagación + serialización de escrituras**
 - **Serialización**: Todas las escrituras a una posición de memoria deben verse en el mismo orden por todos los procesadores

- ❑ **Consistencia Secuencial: Atomicidad de escrituras**
 - **Atomicidad**: Todas las escrituras (a cualquier posición de memoria) deben verse en el mismo orden por todos los

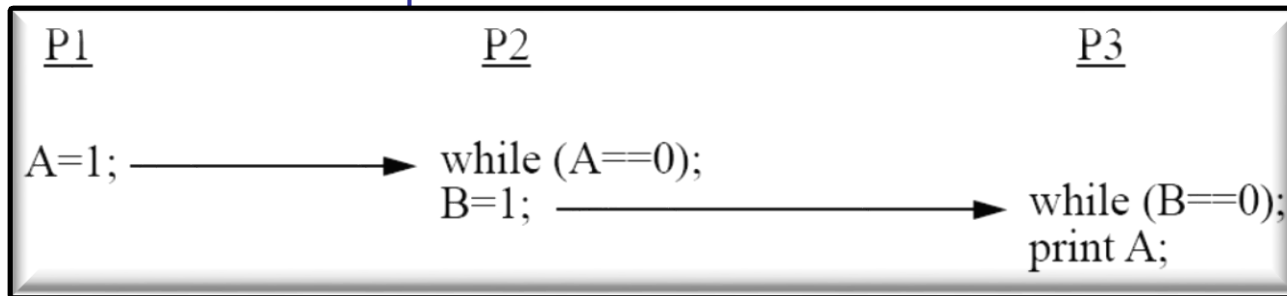
Cartagena99

CLASES PARTICULARES, TUTORIAS TECNICAS ONLINE
LLAMA O ENVIA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Consistencia secuencial

- ❑ La atomicidad de las escrituras es la clave que garantiza que ninguna operación realizada por procesador P2 después de haber visto el nuevo valor producido por una escritura hecha por P1, es visible para otro procesador P3 hasta que P3 ha visto también la escritura hecha por P1
 - o O sea, antes de que un nuevo valor sea visible, la anterior escritura tiene que ser visible para todos los procesadores.
- ❑ Ejemplo: si no se preserva la atomicidad de escrituras de puede violar la consistencia secuencial. Sup: inicialmente A=B=0.



- o P2 espera hasta que A vale 1; luego asigna el valor 1 a B
- o Como P3 espera hasta que B vale 1 y luego lee A y lo imprime, el valor impreso de A debería ser 1

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Consistencia secuencial

- **Condiciones Suficientes** para preservar la consistencia secuencial (Dubois y Scheurich 1987) :
 - o **(Condición Local)** Todo proceso **emite** las peticiones a memoria en el **orden especificado por el programa**
 - o **(Condición Local)** Después de **emitir** una operación de **escritura**, el proceso que la emitió **espera** antes de emitir su siguiente operación a que se haya **completado dicha escritura**
 - o **(Condición Global)** Después de **emitir** una operación de **lectura**, el proceso que la emitió **espera** antes de emitir su siguiente operación a que se haya **completado:**
 - **dicha lectura**
 - la **escritura** que generó el valor devuelto por la lectura (no sólo debe haberse completado con respecto al proceso que realiza la lectura sino también con respecto a todos los procesos)

Cartagena99

CLASES PARTICULARES, TUTORIAS TECNICAS ONLINE
LLAMA O ENVIA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70