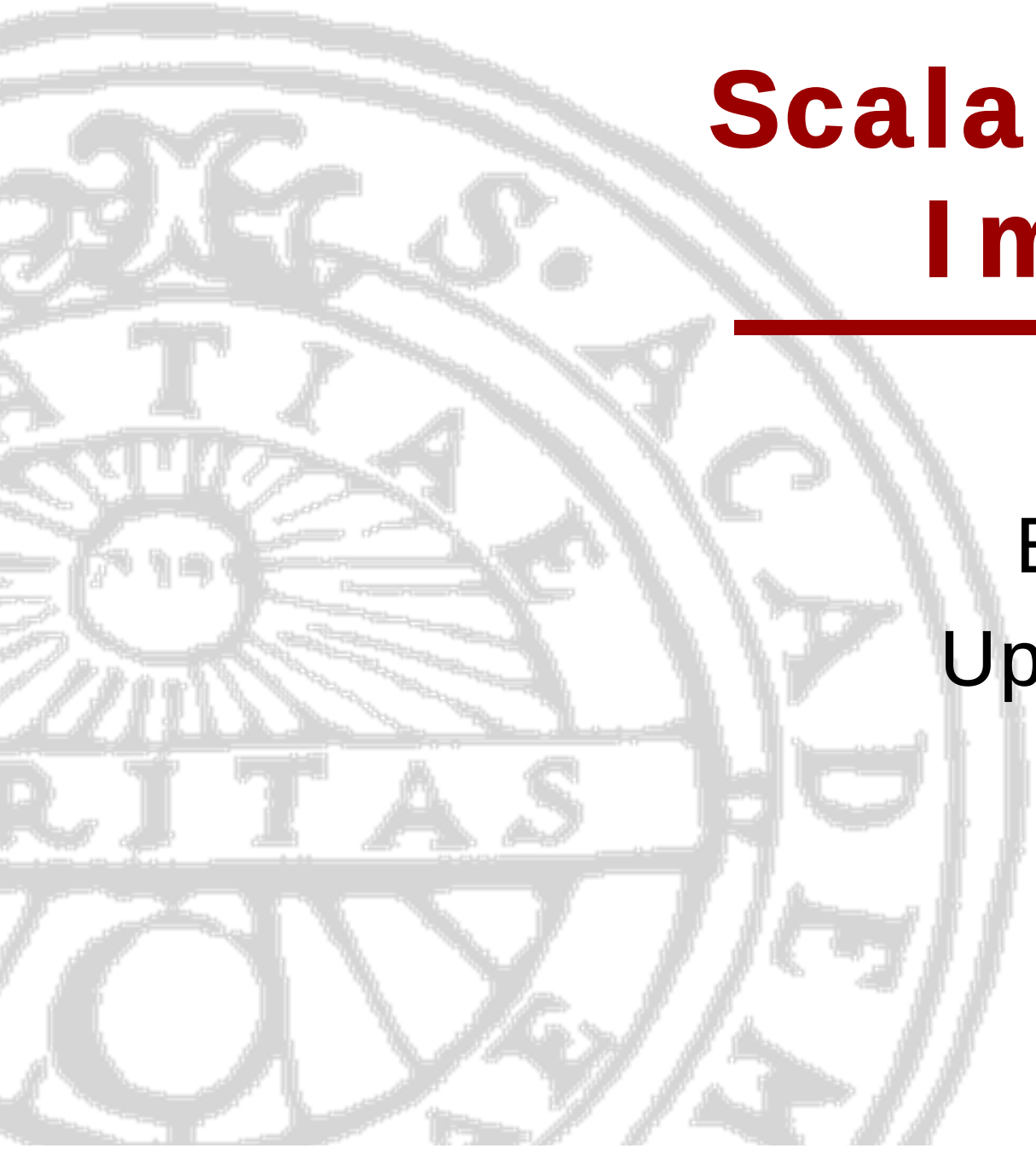
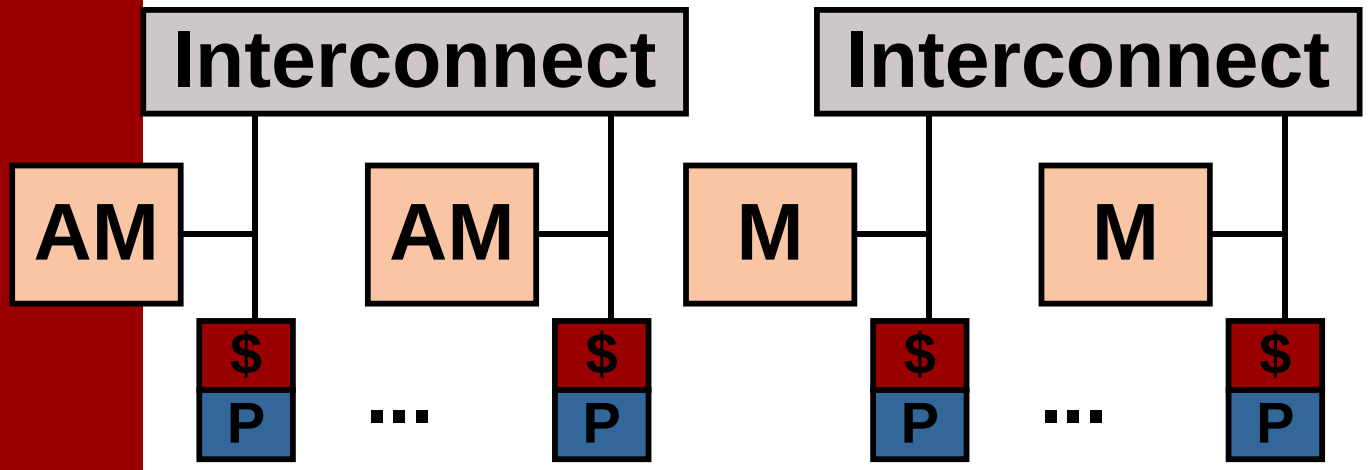


Scalable Coherence Implementation

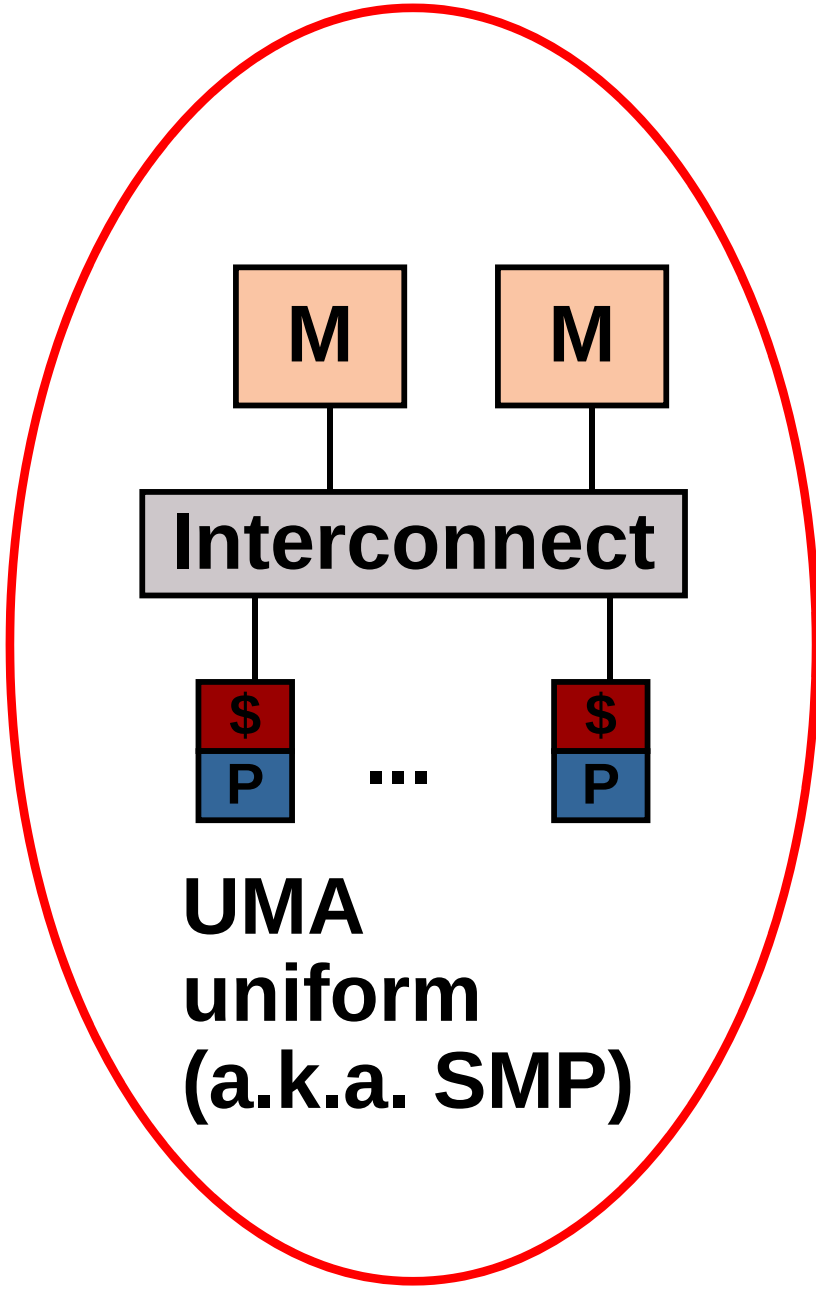
Erik Hagersten
Uppsala University
Sweden

Three options for scalable shared memory



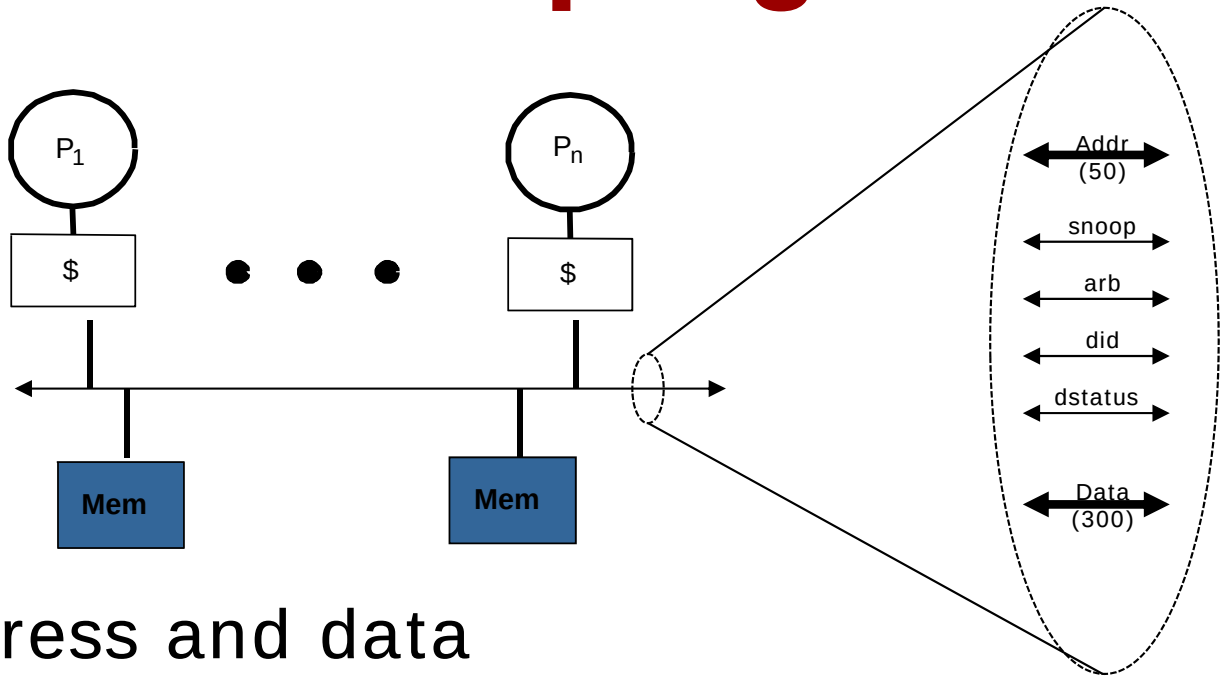
COMA
cache-only

NUMA
non-uniform



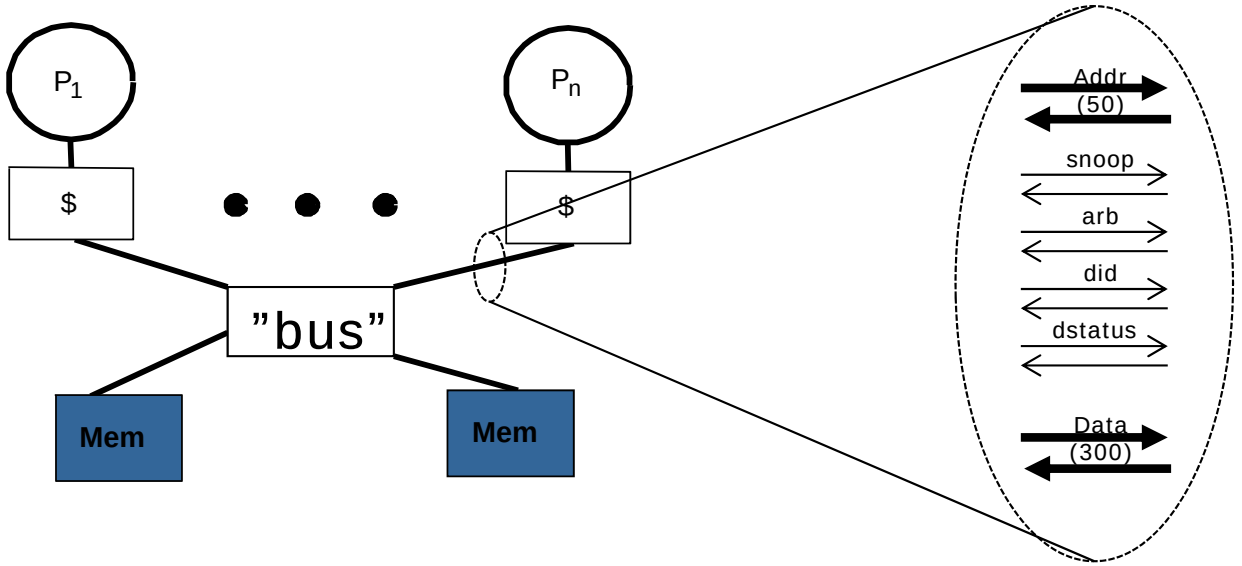
UMA
uniform
(a.k.a. SMP)

Pros/Cons Snooping Bus



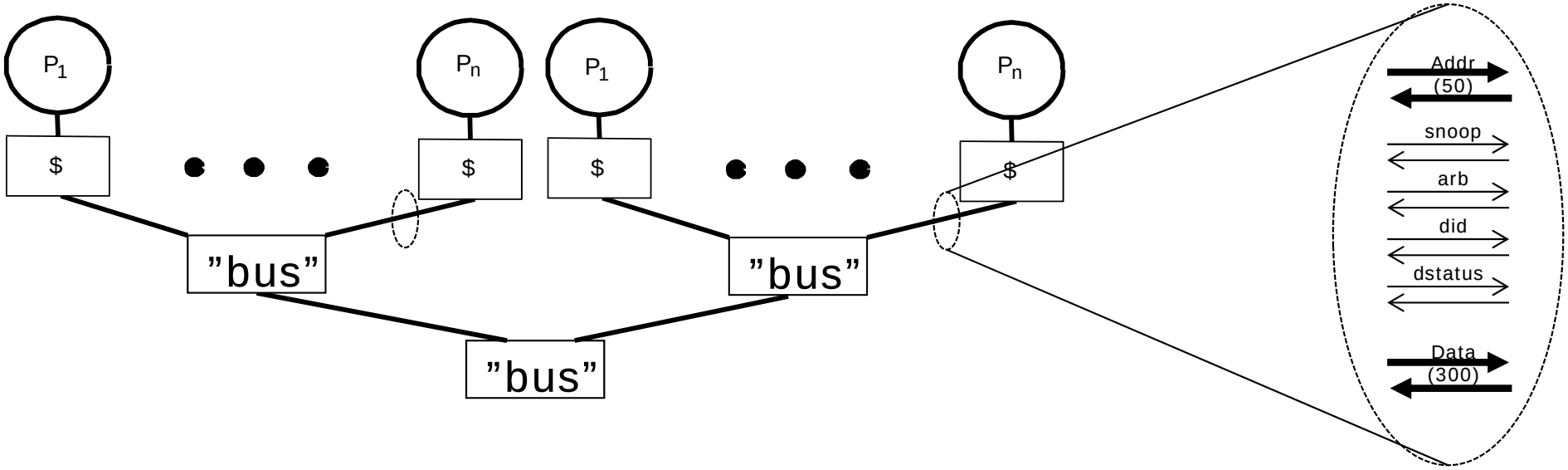
- A bus for address and data
- ++ Low latency
- ++ Makes strong memory models cheap to implement
- Not very scalable in term of
  - Many “loads” on a wire. Adds capacitance
  - Long wires. Adds resistance and signal echo…
- Capacitance & Resistance →slow signal propagation
- Ex: Sun E6000, SGI Challenger, Intel: FSB, P6

Options for building SMPs 1 (5): Central bus implementation



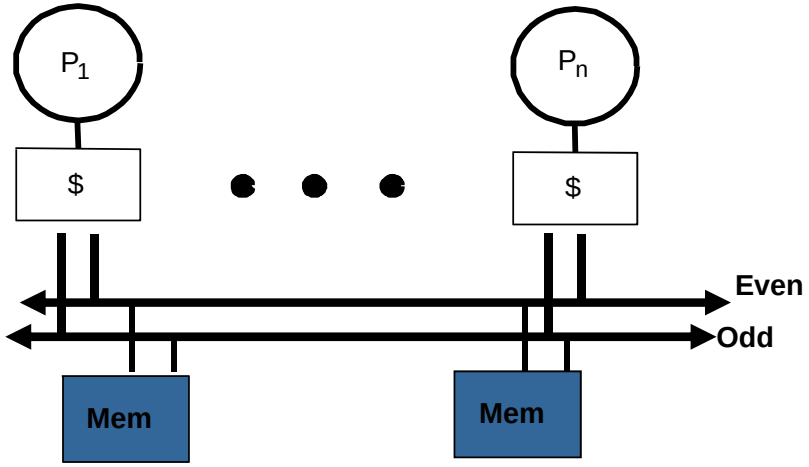
- The functionality of a bus implemented by logic
- Point-to-point interconnect to the "bus"
- - higher latency
- +Higher bandwidth (due to lower RC)
- Still: Trivial memory ordering model

Options for building SMPs 2 (5): Hierarchy of buss repeaters



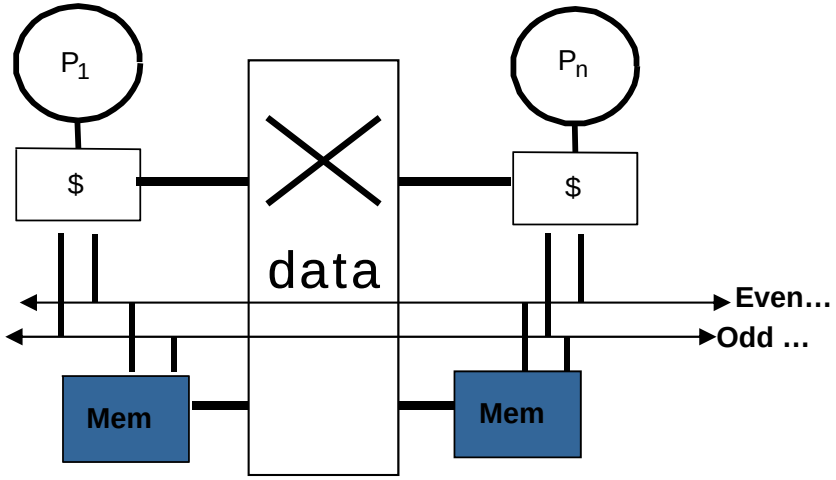
- A hierarchy of logic busses
- Point-to-point interconnect to the “bus”
- ++Shorter wires
- -- Longer latency
- Still: Trivial memory ordering model
- Ex: Sun E6800

Options for building SMPs
3 (5): Multiple busses



- Multiple busses, “striped” on some address bit
- (Potentially with central logic bus implementation)
- ++ More scalable
- + Memory ordering model fix: odd < even
- - Each cache to snoop several busses
- Ex: Sun Dragon SC2000 (2 busses), CRAY CS64000 (4 busses)

Options for building SMPs
4 (5): Data Crossbar switch



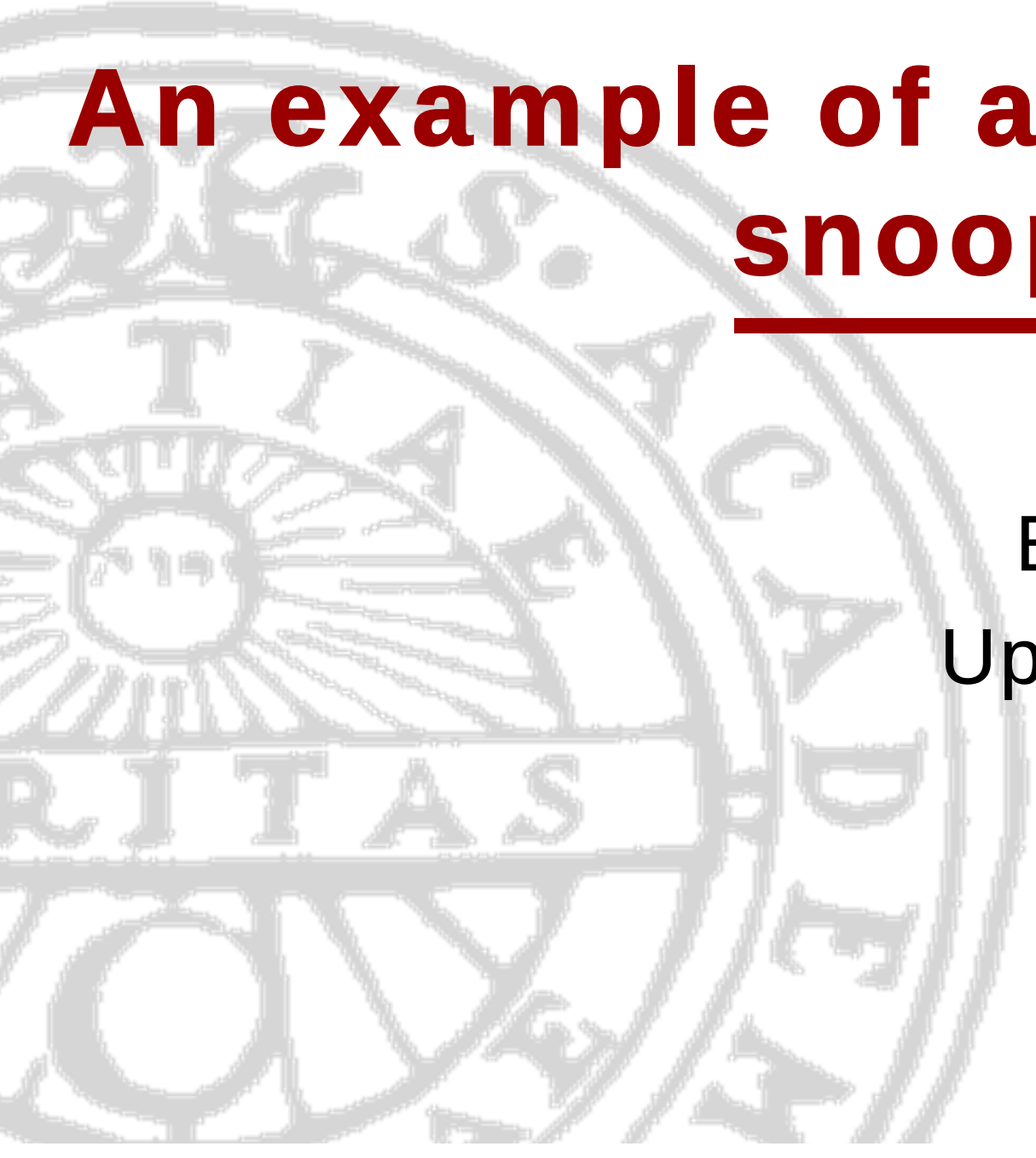
- One or many address busses ...
- A separate crossbar switch for data (p2p, ordering of data replies do not matter)
- ++ Higher bandwidth
- ++ Fewer pins (better use of the wide datapaths)
- -- Longer Latency
- Ex: Sun E10000 StarFire, E15000, E6800

UPPSALA UNIVERSITET

See what you remember

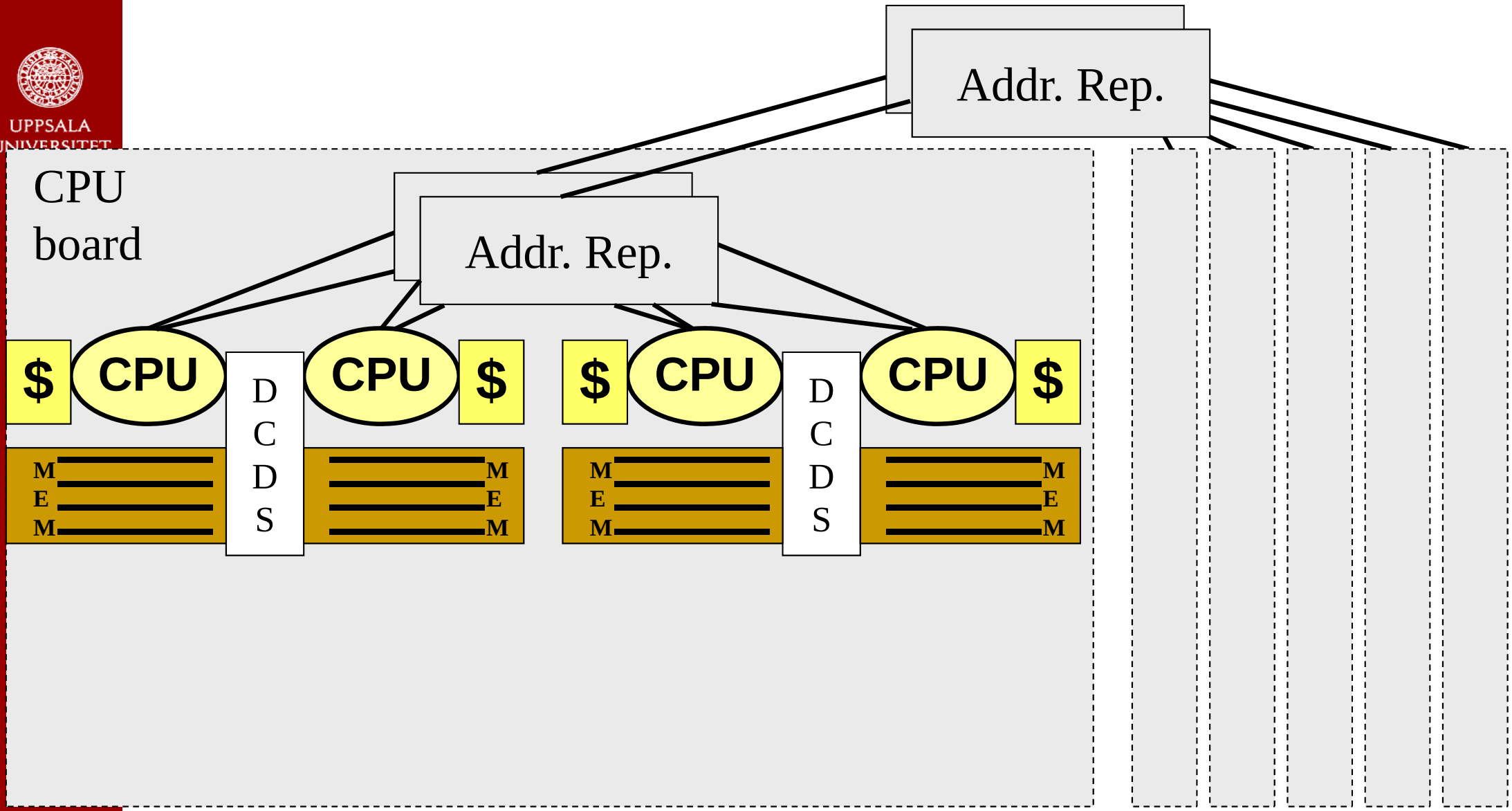
Which of these statements are true?

- ❑ Having many bus ”loads” results in increased resistance
- ❑ Point-to-point wires can be made faster than busses
- ❑ An ordered network for data makes it easier to implement SC
- ❑ A longer wire results in a higher resistance
- ❑ Signal propagation speed depends on the capacitance and resistance of the wire

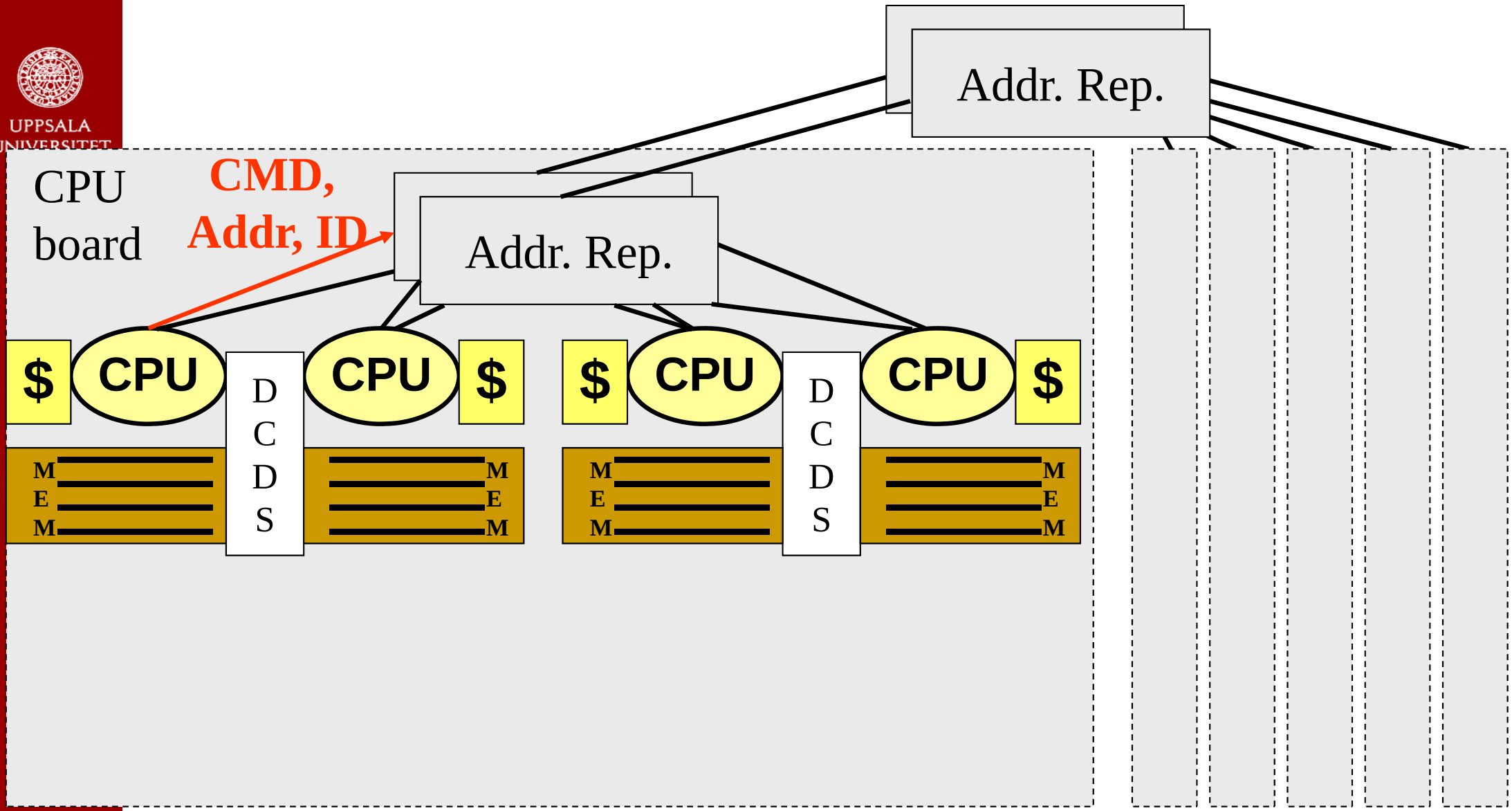


An example of an optimized snooping system

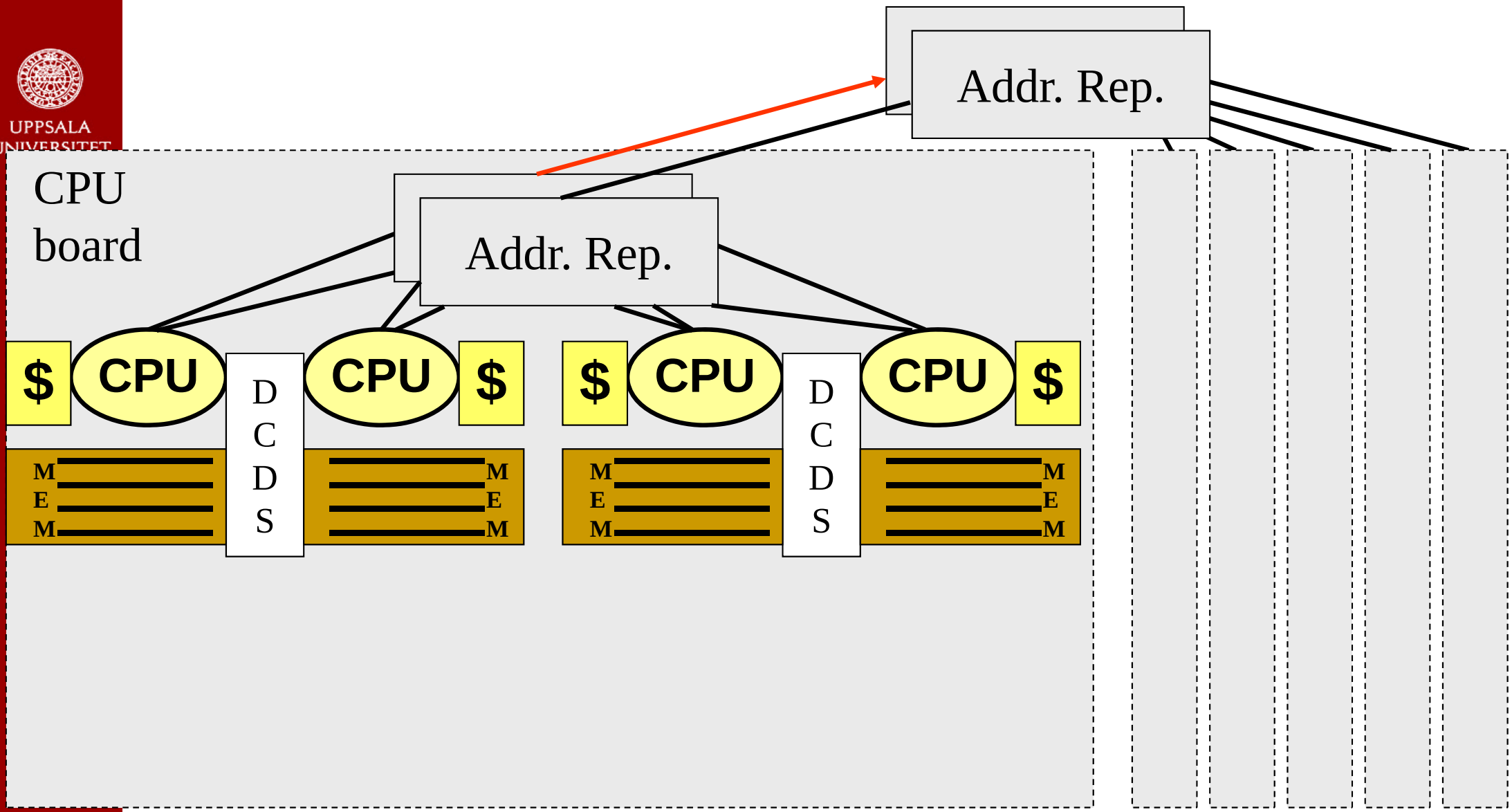
Erik Hagersten
Uppsala University

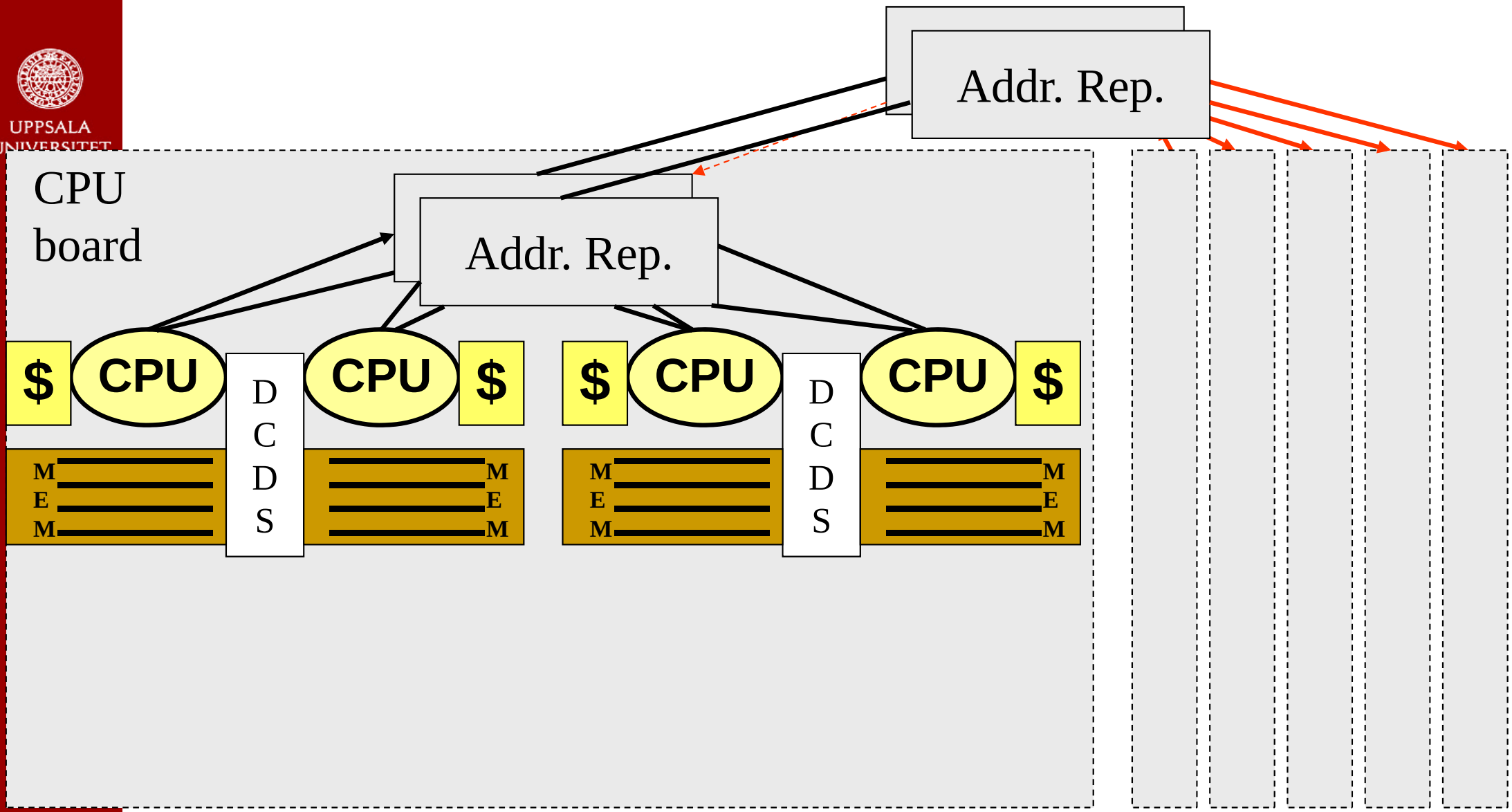


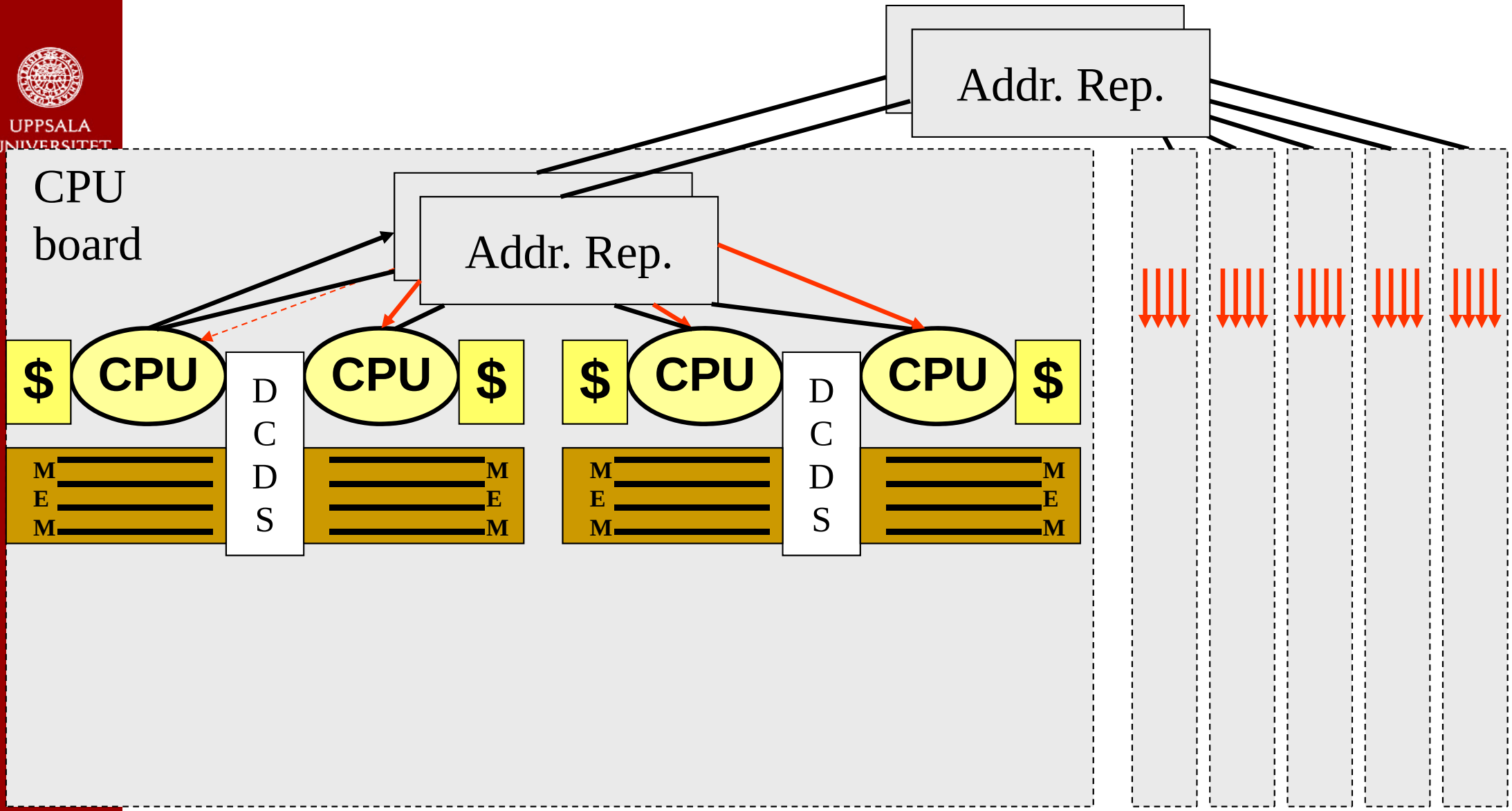
L2 cache = 8MB off chip, snoop tags on-chip

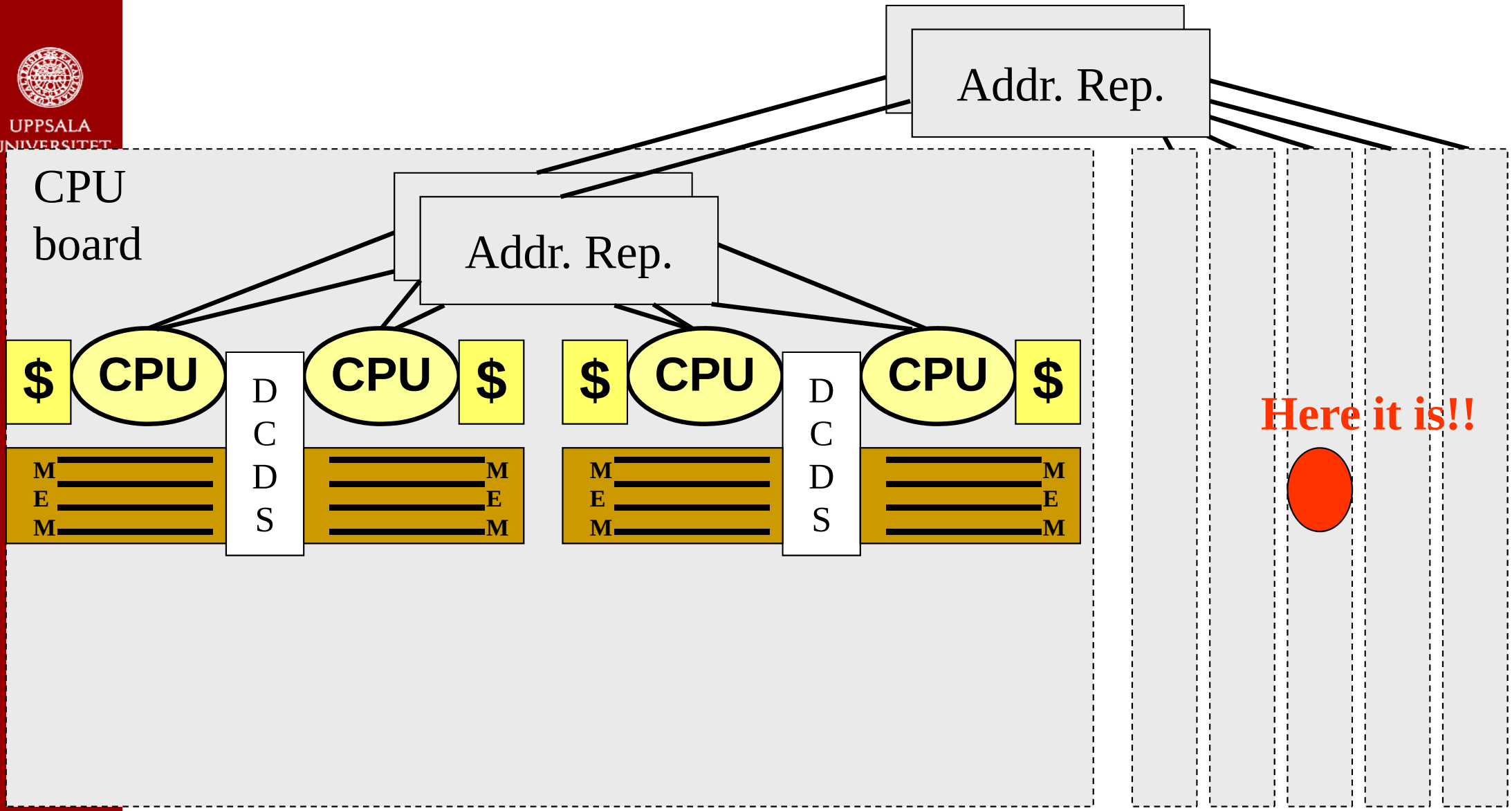

Addr. Rep.
CPU board
CMD, Addr, ID
Addr. Rep.
$
CPU
D C D S
CPU
$
$
CPU
D C D S
CPU
$
M E M
M E M
M E M
M E M

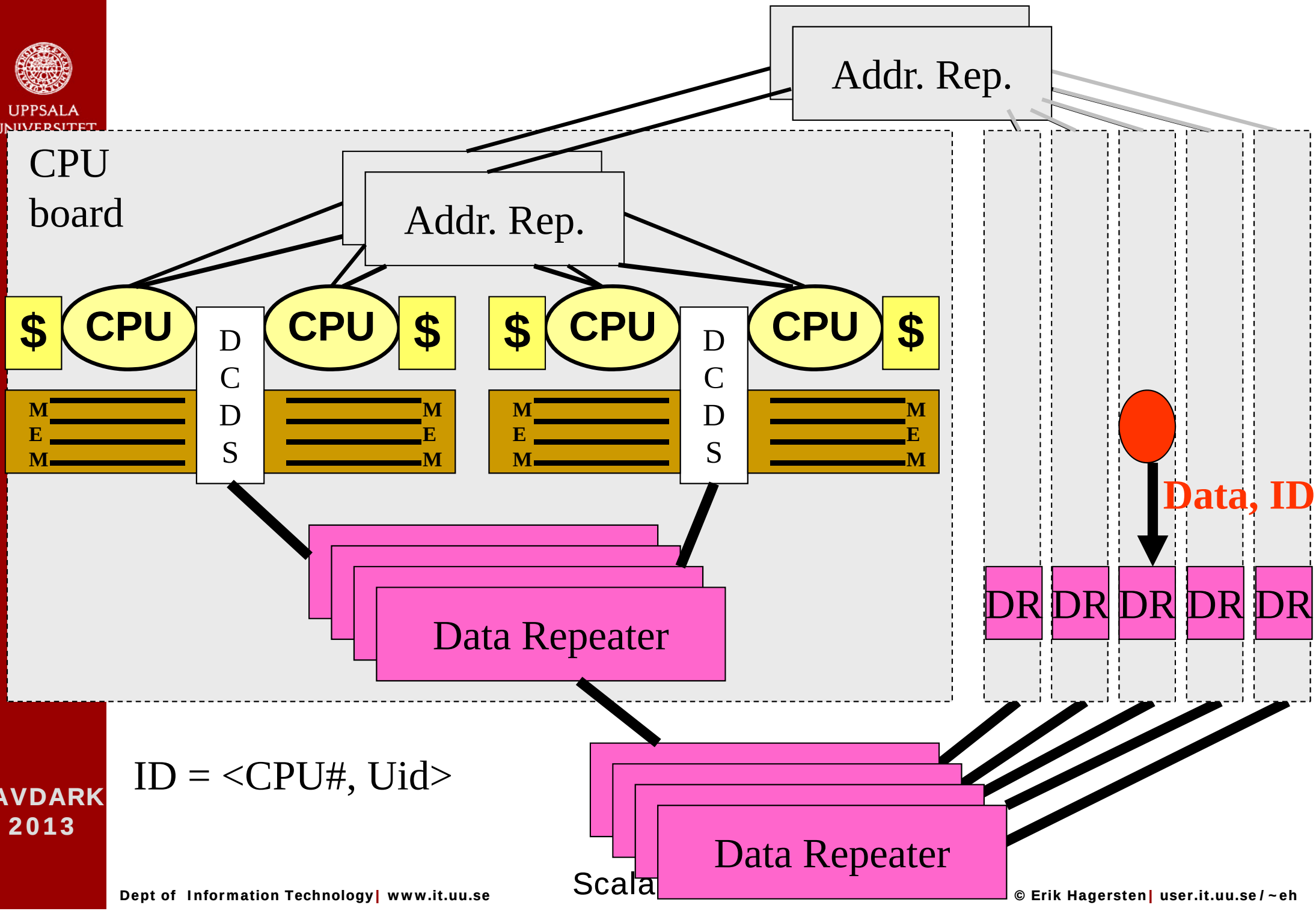
ID = <CPU#, Uid>

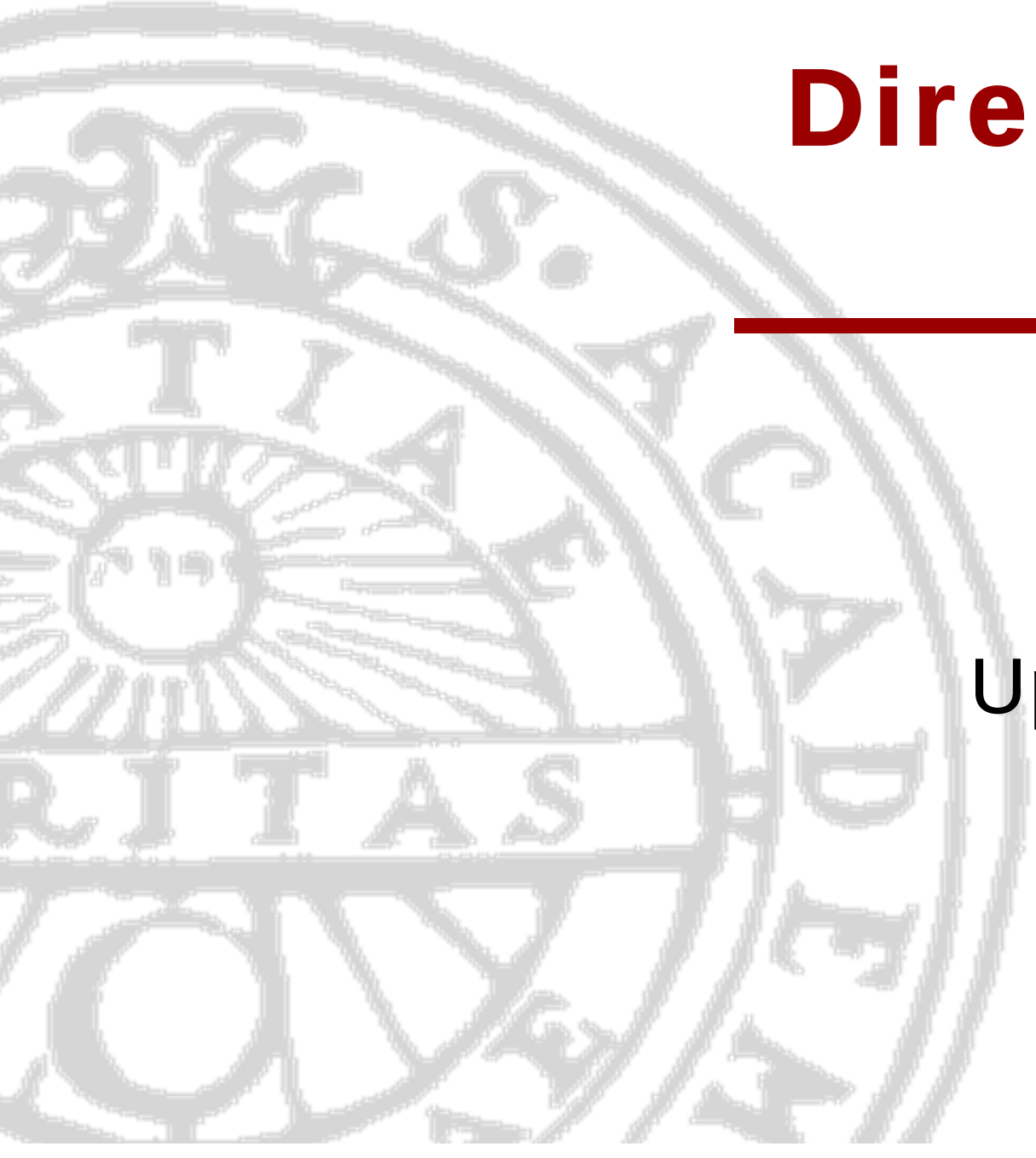

Addr. Rep.
Addr. Rep.
CPU
board
$
CPU
$
CPU
$
CPU
$
CPU
D
C
D
S
D
C
D
S
M
E
M
M
E
M
M
E
M
M
E
M


Addr. Rep.
Addr. Rep.
CPU
board
$
CPU
$
CPU
$
CPU
$
CPU
D
C
D
S
D
C
D
S
M
E
M
M
E
M
M
E
M
M
E
M


Addr. Rep.
Addr. Rep.
CPU
board
$
CPU
D
C
D
S
CPU
$
$
CPU
D
C
D
S
CPU
$
M
E
M
M
E
M
M
E
M
M
E
M


Addr. Rep.
Addr. Rep.
CPU
board
$
CPU
$
CPU
$
CPU
$
CPU
D
C
D
S
D
C
D
S
M
E
M
M
E
M
M
E
M
M
E
M
Here it is!!


Addr. Rep.
CPU board
Addr. Rep.
$
CPU
D C D S
CPU
$
$
CPU
D C D S
CPU
$
M E M
M E M
M E M
M E M
Data, ID
Data Repeater
DR
DR
DR
DR
DR
ID = <CPU#, Uid>
Data Repeater
Scala



Directory-based coherence

Erik Hagersten
Uppsala University
Sweden

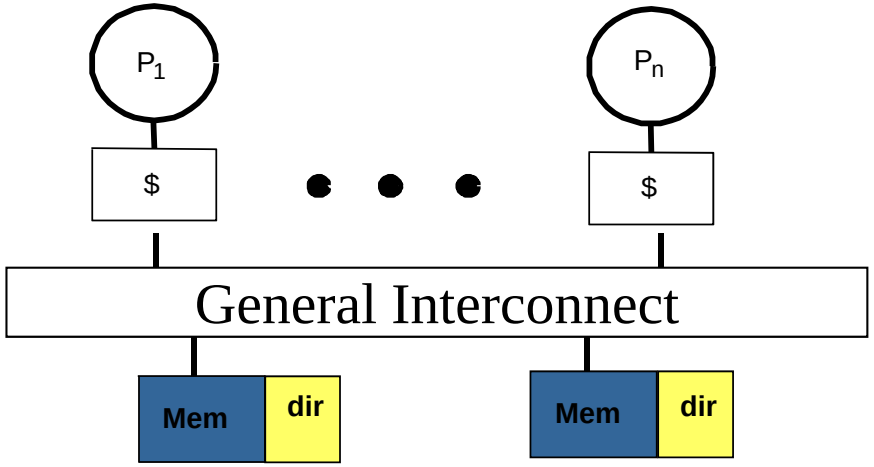
UPPSALA UNIVERSITET

Implementation options for memory coherence

- Two coherence options
  - Snoop-based (”broadcast”)
  - Directory-based (”point to point”)
- Different techniques for implementing memory models
- Varying scalability

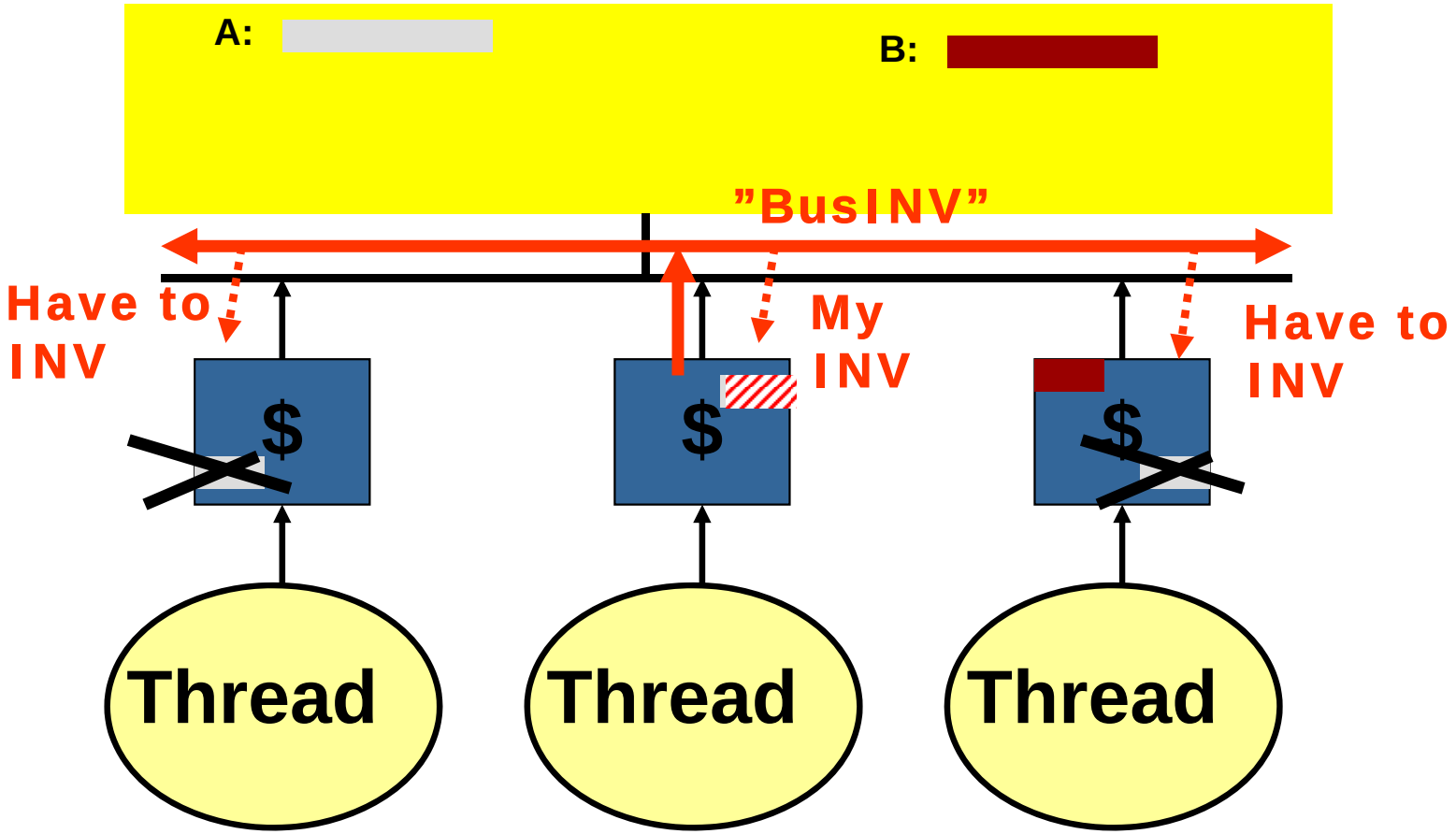
AVDARK 2013

Options for building SMPs
5 (5): Directory-based

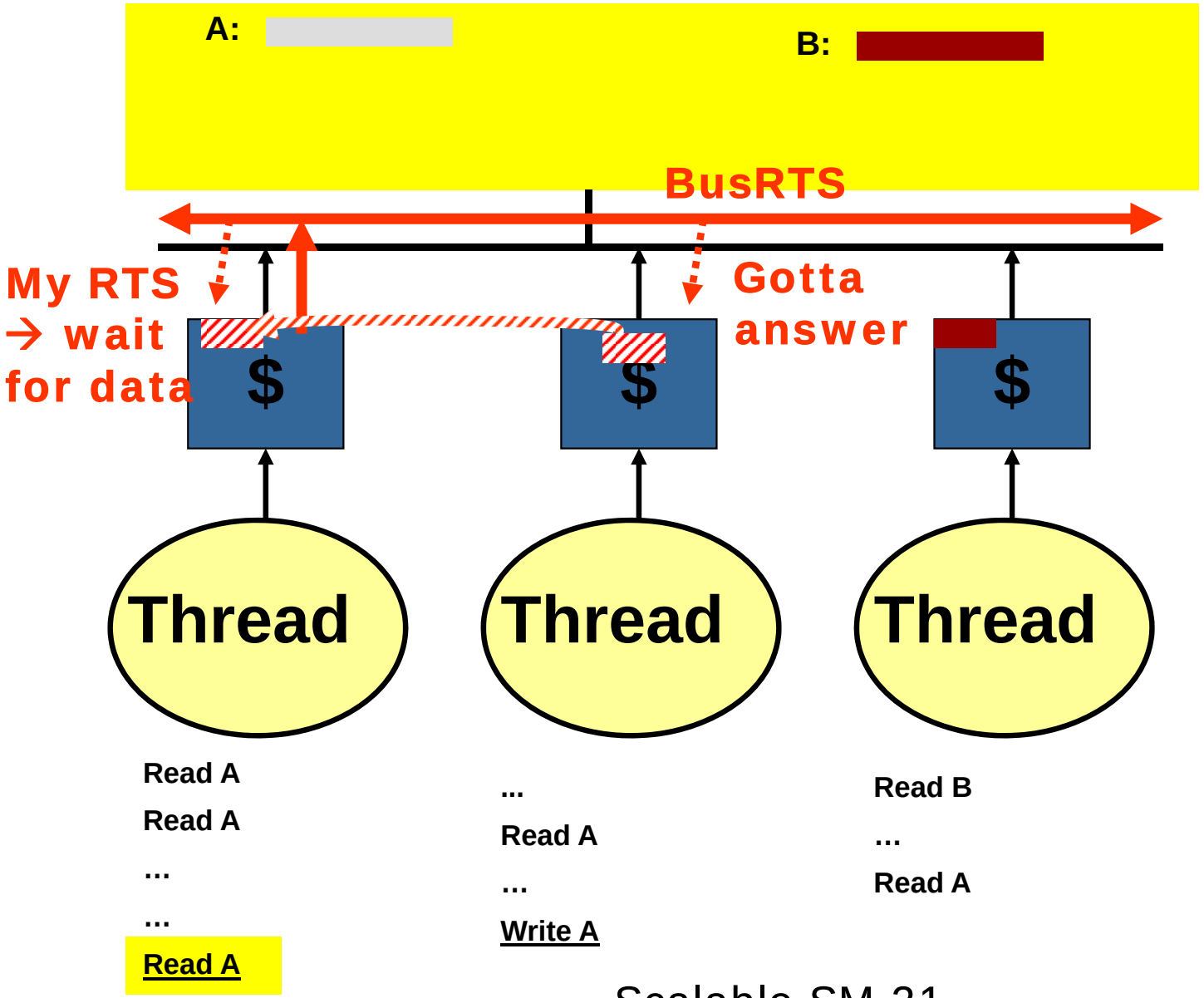


- **General interconnect**
- **Directory-based coherence** (**point-to-point**)
- More expensive to implement stronger mem models
- ++ Scales to more nodes
- -- Longer Latency (especially cache-to-cache)
- Ex: HP V9000, IBM S60

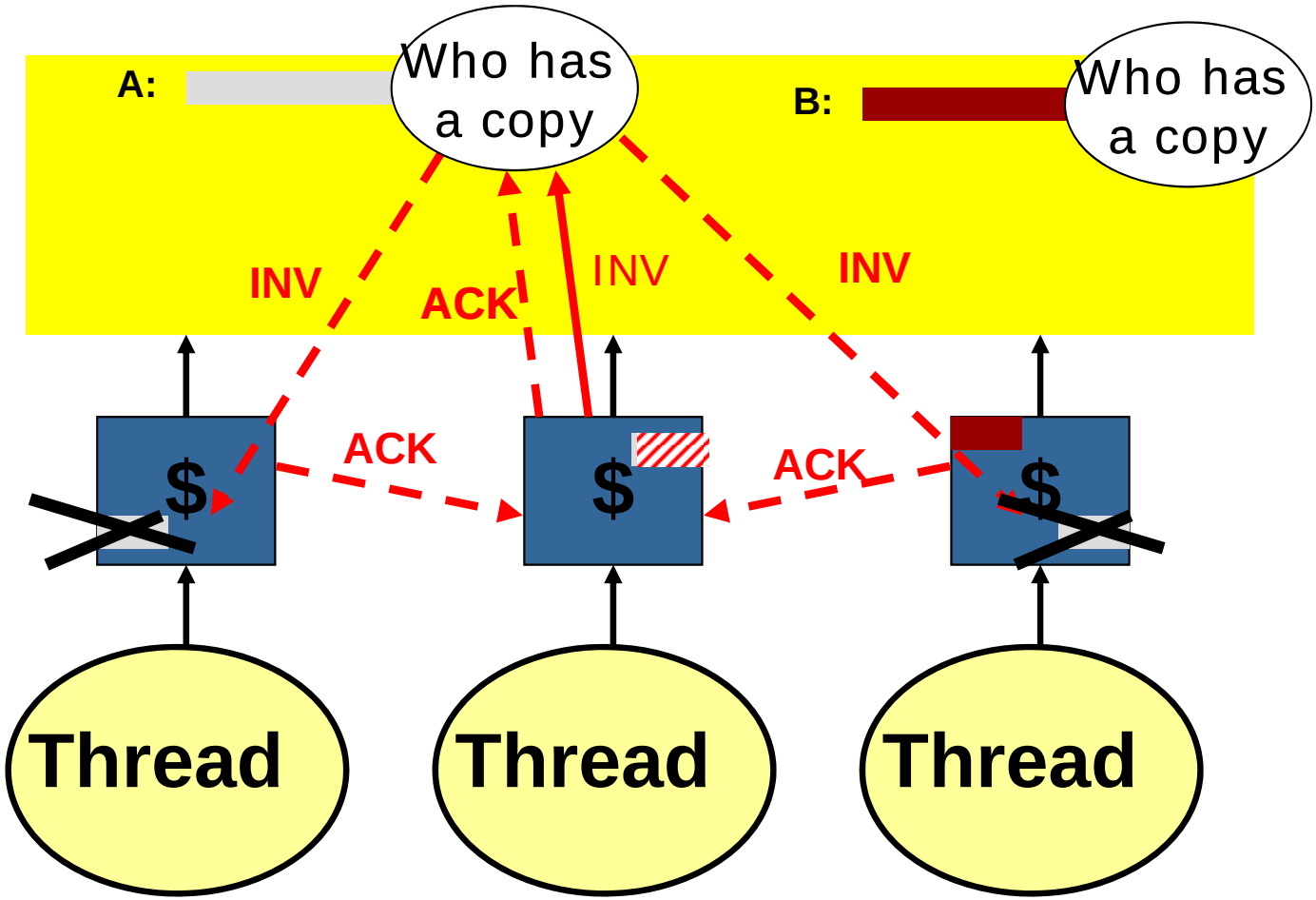
”Upgrade” in snoop-based



Cache-to-cache in snoop-based

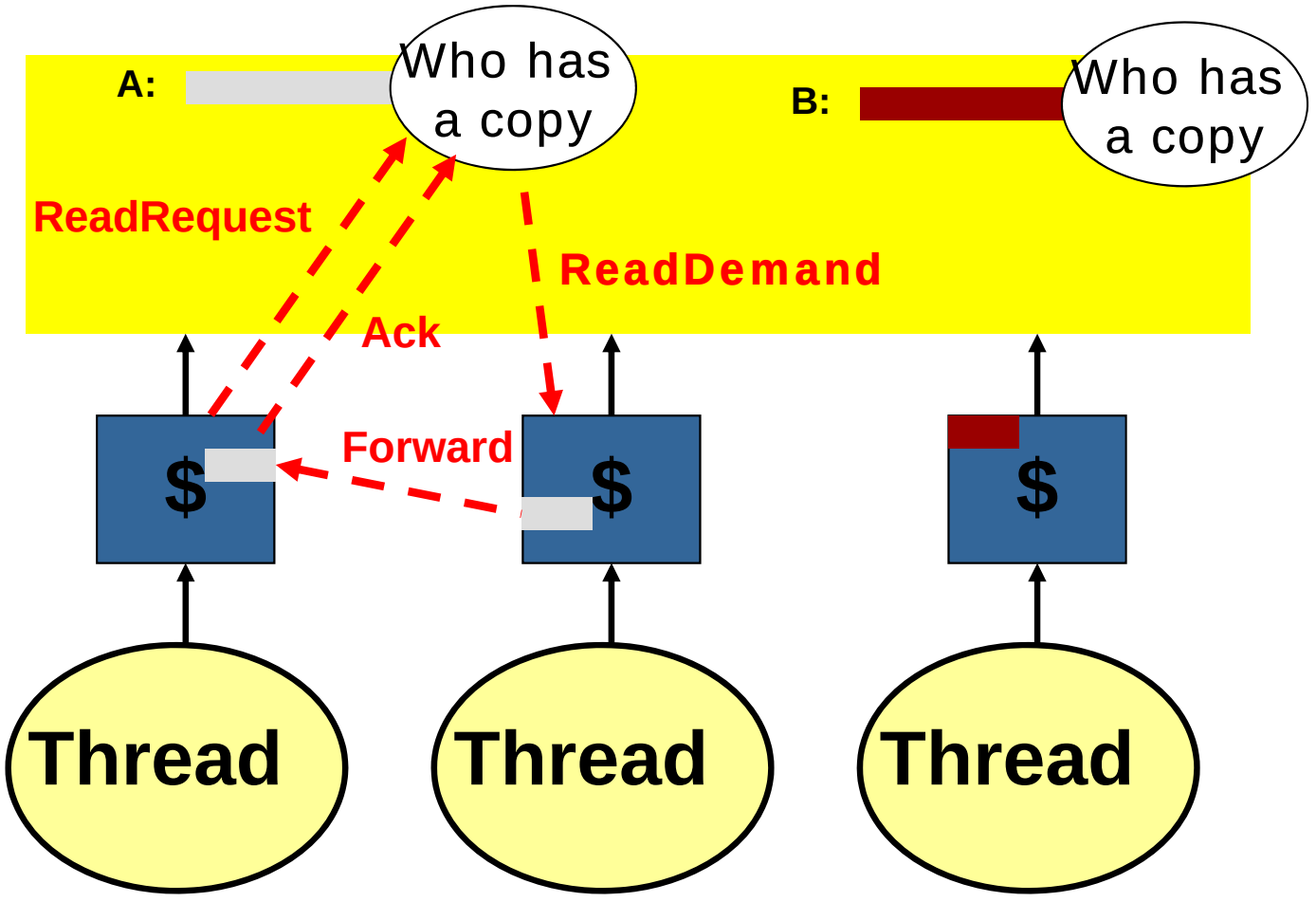


”Upgrade” in dir-based

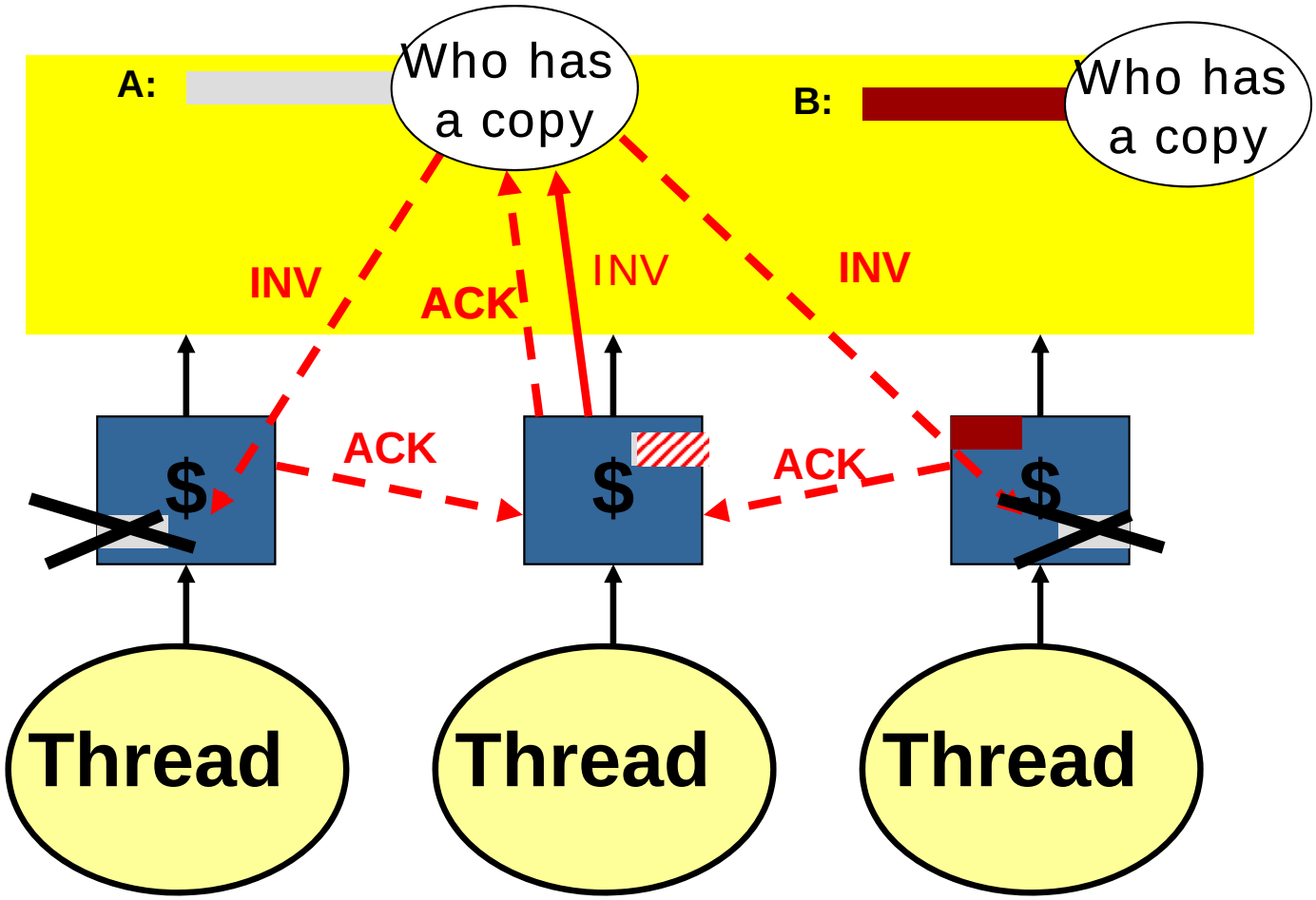


Read A	...	Read B
Read A	Read A	...
...	...	Read A
...	**Write A**	

Cache-to-cache in dir-based



Implementing SC with directory coh



UPPSALA UNIVERSITET

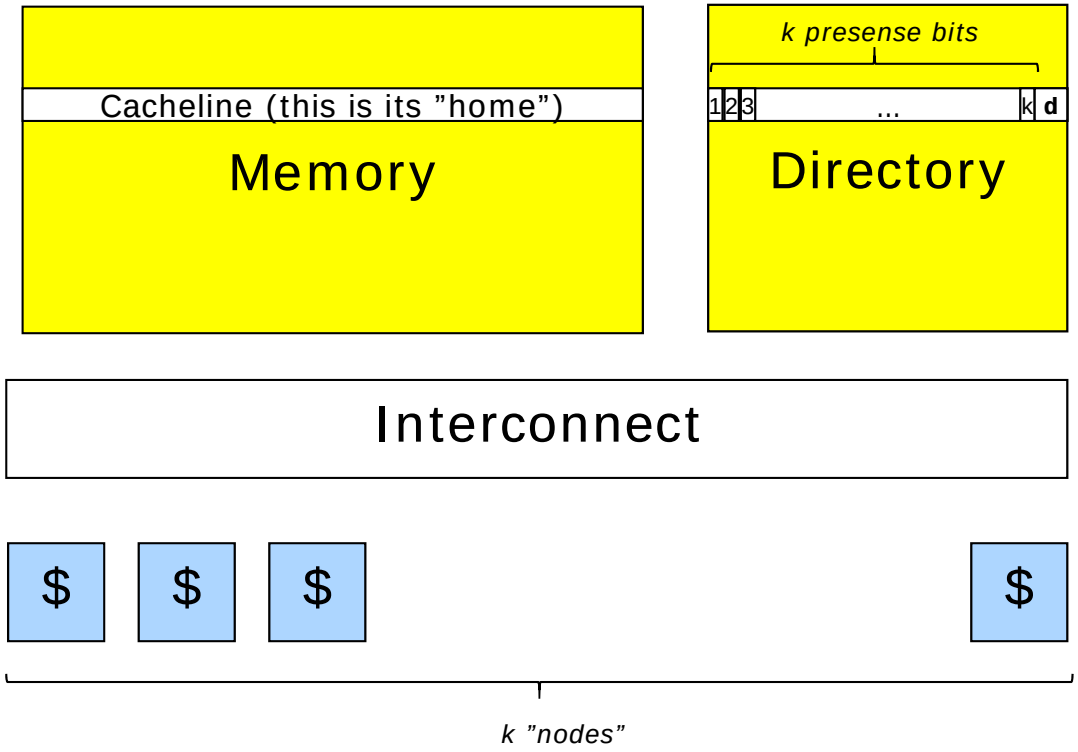
Why snoop?

- A ”bus”: a serialization point helps coherence and memory ordering
- Upgrade is faster (S → M by ”invalidation”)
- Cache-to-cache is often faster
- Synchronization, is a combination of both

- ...but it is expensive to scale the bandwidth☹

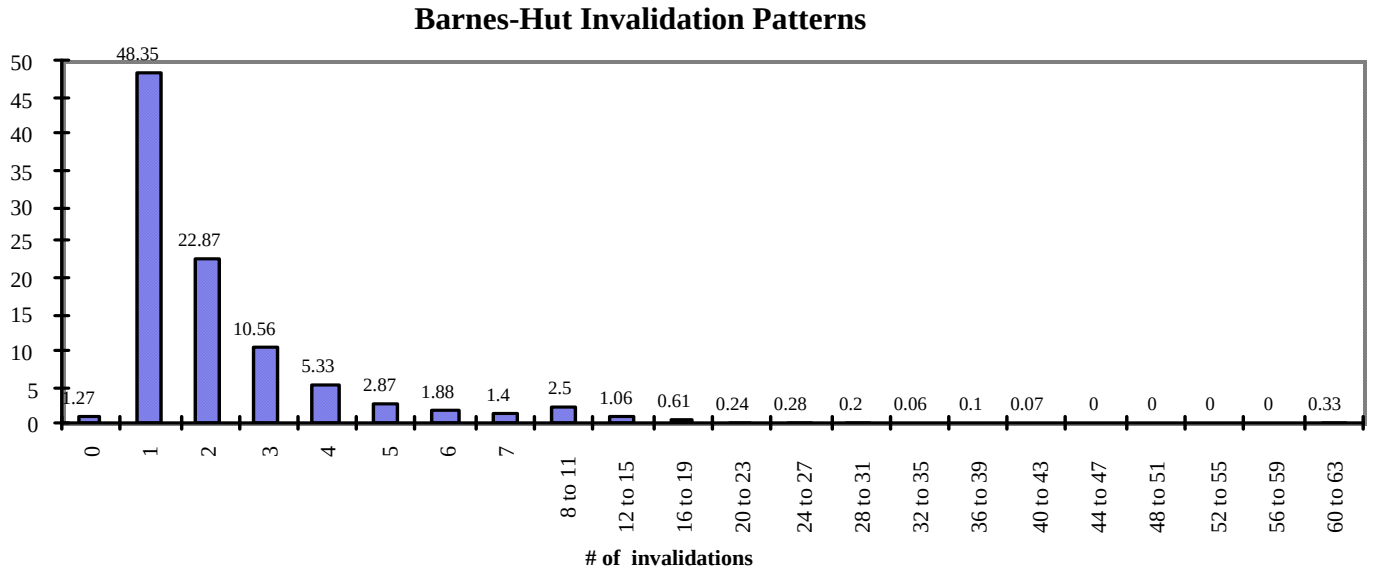
Directory structures

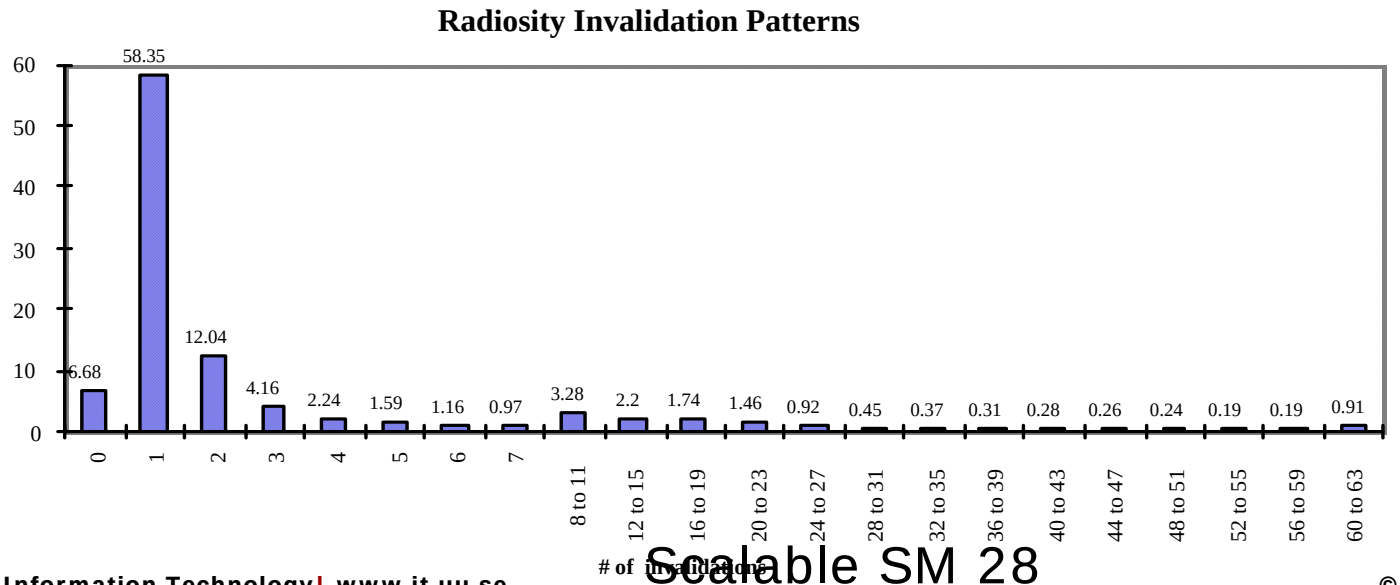
Fully mapped directory



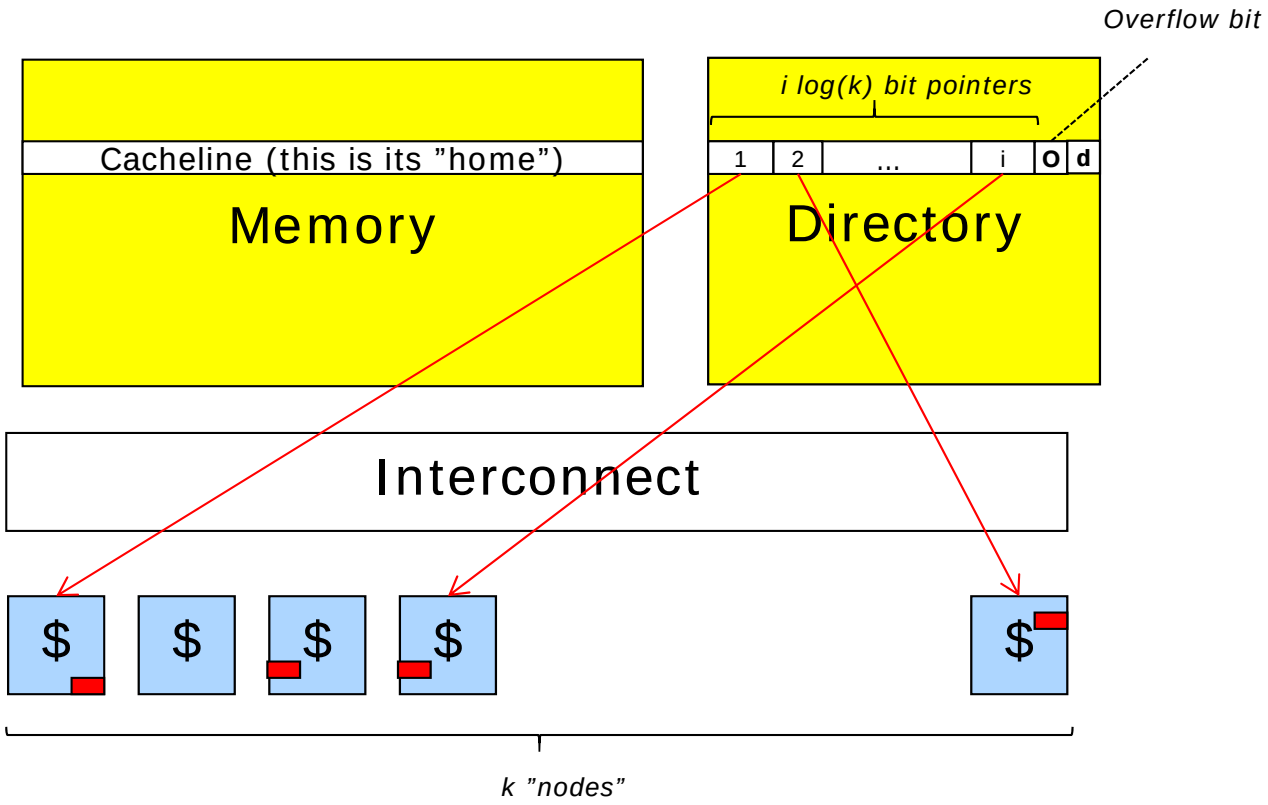
- ***k* Nodes**
- **Dir entry per cacheline in home memory: *k* presence-bits + 1 dirty-bit**

Cache Invalidation Patterns



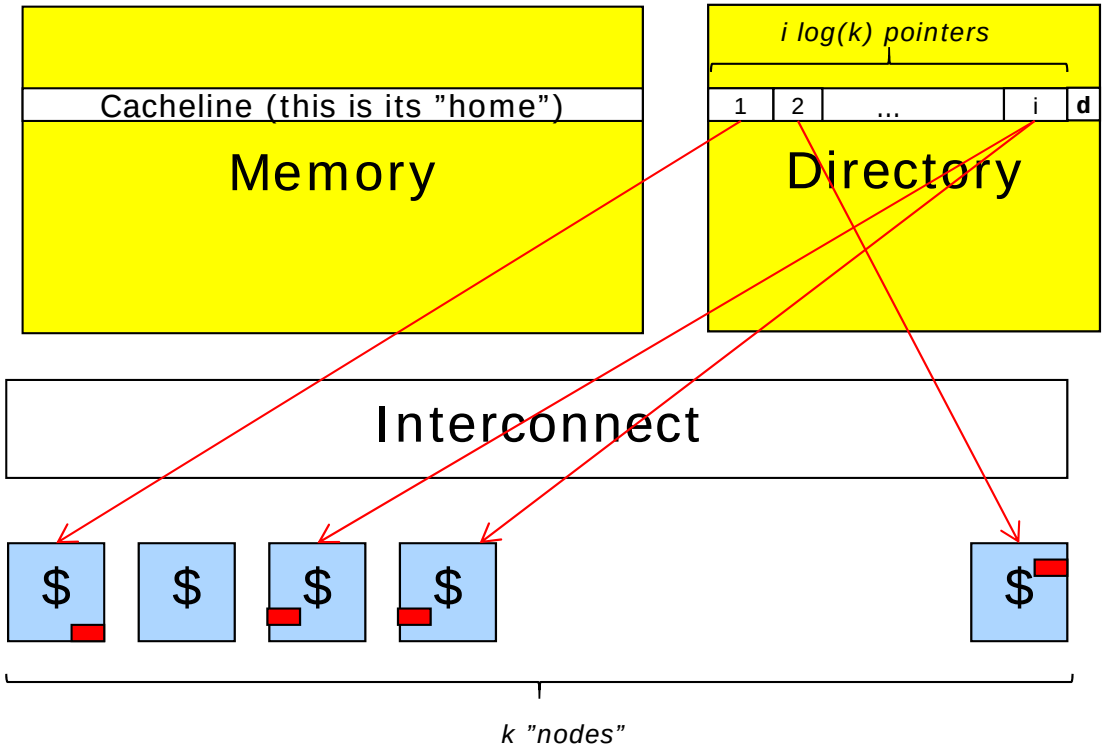


Broadcast Dir_iB (e.g. Dir_3B)



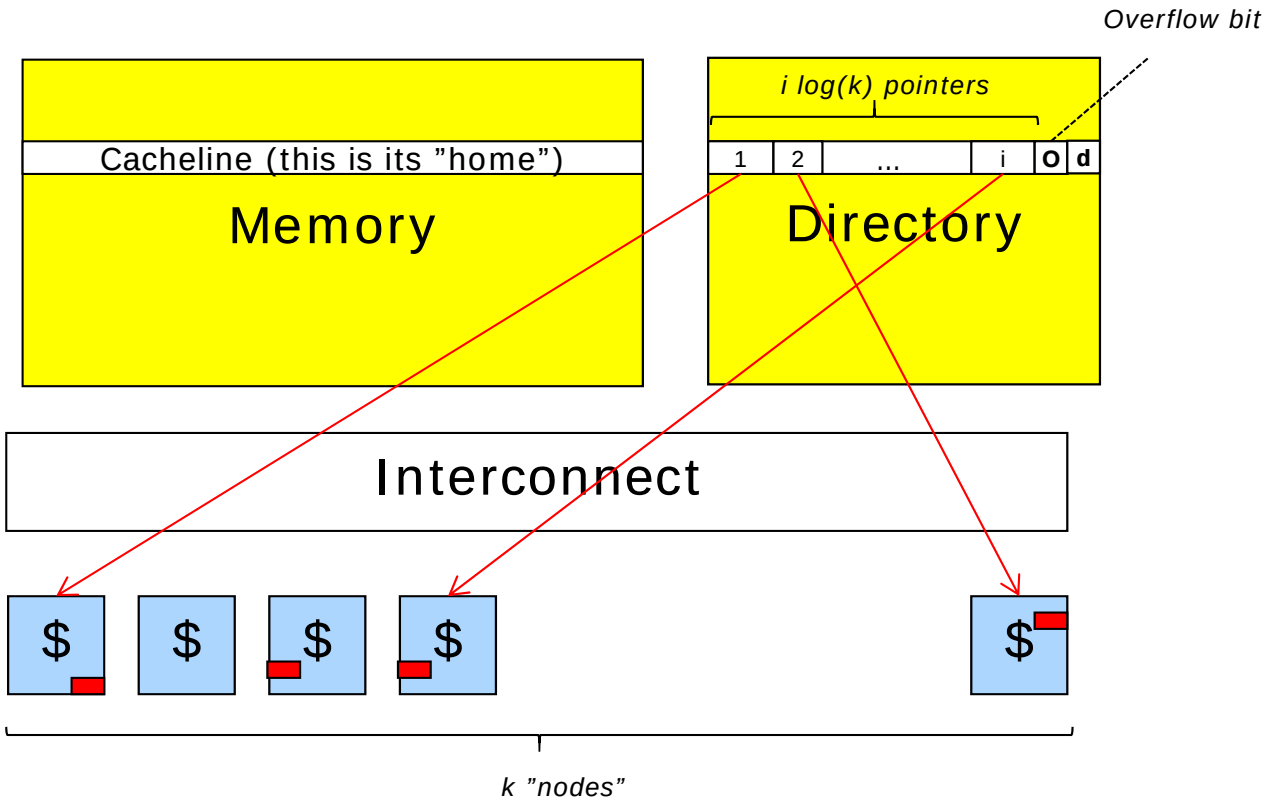
- Broadcast (Dir_iB)
  - A limited number of log(k) bit pointers (here *i*)
  - Typically, *i* is small → Fewer bits needed in the directory
  - Overflow bit turned on if more sharers than pointers
  - Turns into ”broadcast mode” and send invalidates to all
  - Changes back to pointers after invalidate
  - Bad for widely shared invalidated data

Dir_iNB (e.g. Dir_3NB)



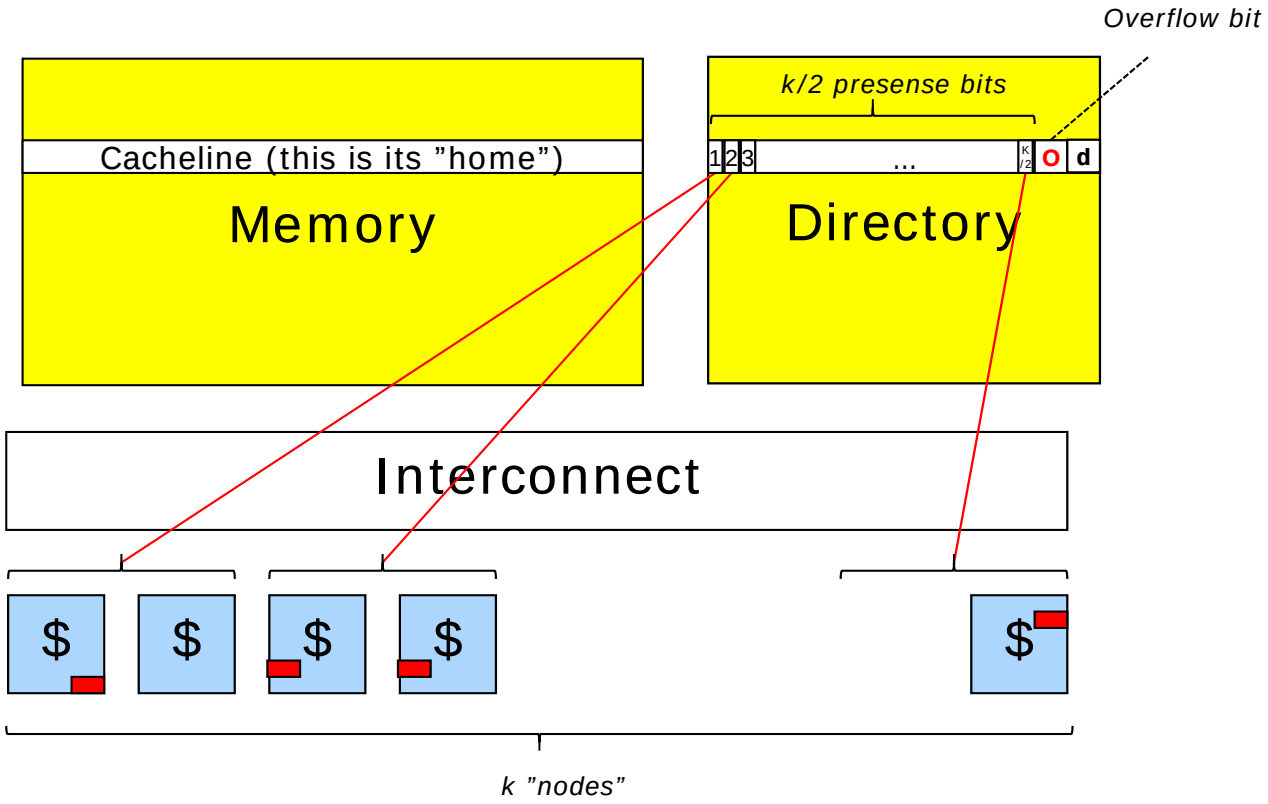
- (Dir_iNB)
  - If more sharers than pointers, force eviction and reuse pointer
  - Bad for widely shared data

Course-grain Vector Dir_iCV (Dir_3CV)



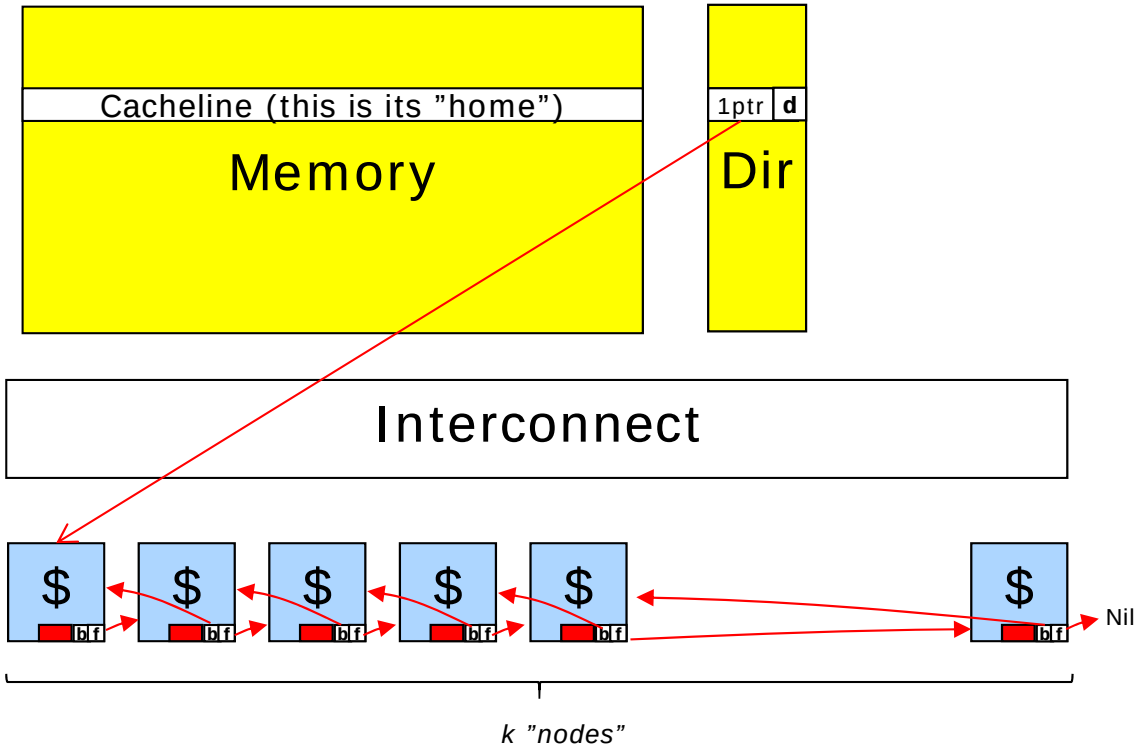
- Dir_iCV
  - Overflow bit turned on if more sharers than pointers

Broadcast Dir_iCV (e.g. Dir_3CV)

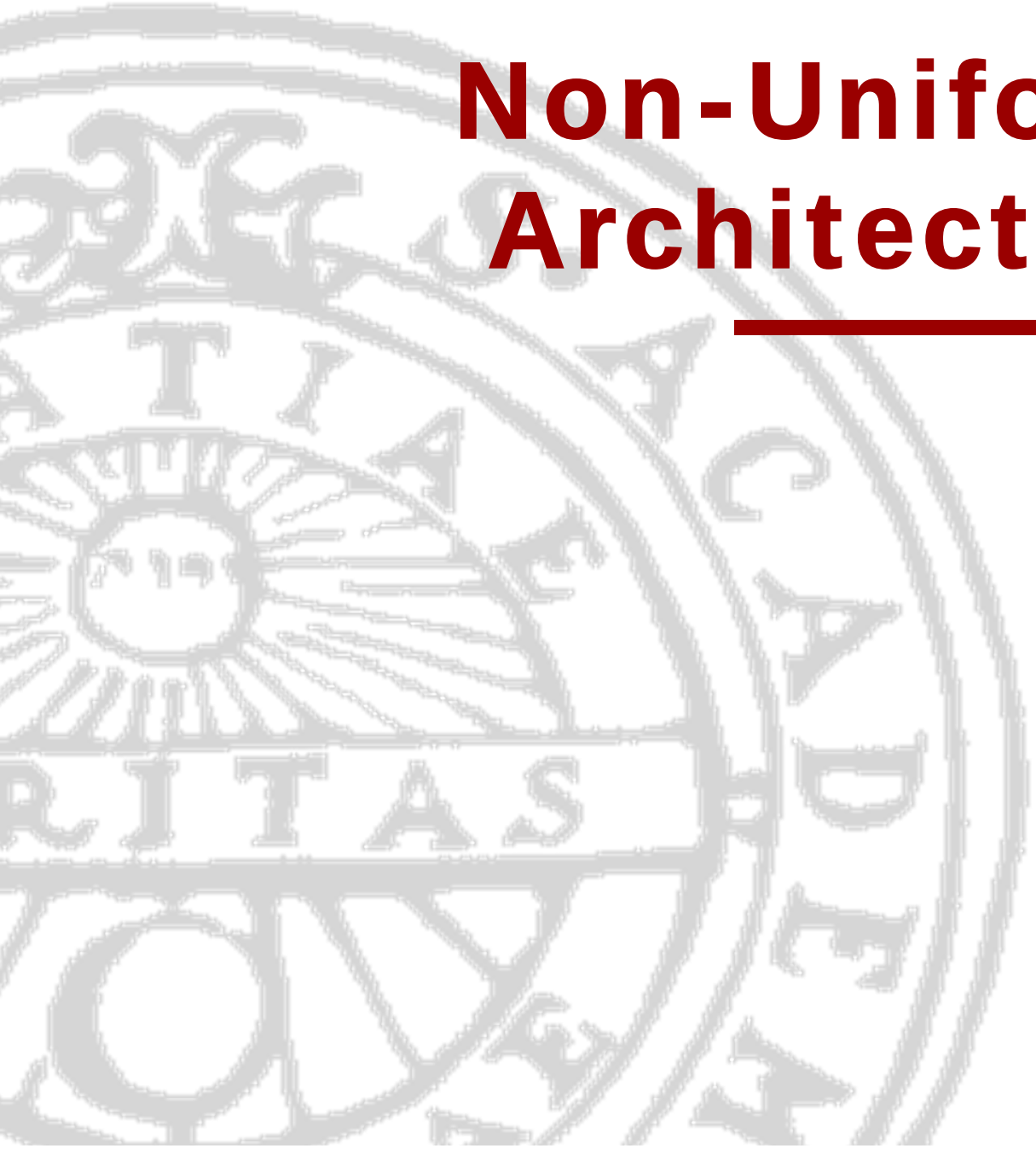


- Broadcast (Dir_iCV)
  - Overflow bit turned on if more sharers than pointers
  - Directory changed to (e.g., k/2) course presence bits
  - Each bit indicates presence in a group of nodes (here 2)
  - Invalidates sent to all in the group if bit is set
  - Change back to pointers on invalidates
  - Potentially less traffic than DIRiB due to limited broadcast

Scalable Coherence Interface, SCI

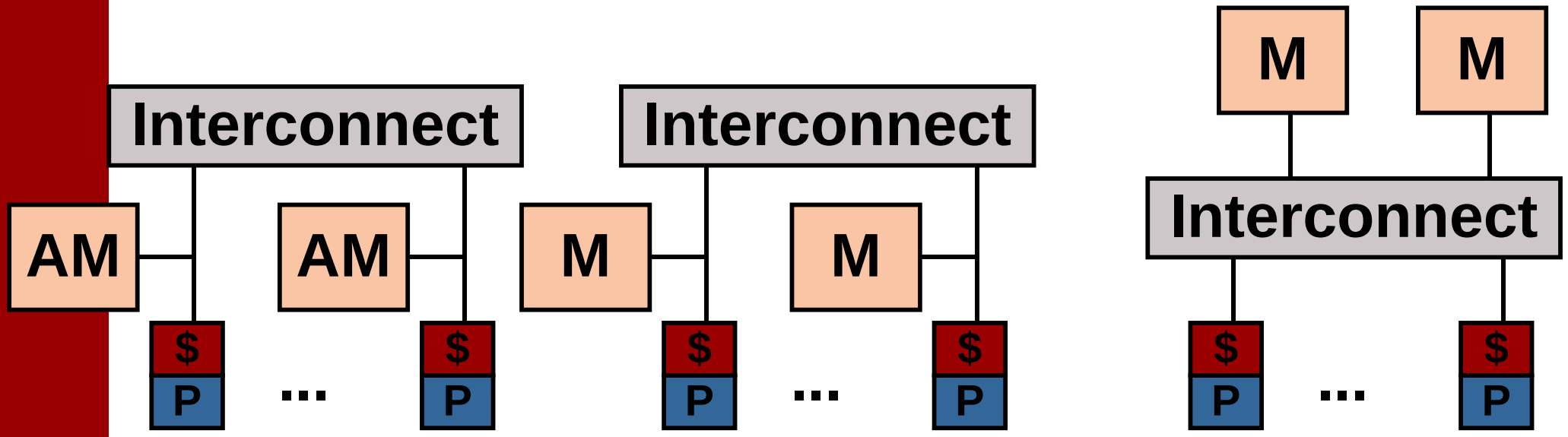


- Scalable Coherence Interface (SCI)
  - A single pointer in the directory, pointing to one sharer
  - Each caches cache line has two pointers (forward, backwards)
  - Doubly-linked list containing all sharers
  - Insertion at the head. Deletion adds complexity
  - Invalidates follows the linked list until <f = Nil>
  - High complexity and long latency for invalidates



Non-Uniform Memory Architecture (NUMA)

Three options for scalable shared memory



COMA
cache-only

NUMA
non-uniform

UMA
uniform
(a.k.a. SMP)

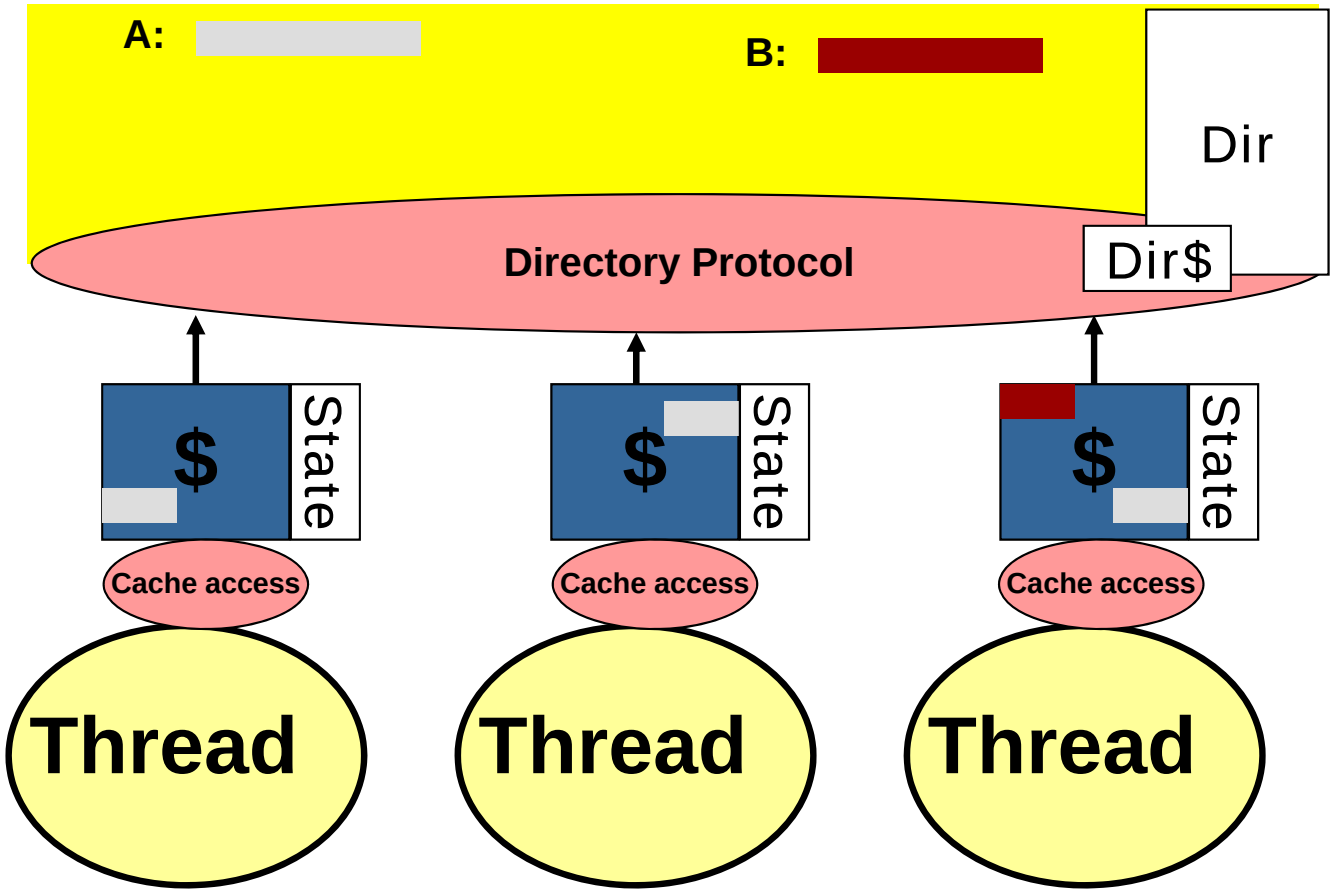
UPPSALA UNIVERSITET

Why directory-based

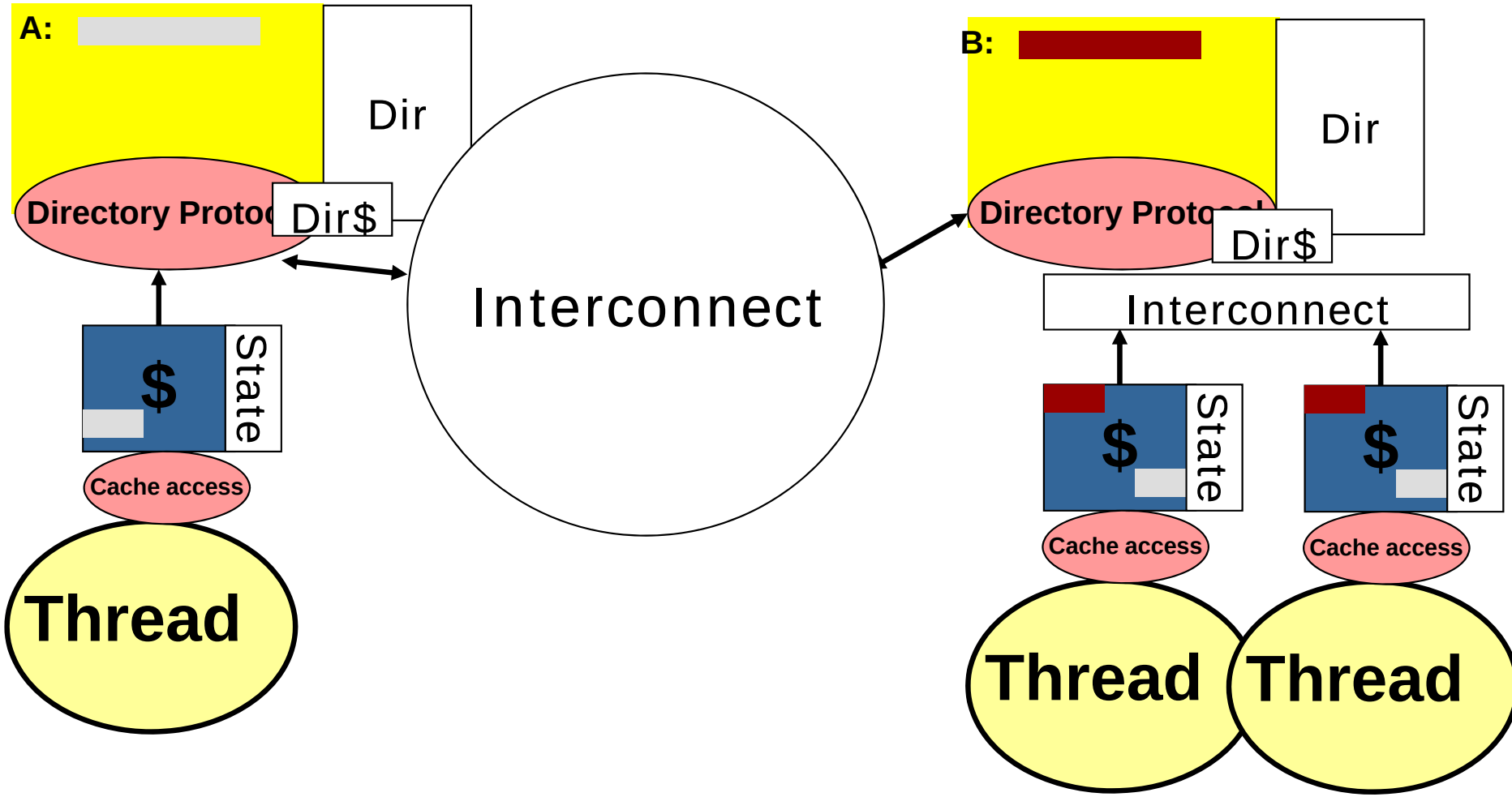
- P2P messages → **high** bandwidth
- Much more scalable!
- Suits out-of-the-box coherence

- Note:
  - Dir-based coherence can also be used to build a uniform-memory architecture (UMA/SMP)

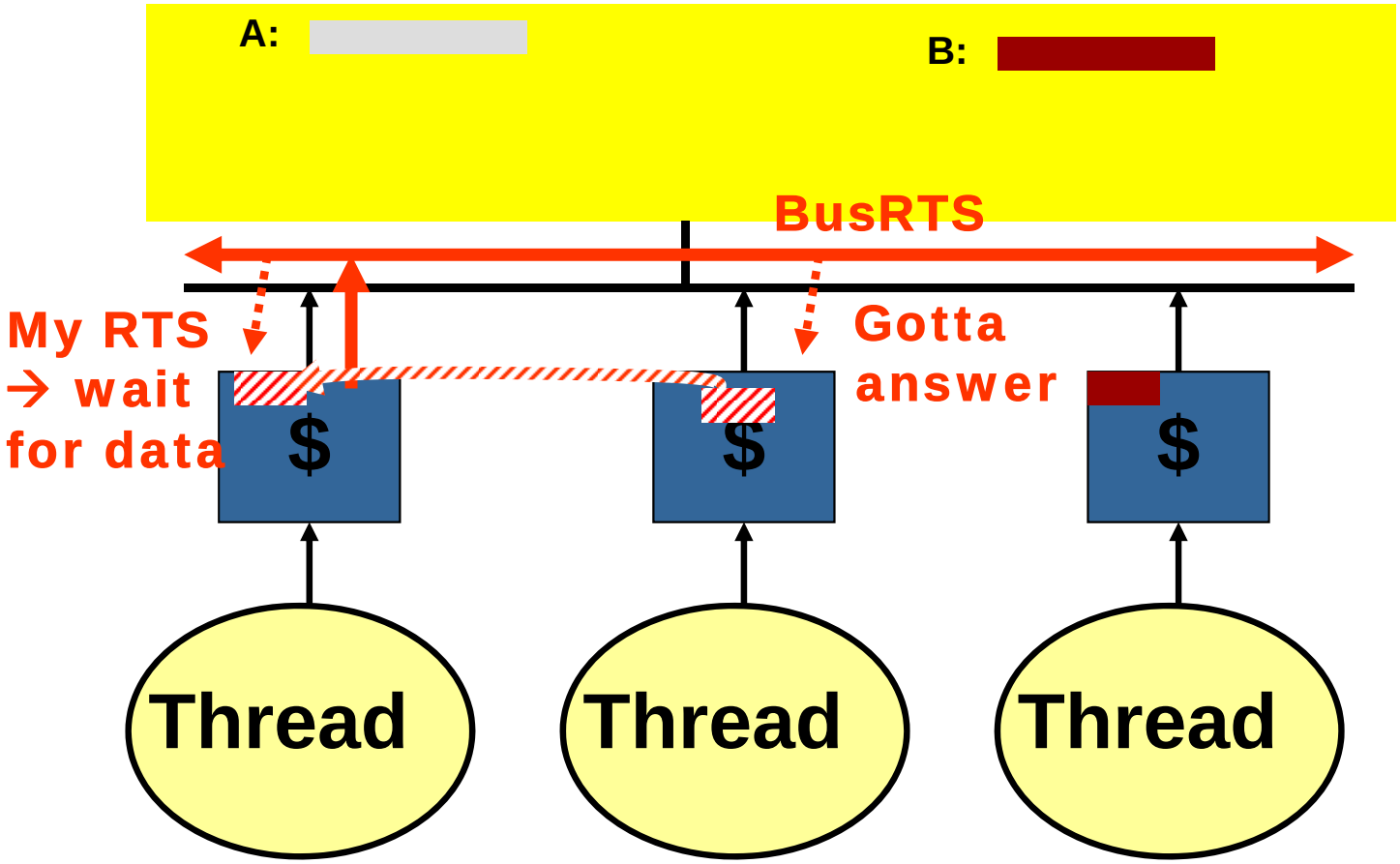
Directory-based coherence: per-cachline info in the memory



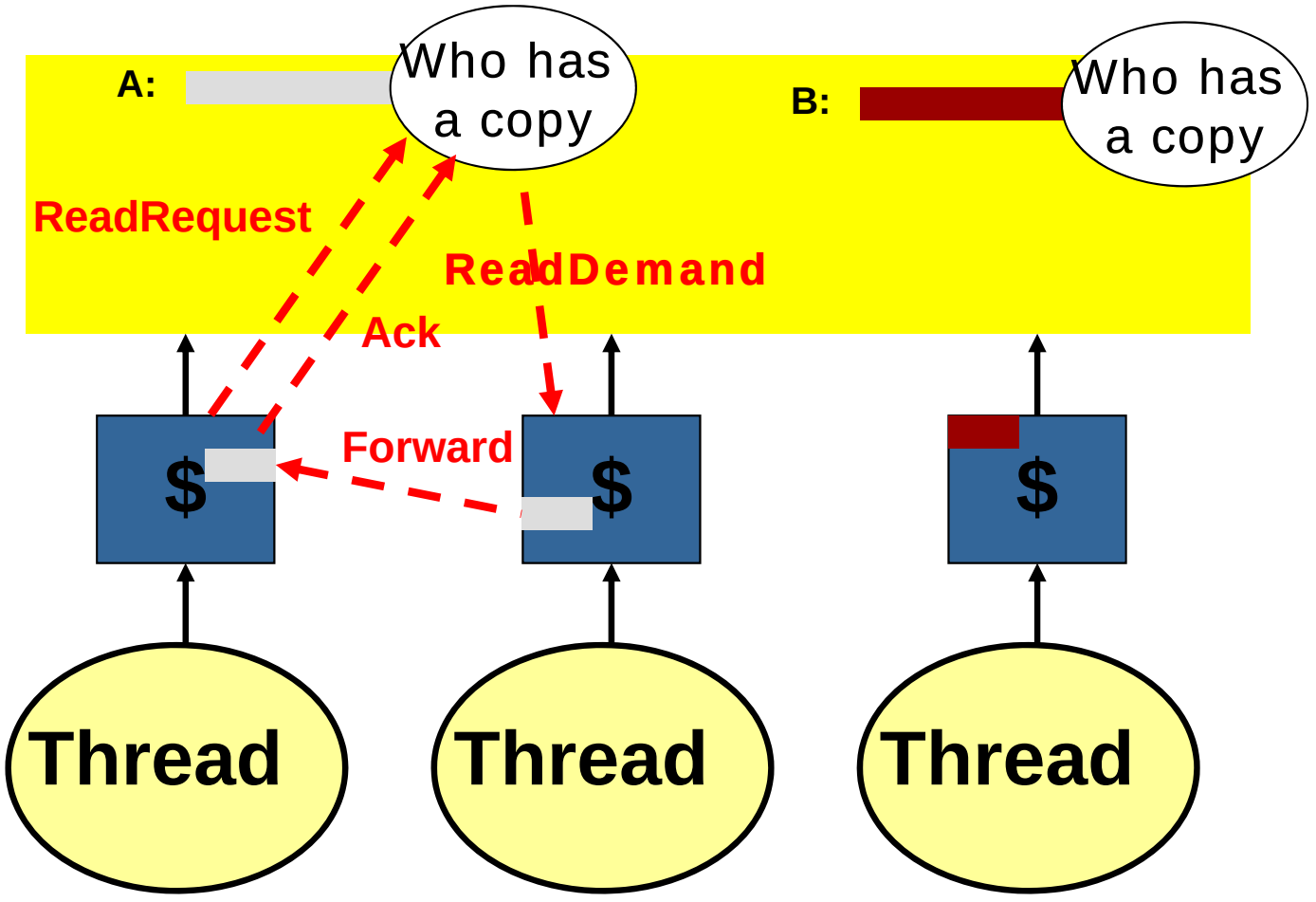
Directory-based snooping: NUMA. Per-cachline info in the home node



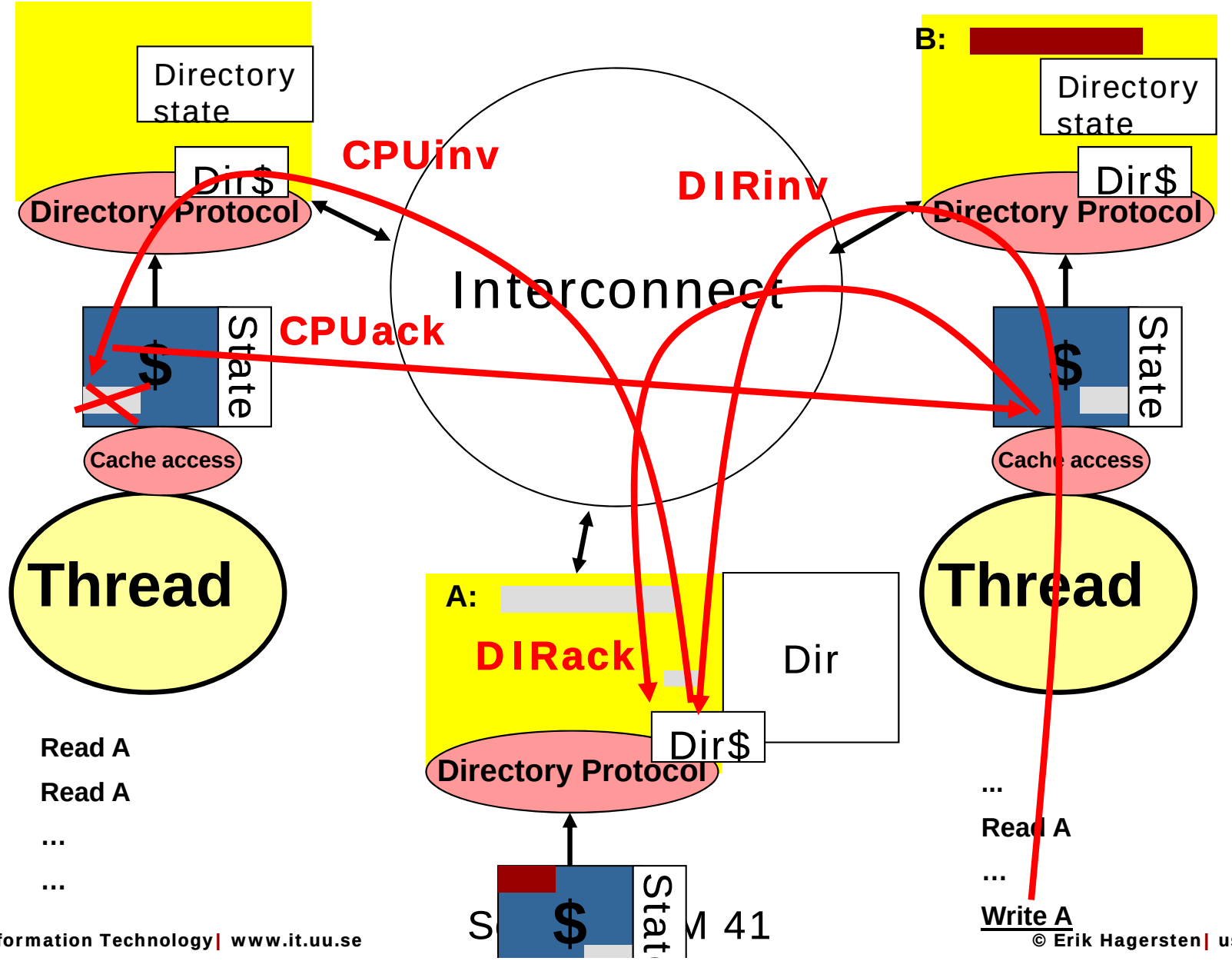
Cache-to-cache in snoop-based



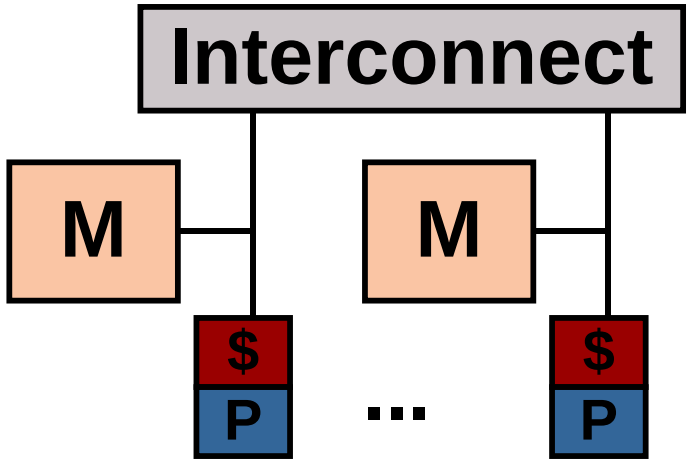
Cache-to-cache in dir-based



Directory cache & distribution

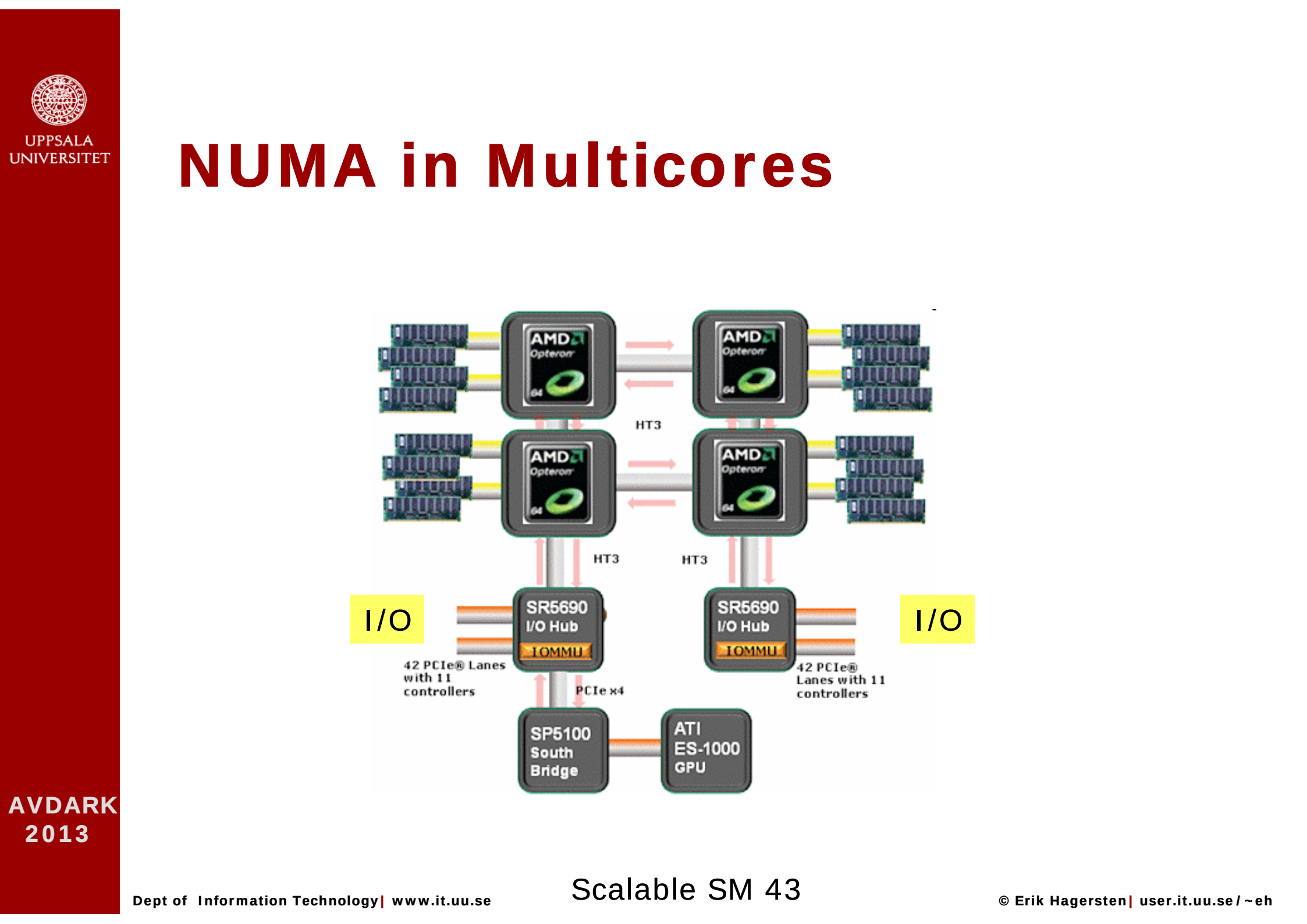


cc-NUMA issues



- Memory placement is key!
- Gotta' migrate NUMA home to where data is used (copy page and change VA → PA)
- Migrate thread that communicate with each other closer

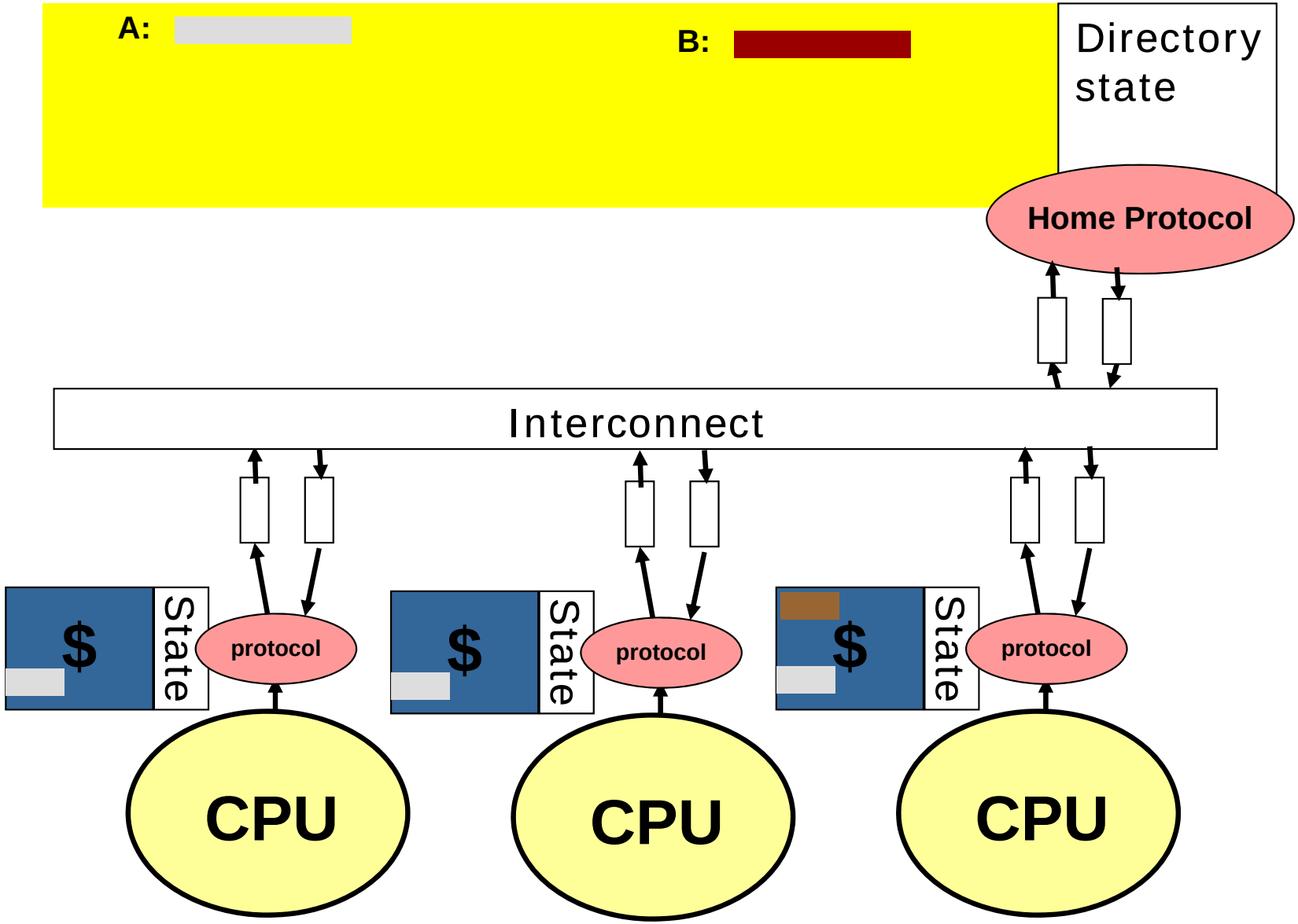
- Today, OS:es often use first-touch placement policy and places a physical page in the node where it was first accessed, but rarely migrate a page.



NUMA in Multicores

Directory protocols

Directory-based coherence



UPPSALA UNIVERSITET

Common Cache States

- M – Modified

  My dirty (i.e. modified) copy is the only cached copy
- E – Exclusive

  My clean copy is the only cached copy
- O – Owner

  I have a dirty copy, others may also have a copy
- S – Shared

  I have a clean copy, others may also have a copy
- I – Invalid

  I have no valid copy in my cache

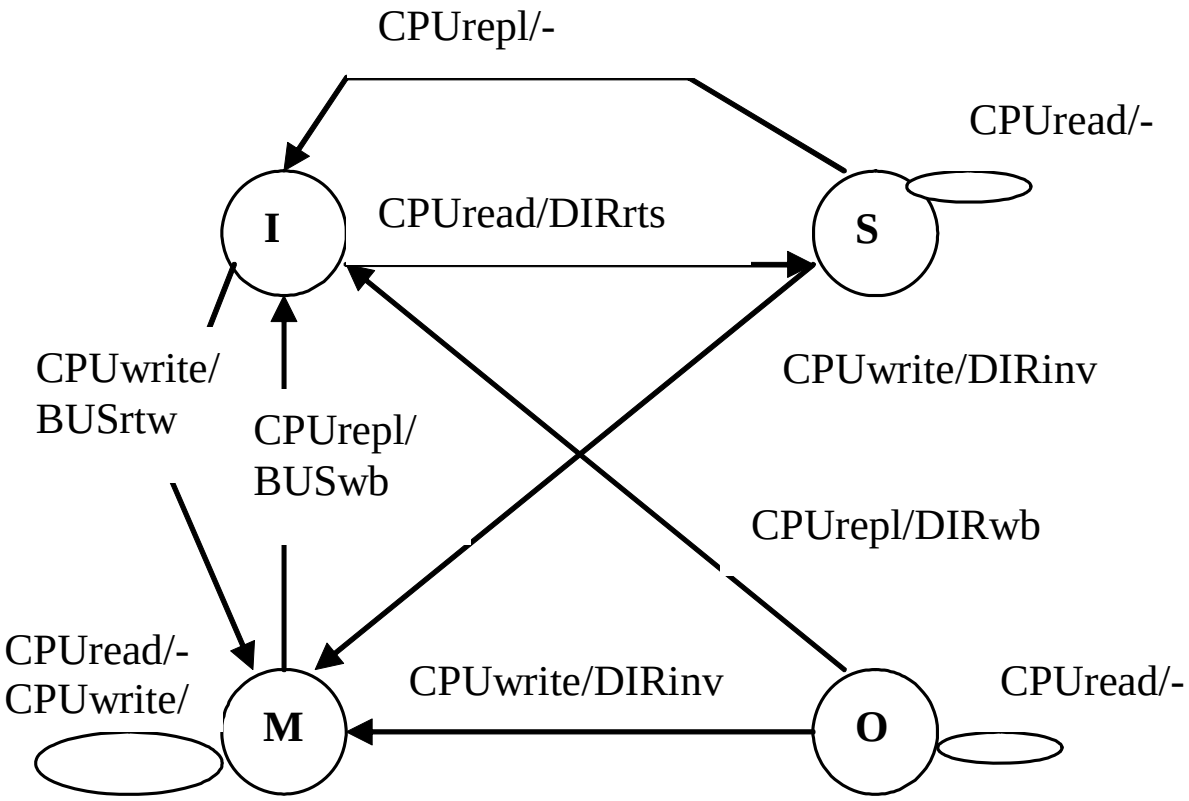
AVDARK 2013

UPPSALA UNIVERSITET

Some Coherence Alternatives

- MOSI
  - Leave one dirty copy in a cache on a cache2cache transfer
- MSI
  - Writeback to memory on a cache2cache.
- MOESI
  - The first reader will go to E and can later become a writer cheaply
- MESI
  - The first reader will go to E and can later become a writer cheaply. Writeback to memory on a cache2cache.

Part of the CPU protocol
In principle the same as snooping



FROM MY CPU:

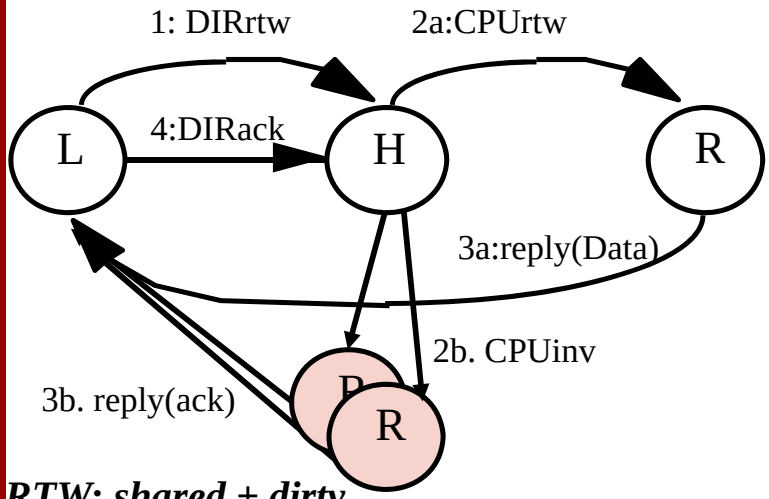
CPUread Caused by a Load instruction

CPUwrite: Caused by a Store or Atomic instruction

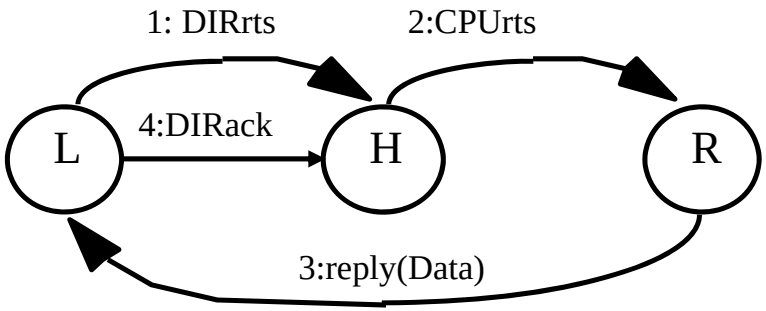
CPUrepl: Caused by a replacement of this cachline (caused by murphy ☺)

A few protocol examples

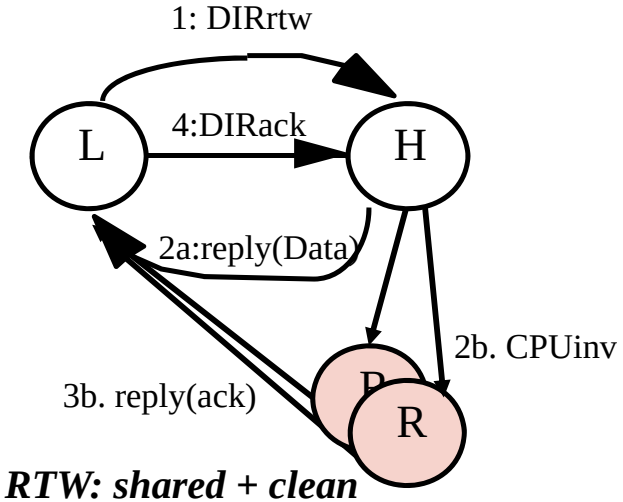
- L: The local ”requesting” CPU
- H: The home node with the directory
- R: A remote CPU node with a cached copy



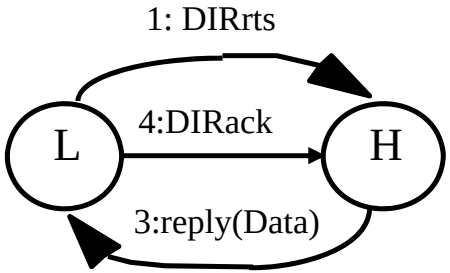
RTW: shared + dirty



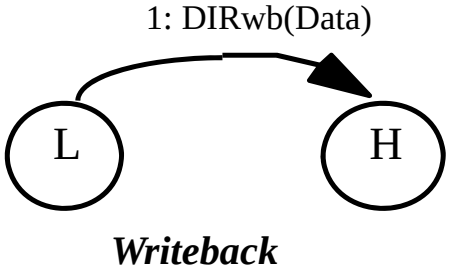
RTS: Remote dirty



RTW: shared + clean



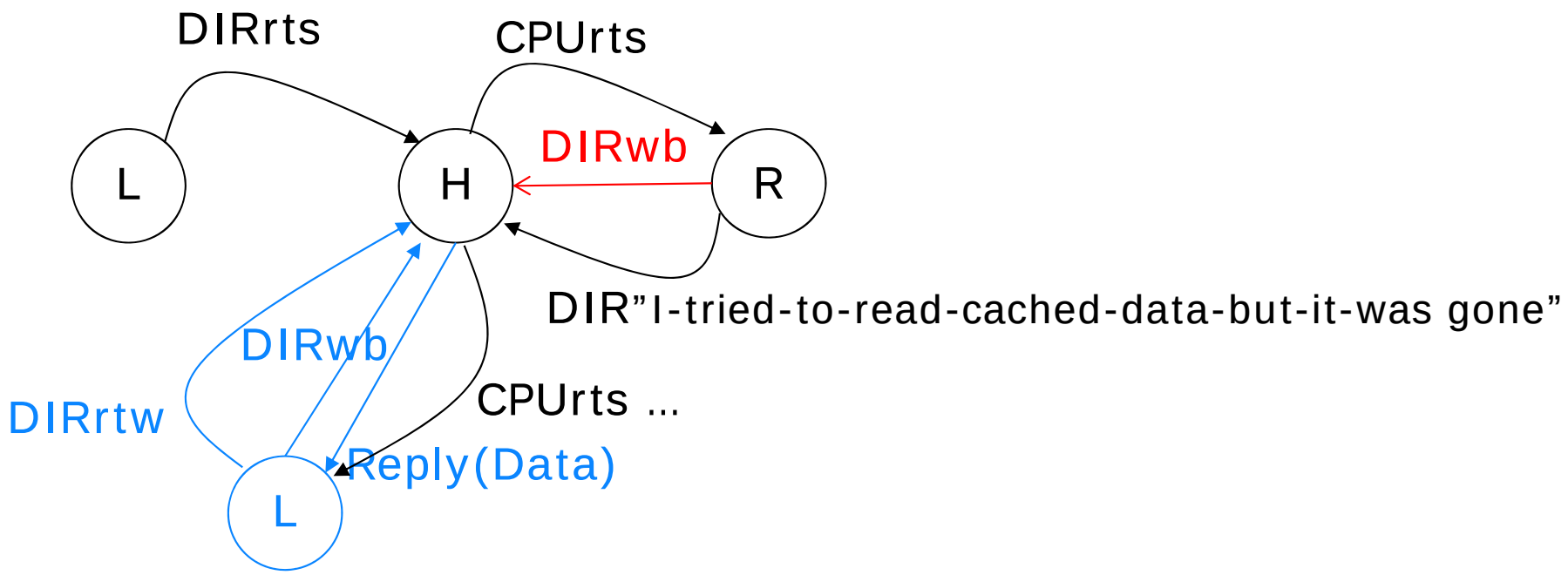
RTS: Home clean



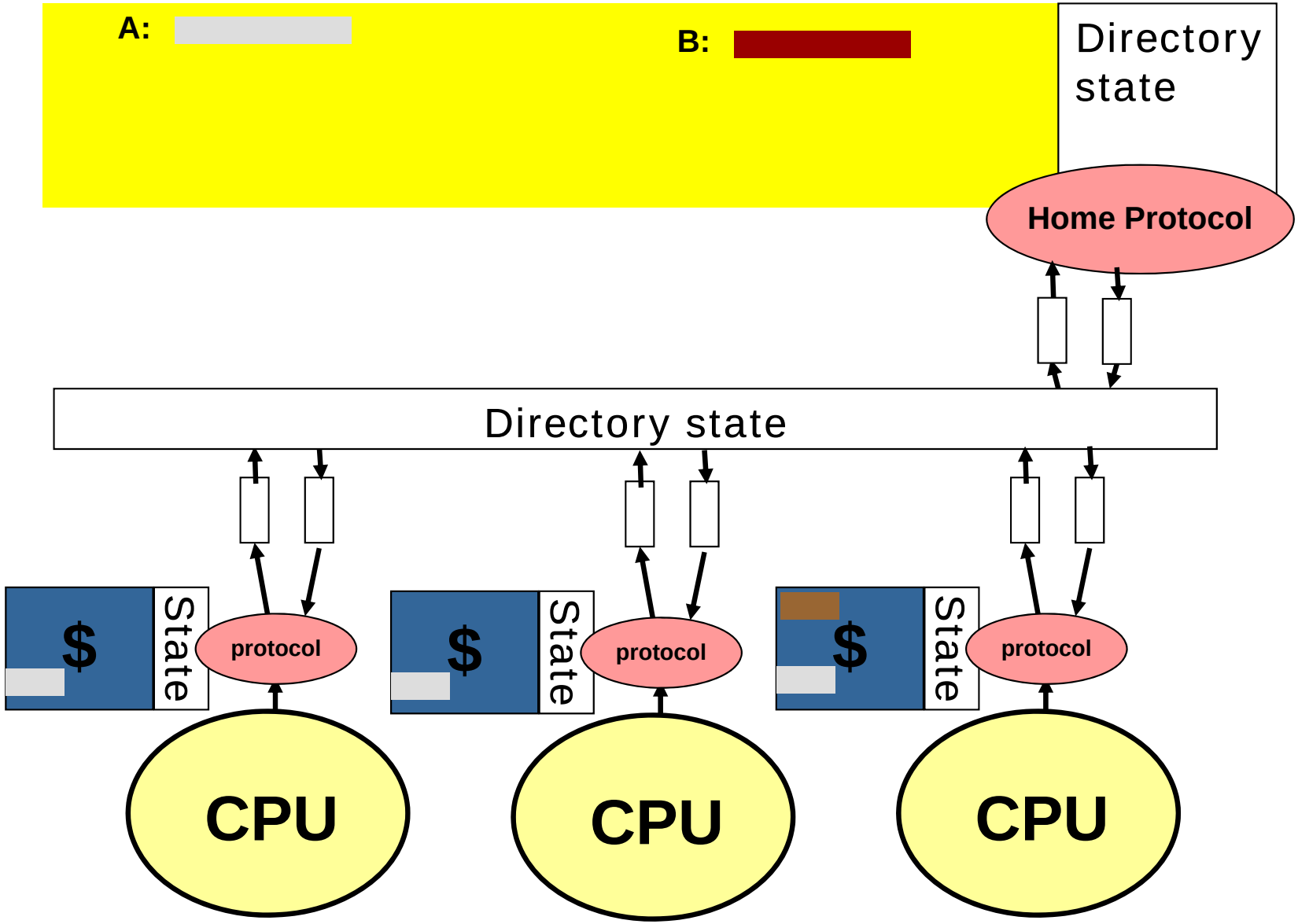
Writeback

Interesting corner cases…

- **Here RTS, dirty in remote cache**



Directory-based coherence



UPPSALA UNIVERSITET

Directory protocol

- Often specified by code examples
- Many interesting corner cases
- Often uses ”transient states”

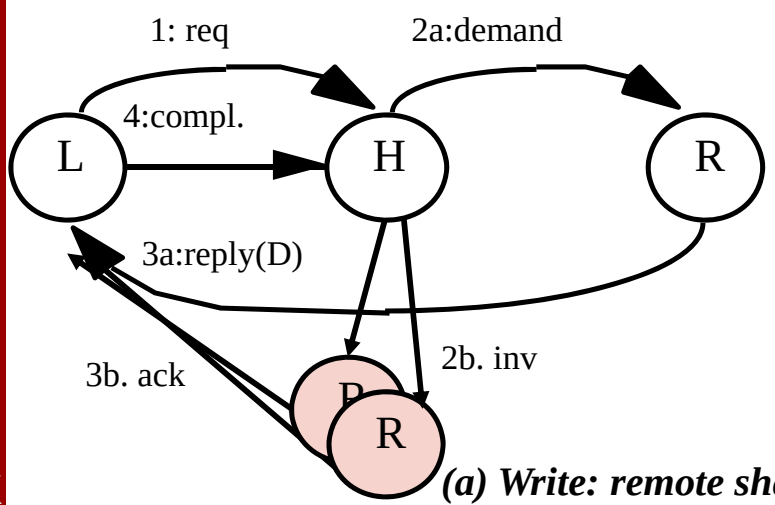
EX: ST to state S: Send INV goto ”SM”

AVDARK 2013

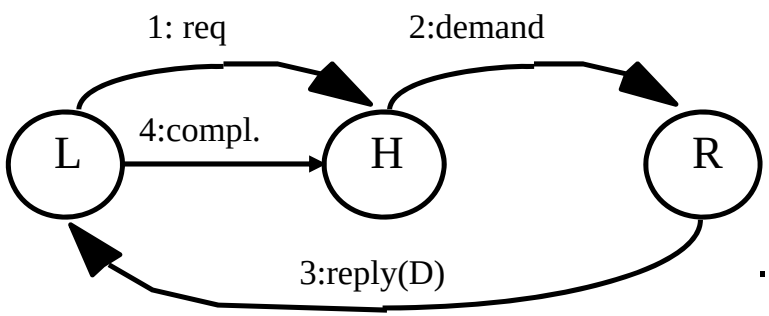
Deterministic Directory (WildFire)

No explicit transient states

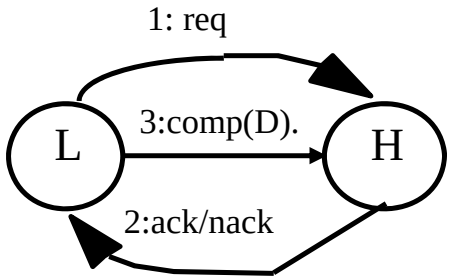
- Directory blocking: one outstanding trans/cache line
- New requests are blocked until a completion is received
- The directory state and cache state are always in agreement (except for silent replacement: S → I)
- (Optimization for sync: RTW can bypass RTS at directory)



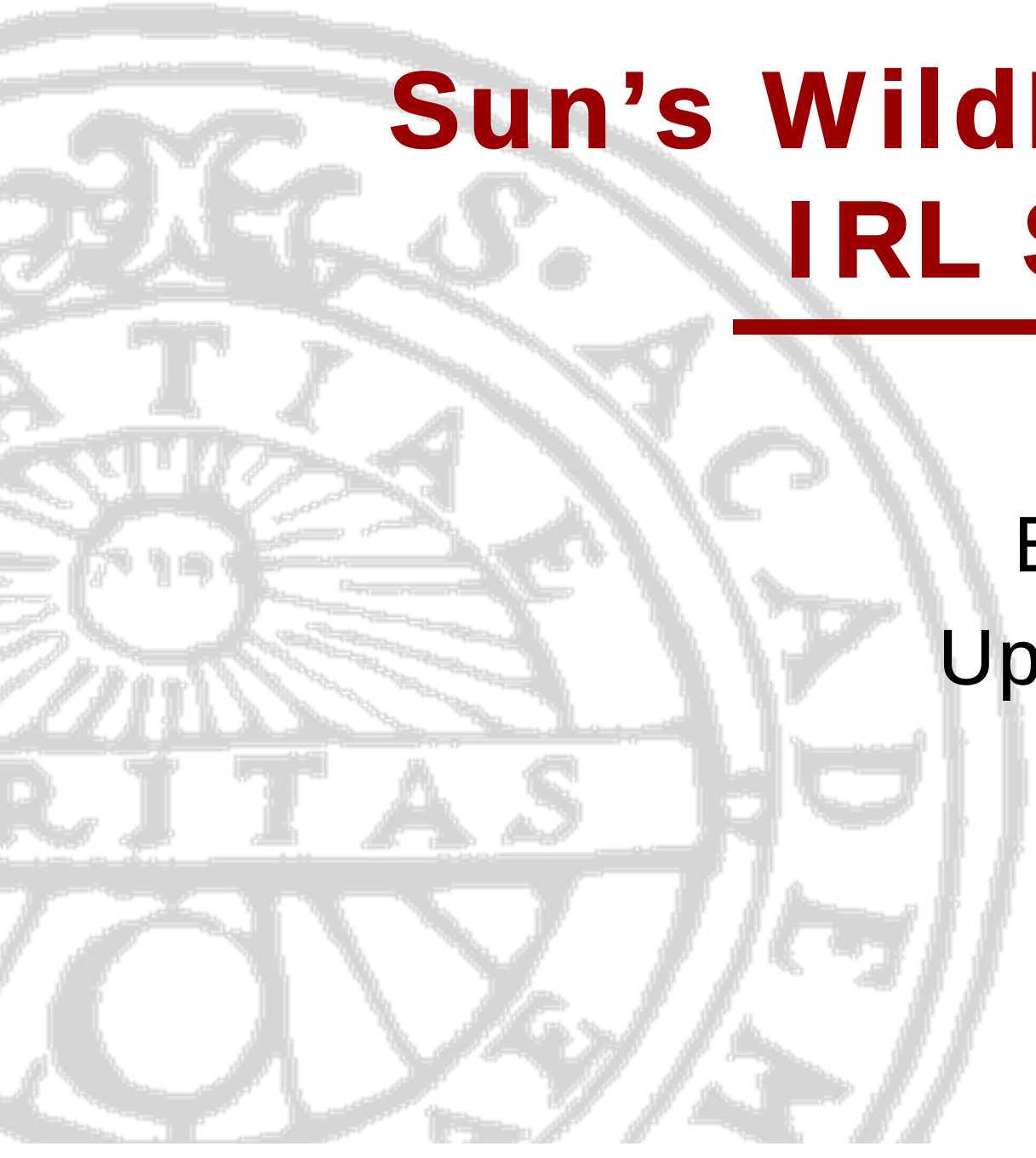
(a) Write: remote shared



(b) Read: Remote dirty



(c) Writeback



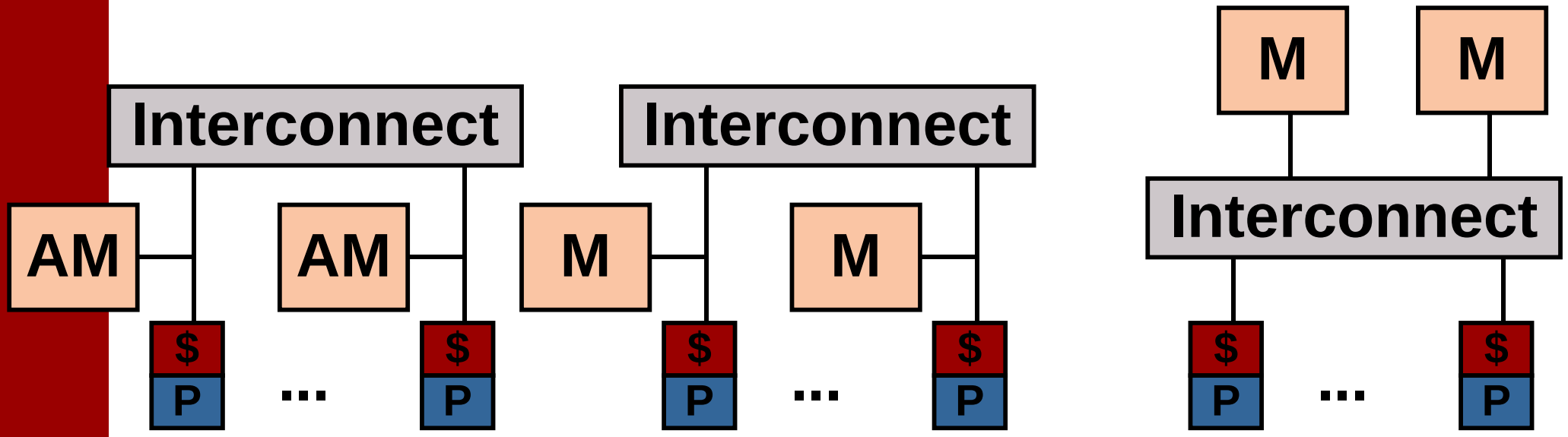
Sun's WildFire System IRL Show & Tell

Erik Hagersten
Uppsala University

WildFire:

One UNIX operating system spanning 4 nodes
Coherent shared memory

Three options for scalable shared memory



COMA
cache-only

NUMA
non-uniform

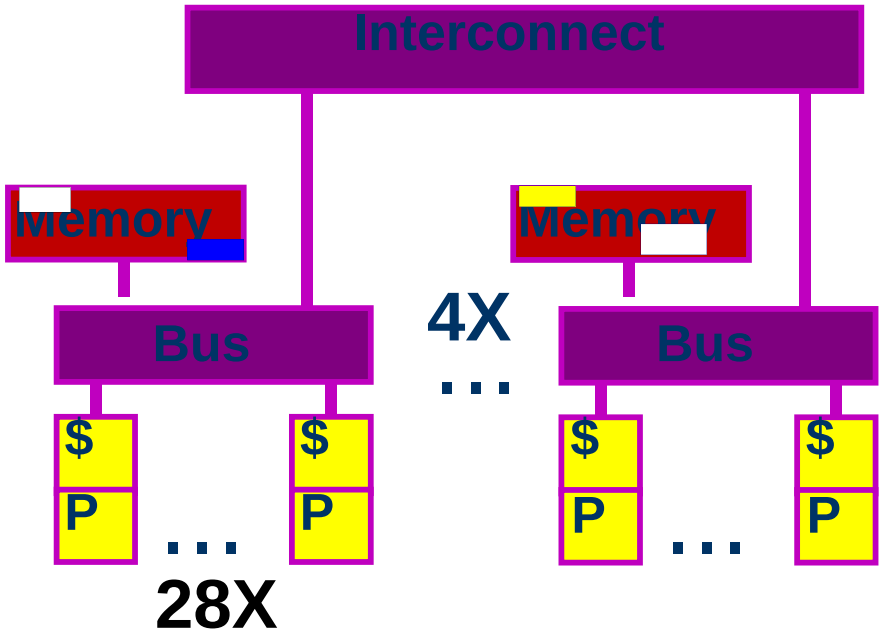
UMA
uniform
(a.k.a. SMP)

UPPSALA UNIVERSITET

Sun's WildFire System

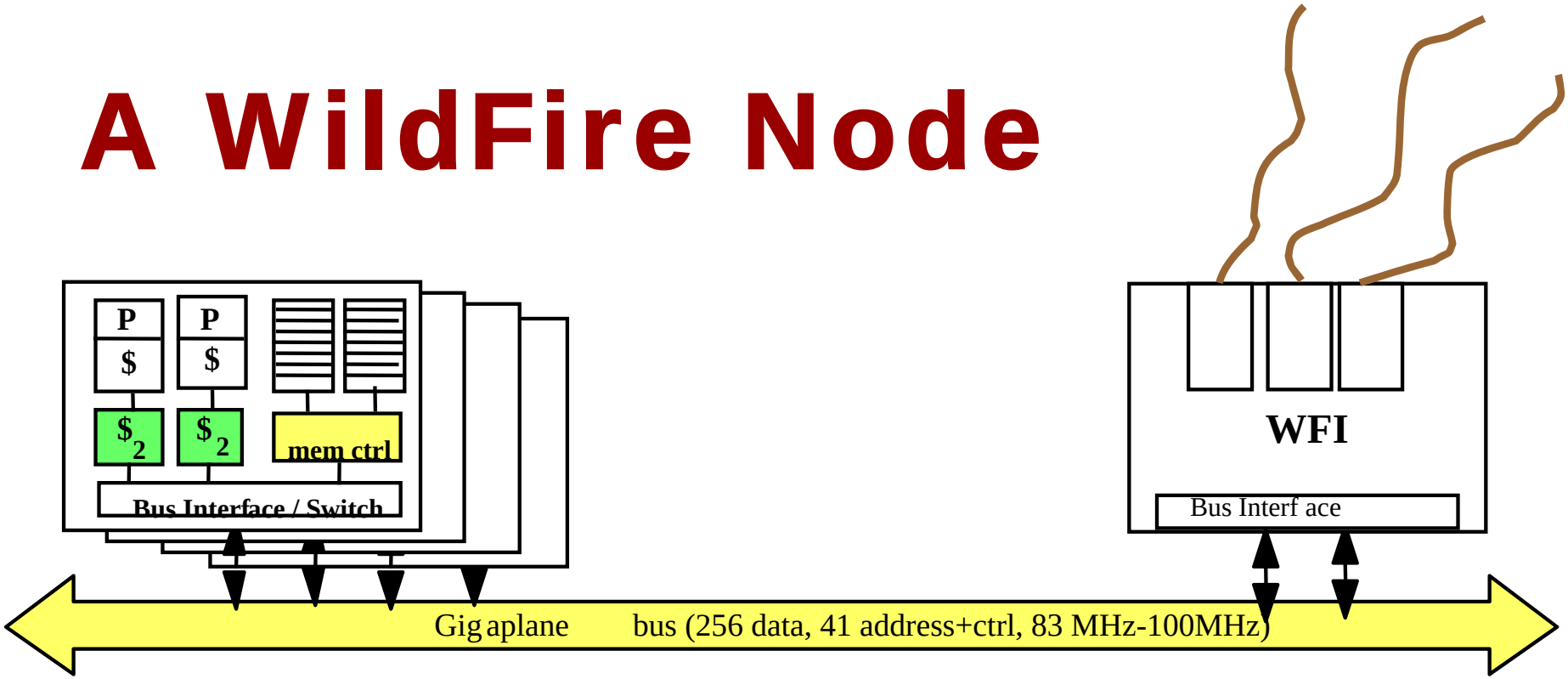
- Runs unmodified SMP apps in a more scalable way than E6000
- Minor modifications to E6000 snooping protocol
- CPUs generate local address OR global address
  - Global address --> no replication (NUMA)
  - Local address → Replication (~COMA)
- Coherent Memory Replication(~COMA)
- Hardware support for detecting migration/replication pages
- Deterministic directory implementation (easy to verify)

Adaptive S-COMA of Large SMPs



- Unified NUMA and COMA support
- A page may be replicated in many nodes
- HW maintains coherence per cache line for replicated pages
- Migration/Replication under SW control --> simple HW (S-COMA)
- Adaptive replication strategy in SW (R-NUMA)
- Fully connected point-to-point network
- Hierarchical affinity scheduler (HAS)

A WildFire Node

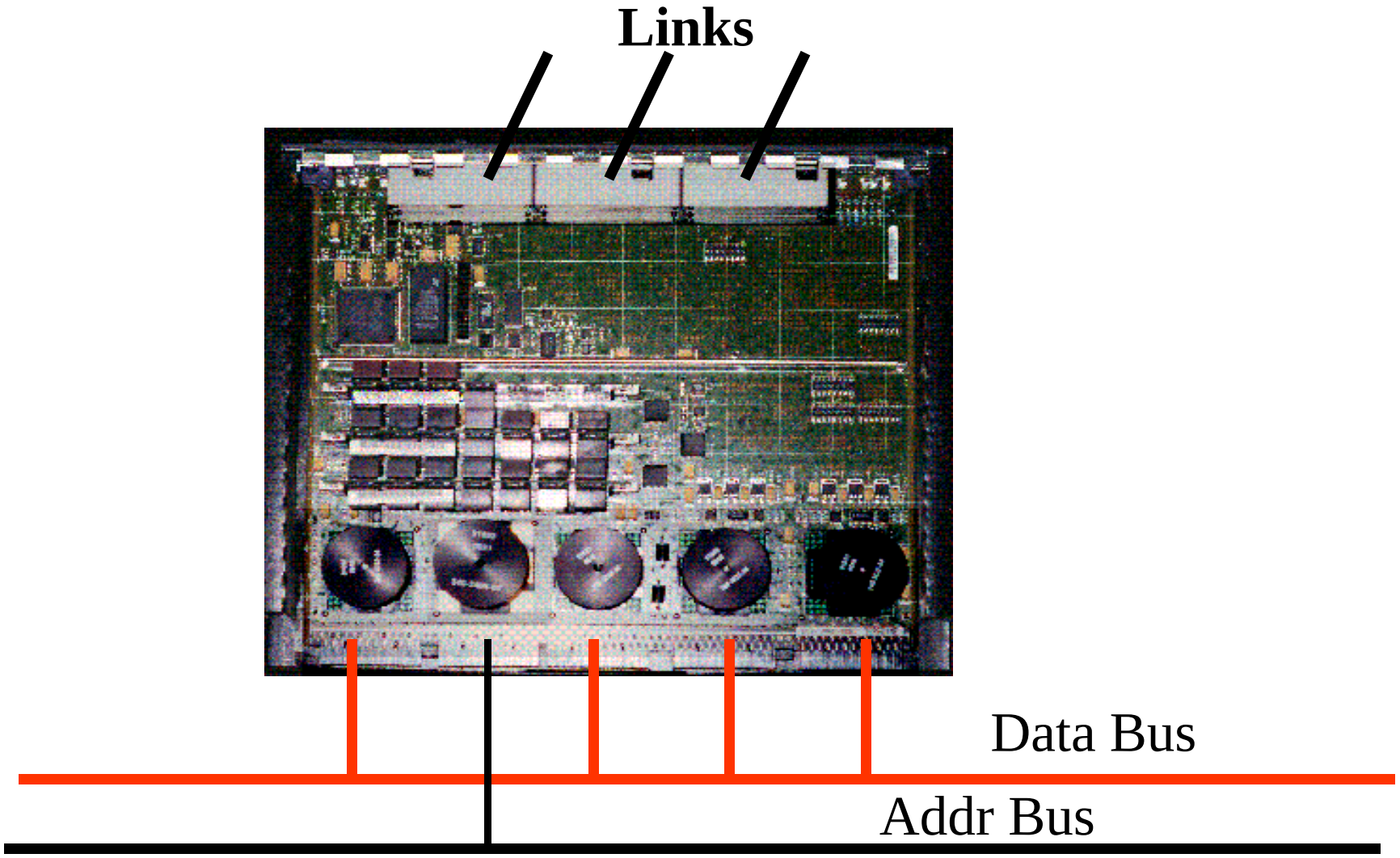


- 16 slots with either CPUs, IO or...

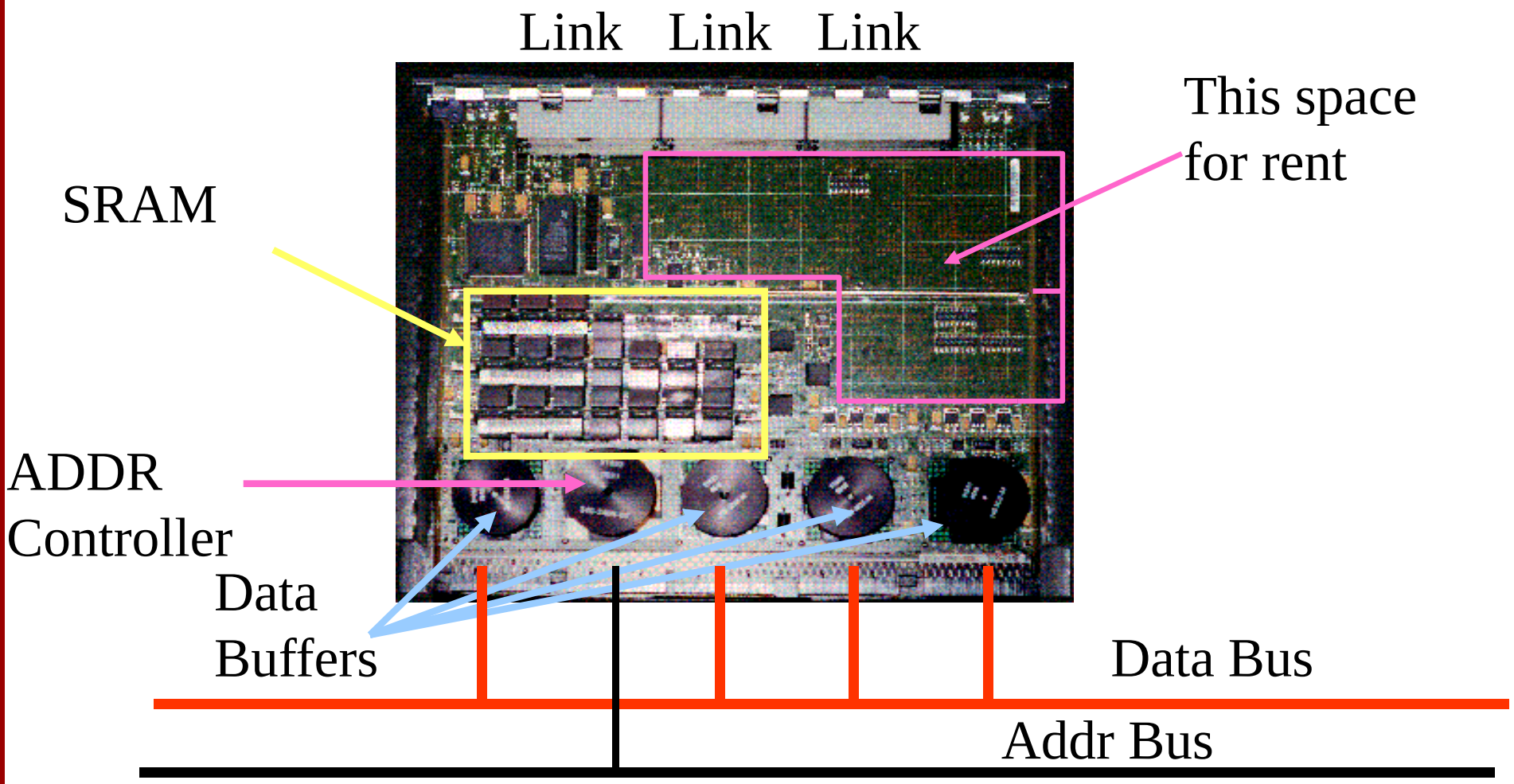
...WildFire extension board ➔

- Up to **28** UltraSPARC processors per node

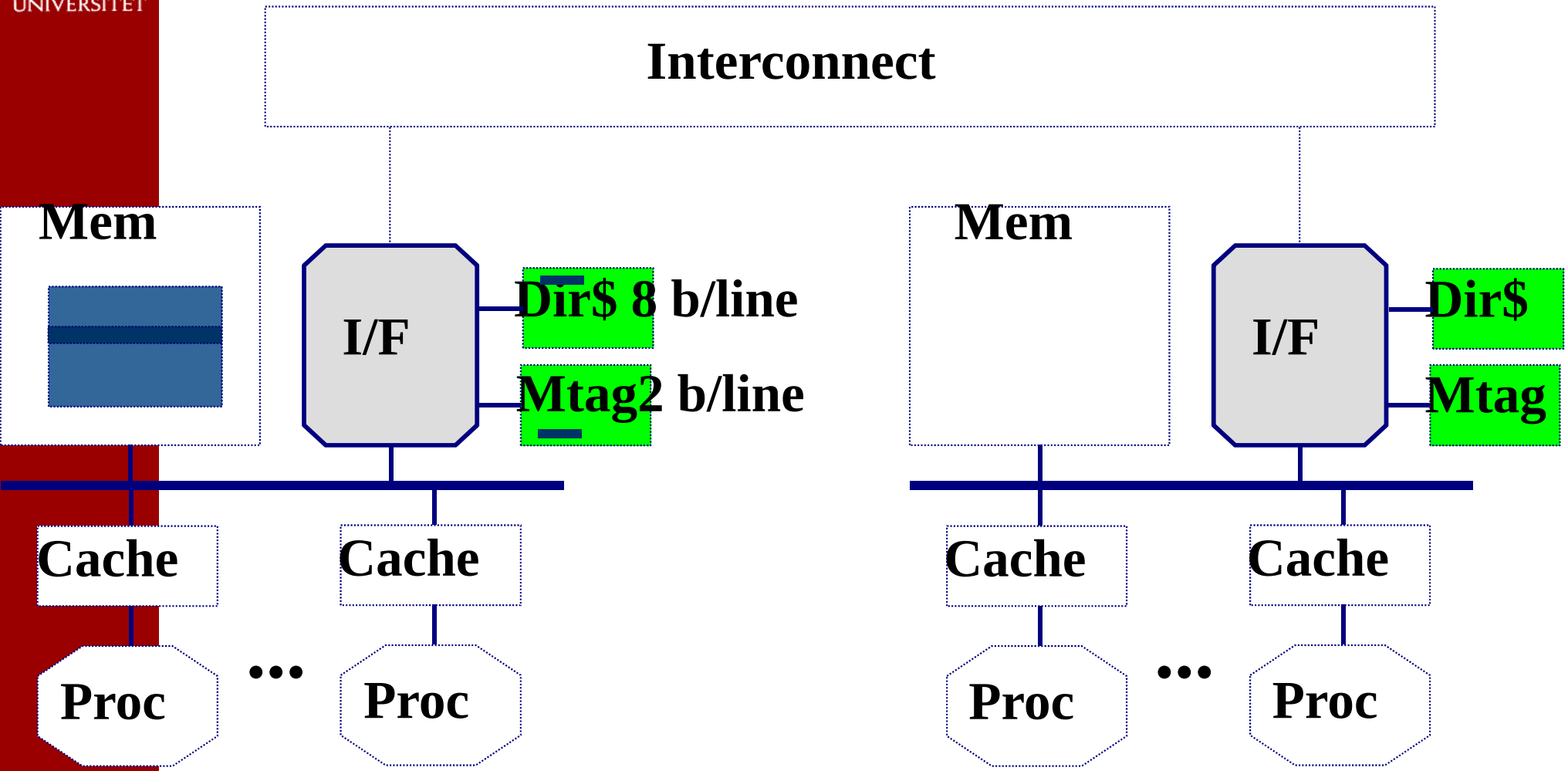
Sun WildFire Interface Board



Sun WildFire Interface Board



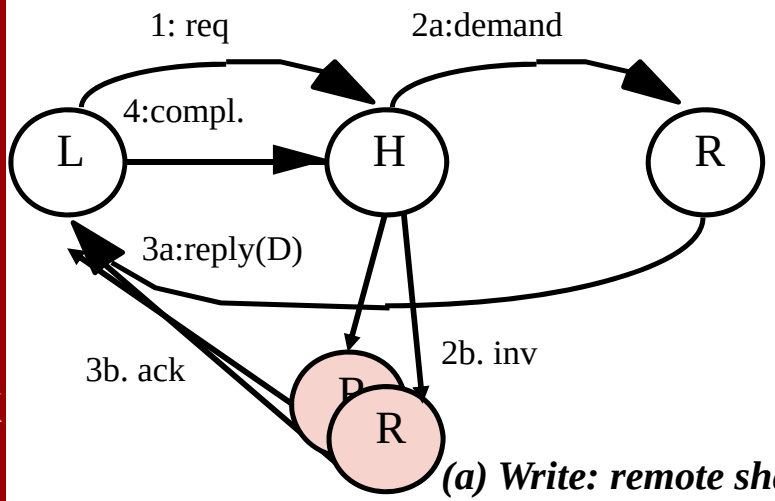
WildFire as a vanilla "NUMA"



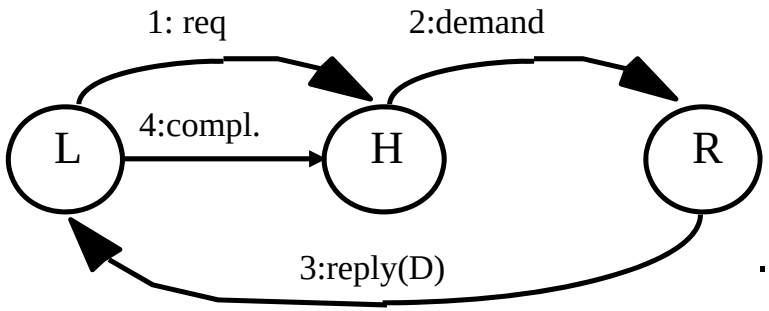
Deterministic Directory (WildFire)

No explicit transient states

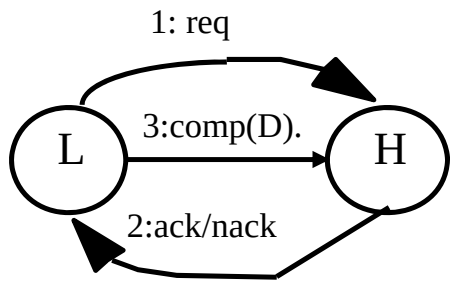
- MOSI protocol, fully mapped directory (one bit/node)
- Directory blocking: one outstanding trans/cache line
- Directory blocks new requests until completion received
- The directory state and cache state always in agreement (except for silent replacement of clean data S → I)
- What about contended spin locks? Aren't you serializing?
  - Writes can bypass reads while waiting for a blocked directory.



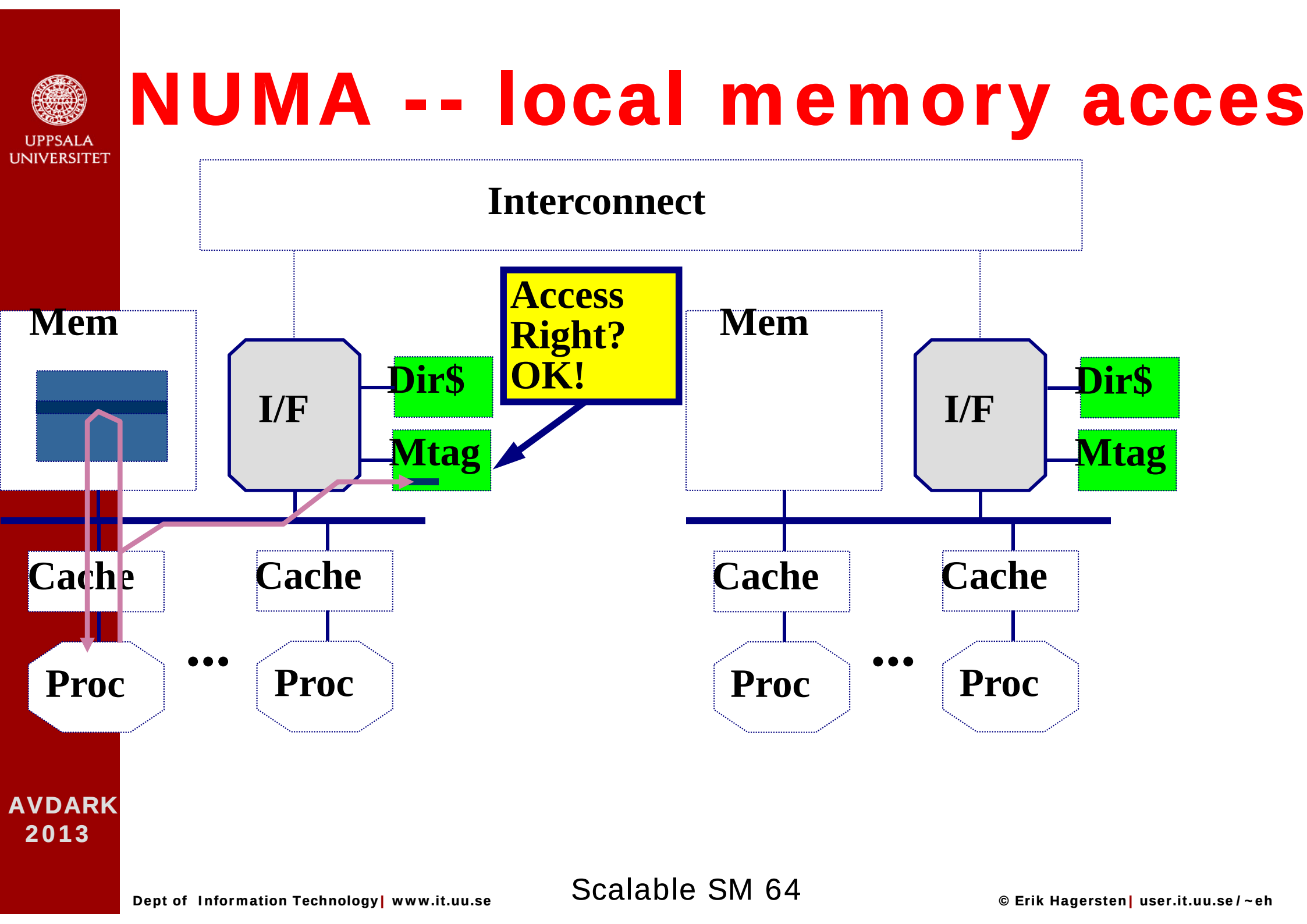
(a) Write: remote shared



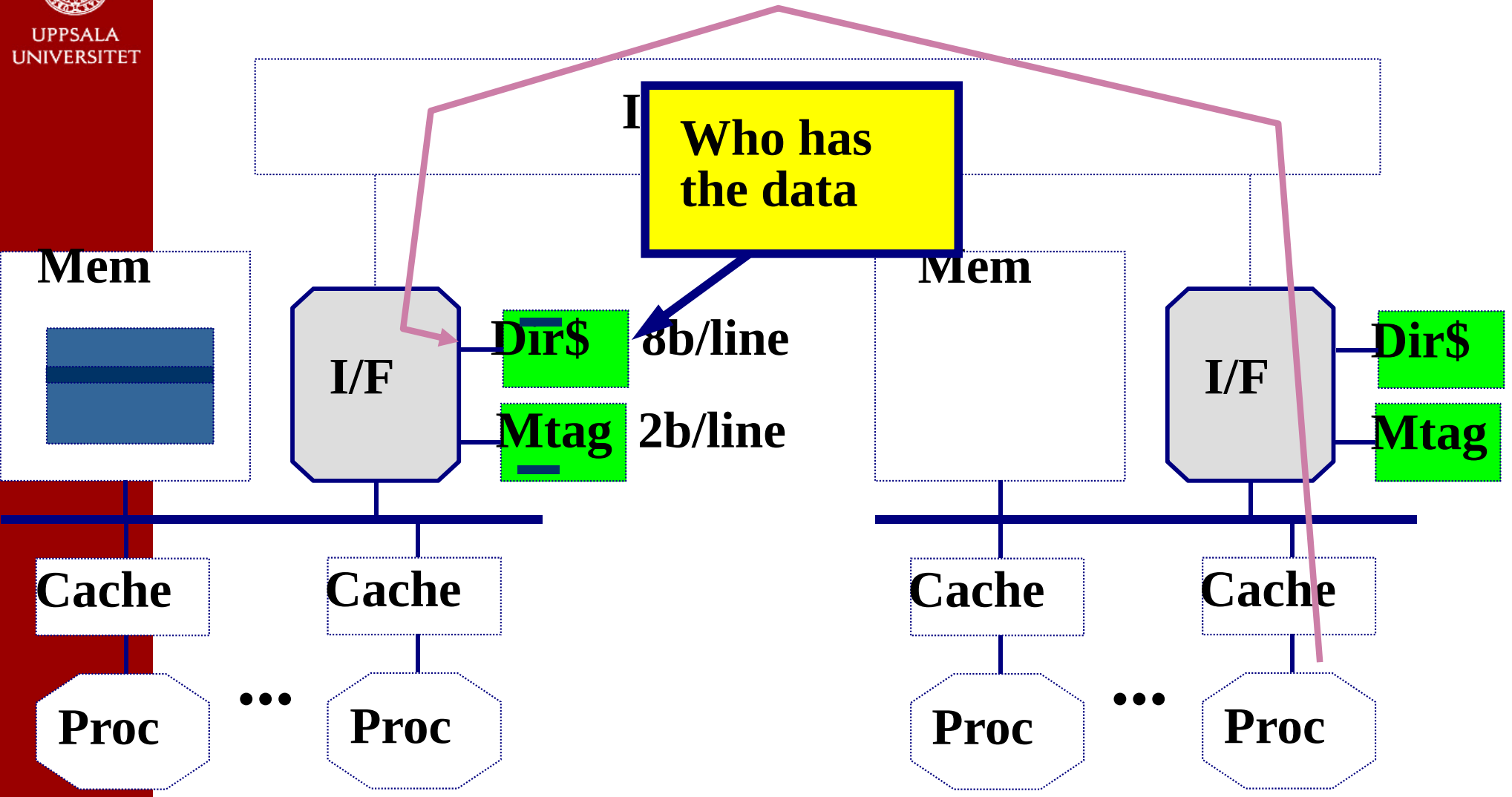
(b) Read: Remote dirty



(c) Writeback

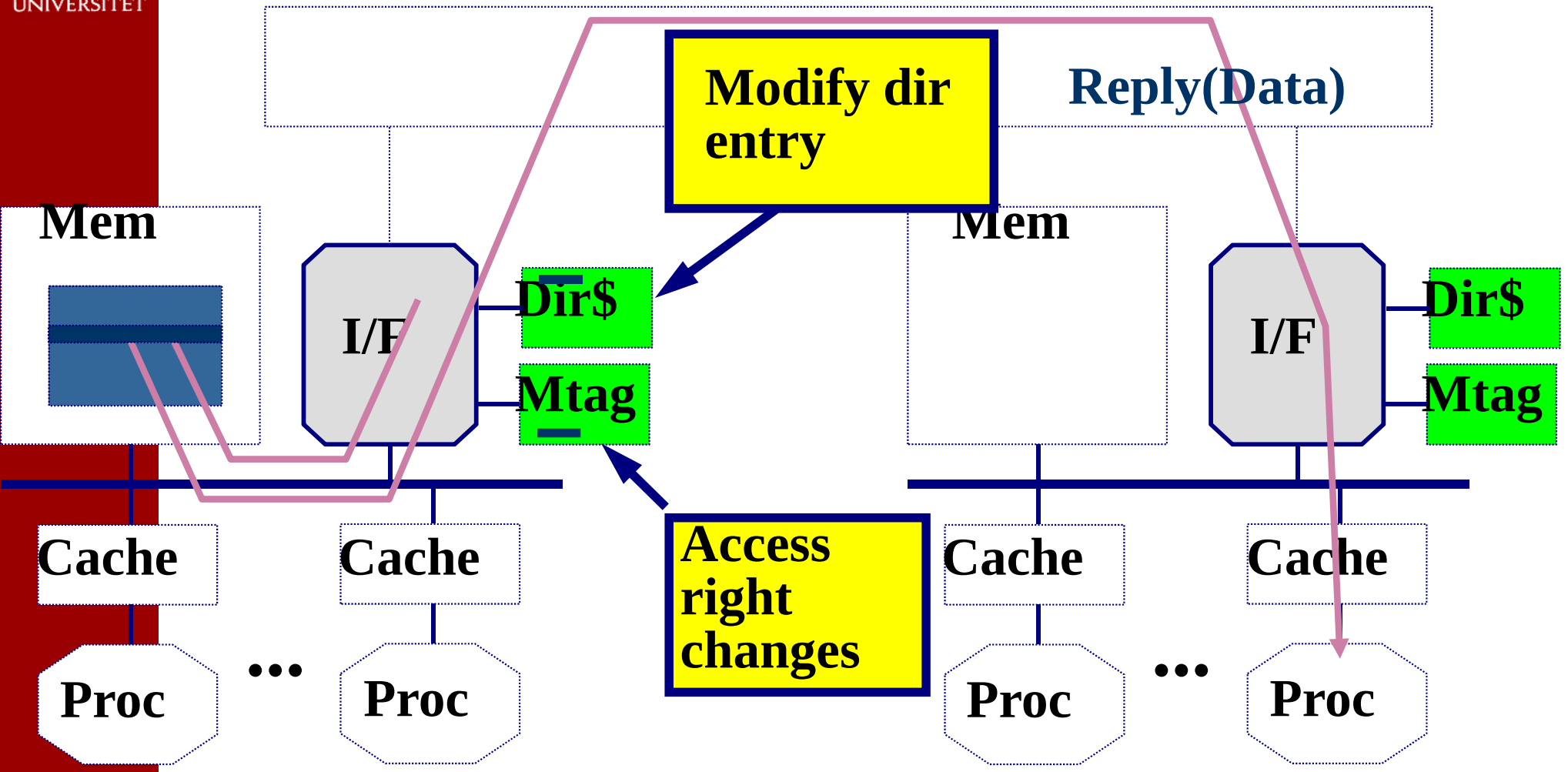

UPPSALA UNIVERSITET
NUMA -- local memory acces
Interconnect
Access Right? OK!
Mem
I/F
Dir$
Mtag
Mem
I/F
Dir$
Mtag
Cache
Cache
Cache
Cache
Proc
Proc
Proc
Proc
AVDARK 2013
Dept of Information Technology| www.it.uu.se
Scalable SM 64
© Erik Hagersten| user.it.uu.se/~eh

NUMA -- remote memory access

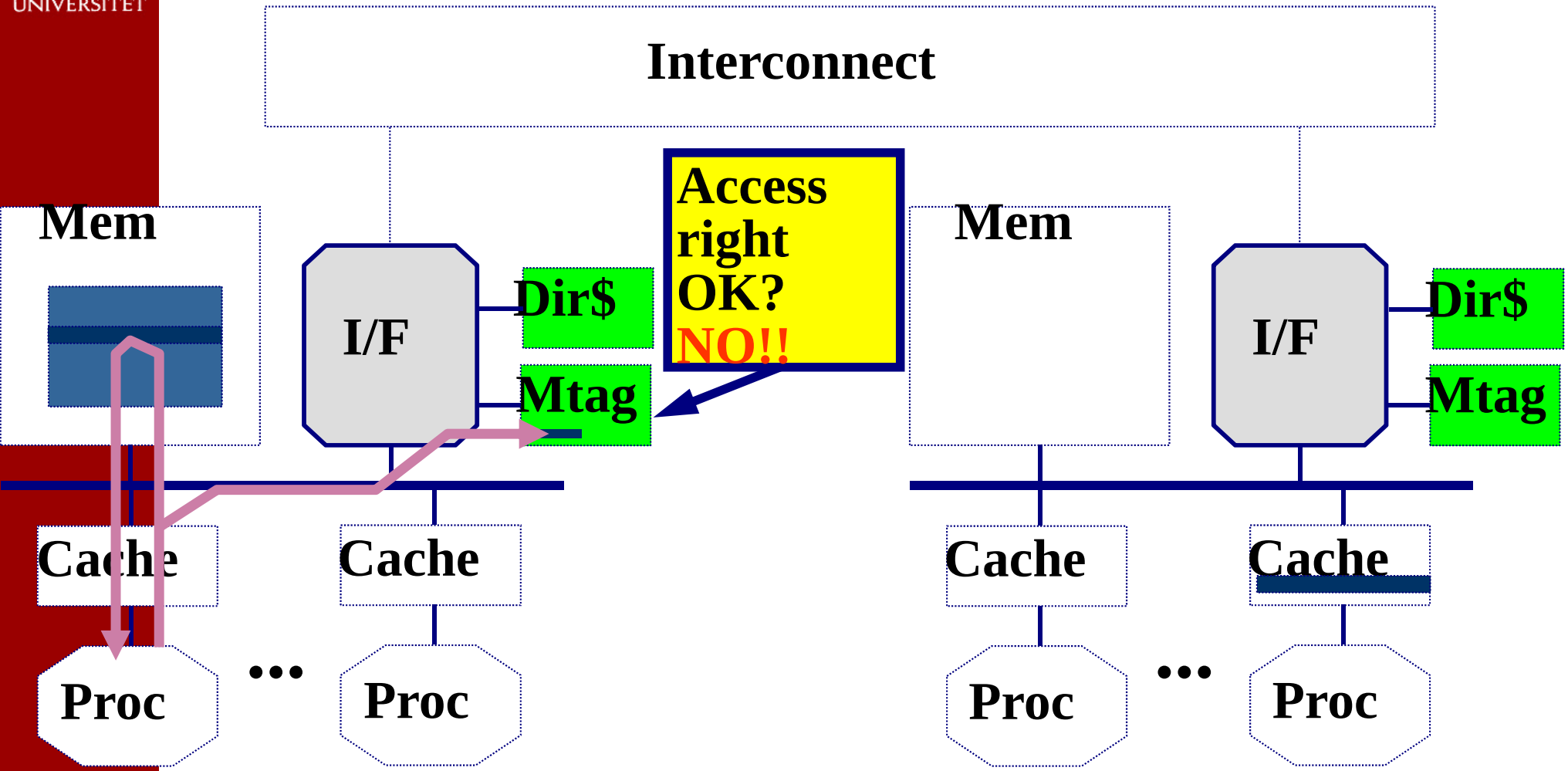


SRAM overhead = 10/512 = 2% (lower bound 2/512 = 0.4%)

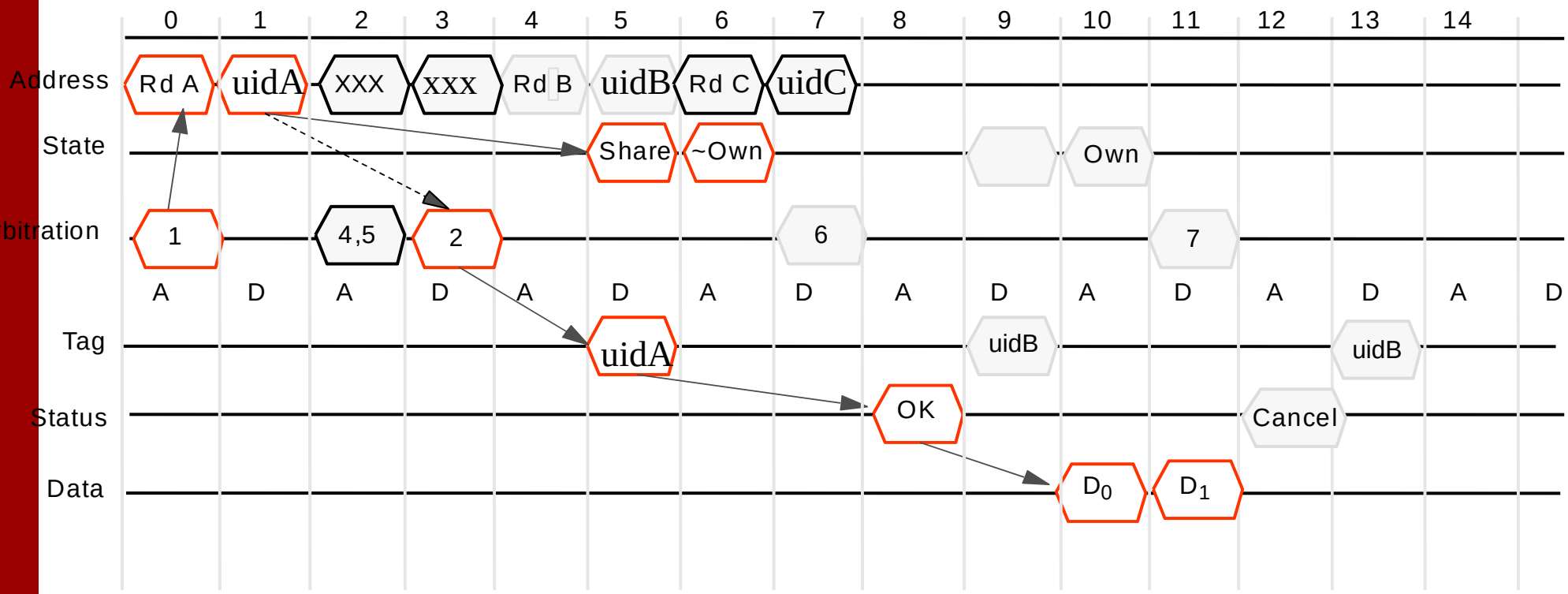
Global Cache Coherence Prot.



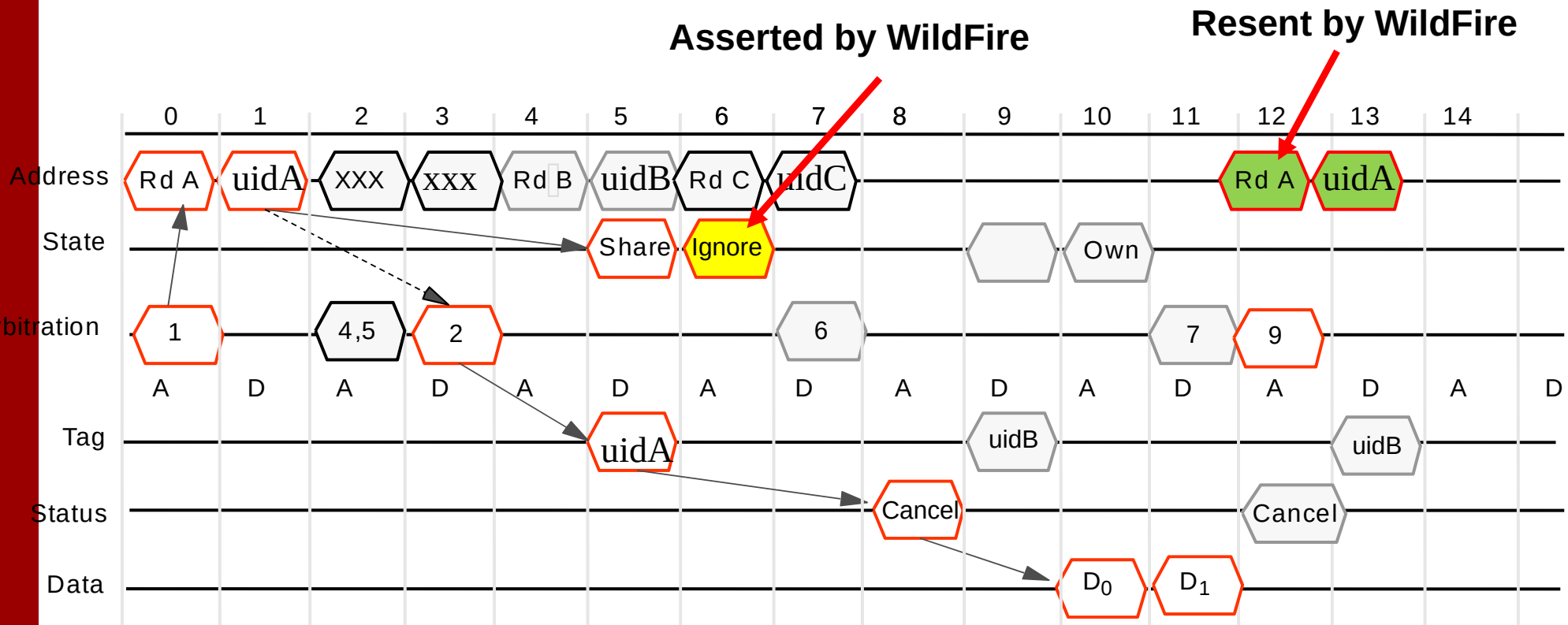
NUMA -- local memory access



Gigaplane Bus Timing



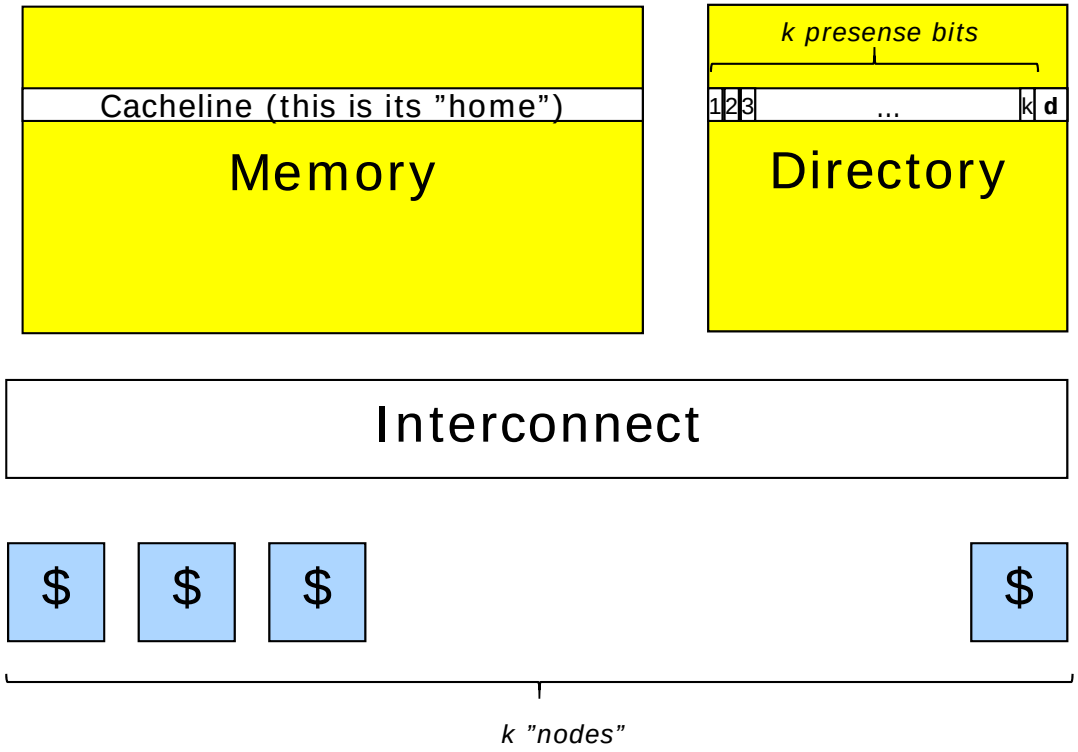
WildFire Bus Extensions



Ignore snoop-reply squashes an ongoing transaction => not put in IQ
WildFire eventually reissues the same transaction (w/ same uid)

(RTSF -- a new transaction sends data to CPU and memory)

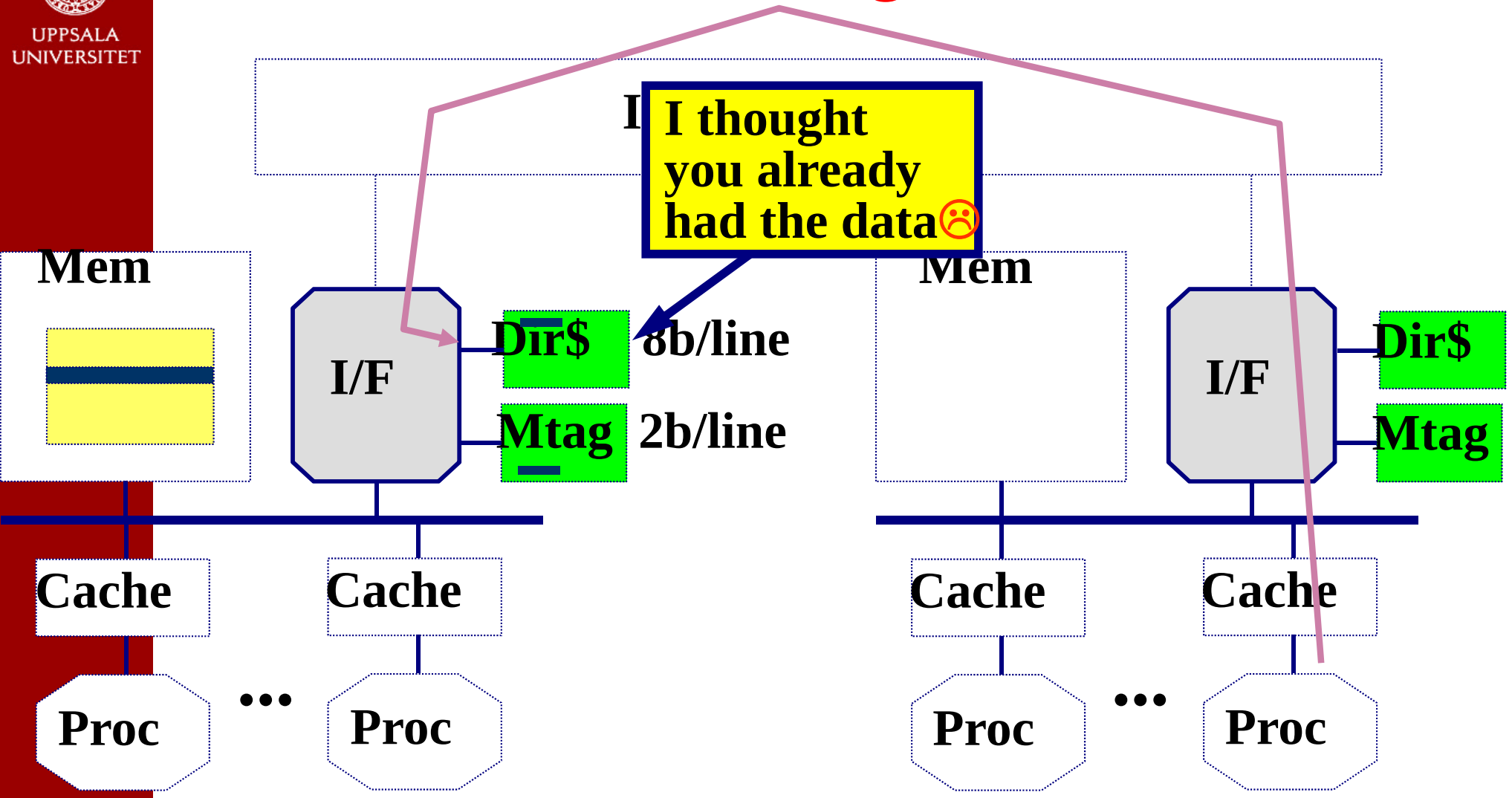
WildFire Directory: k = 4

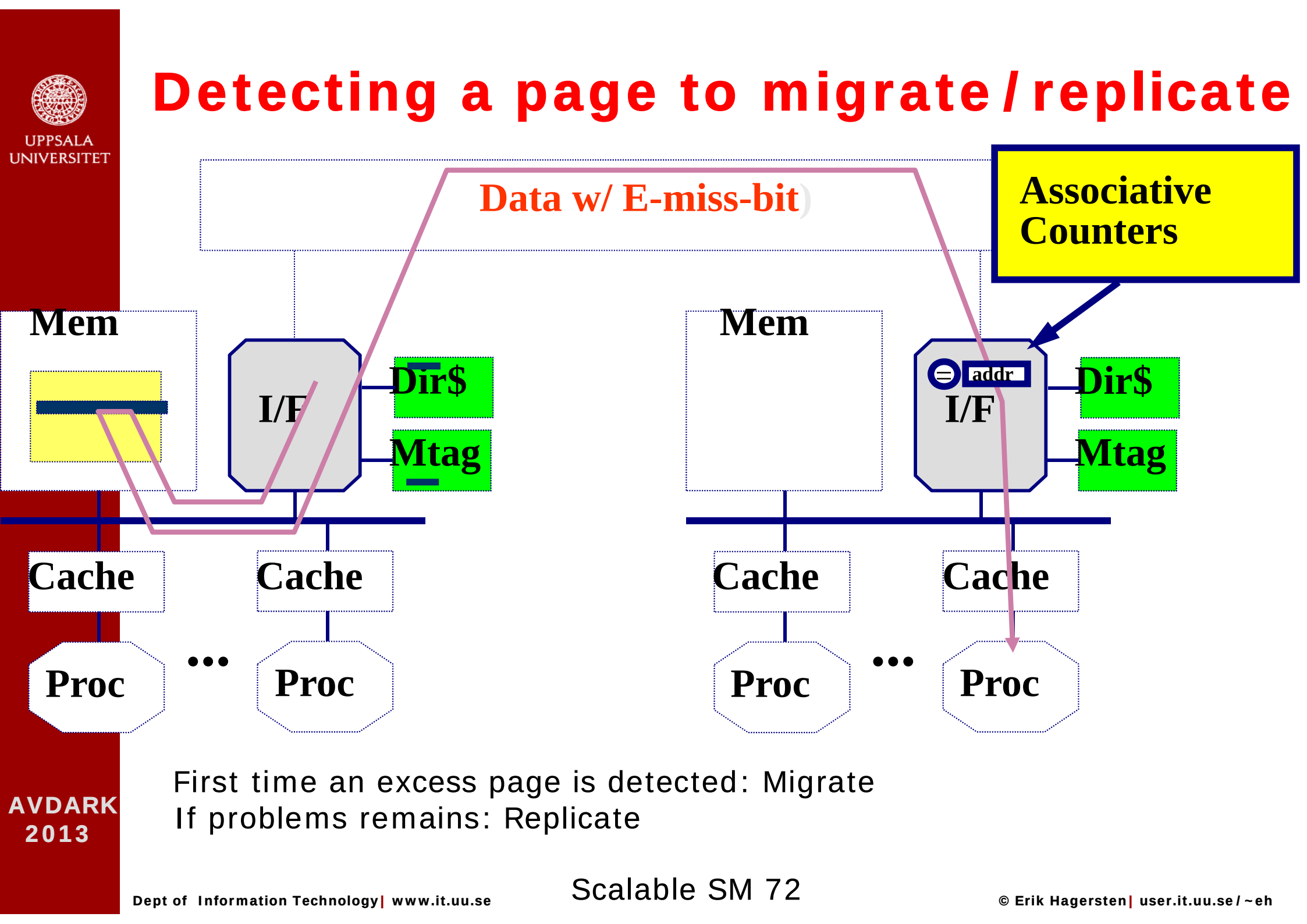


Fully mapped directory

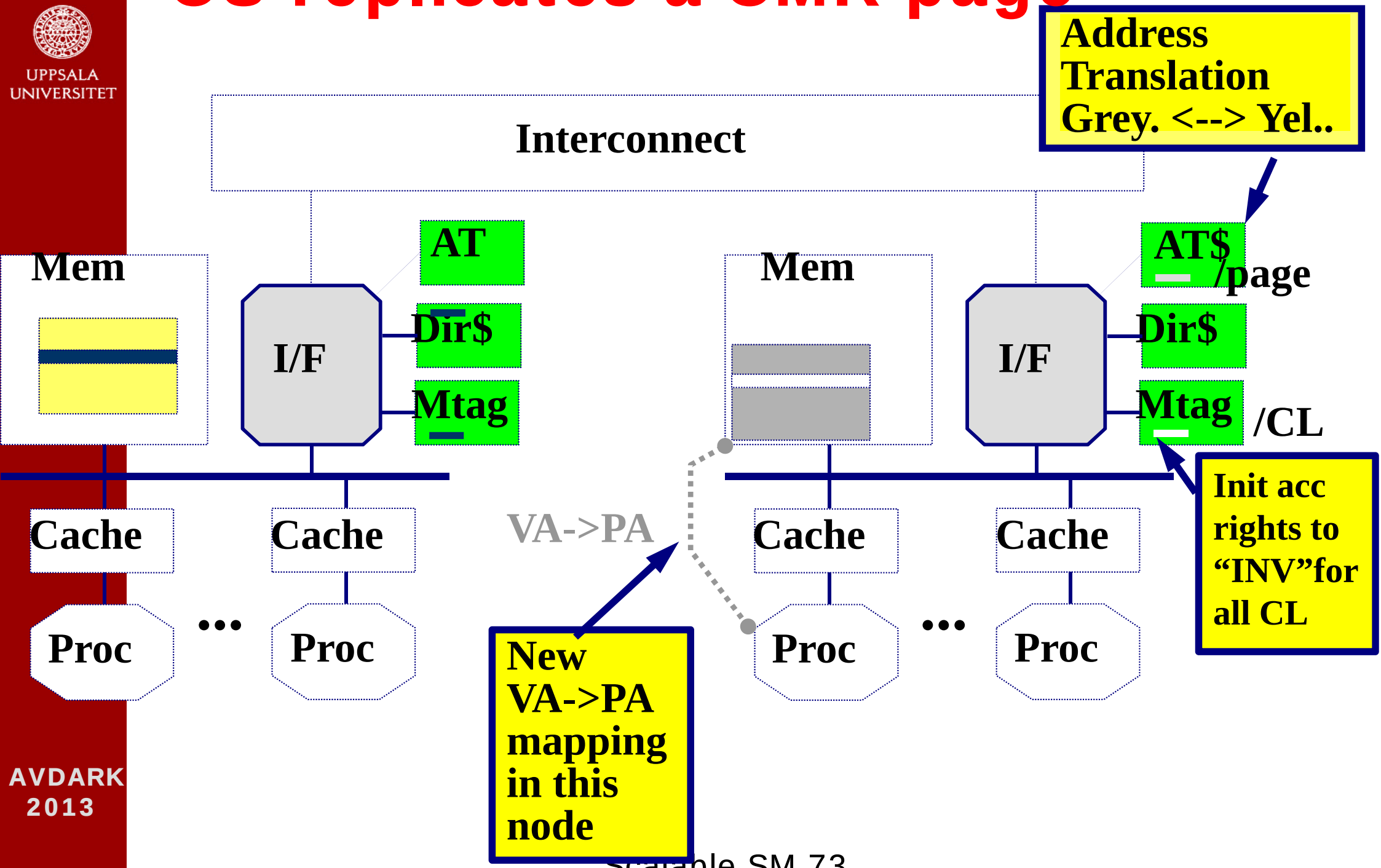
- ***k* Nodes**
- **Dir entry per cacheline in home memory: *k* presence-bits + 1 dirty-bit**

NUMA "detecting excess misses"

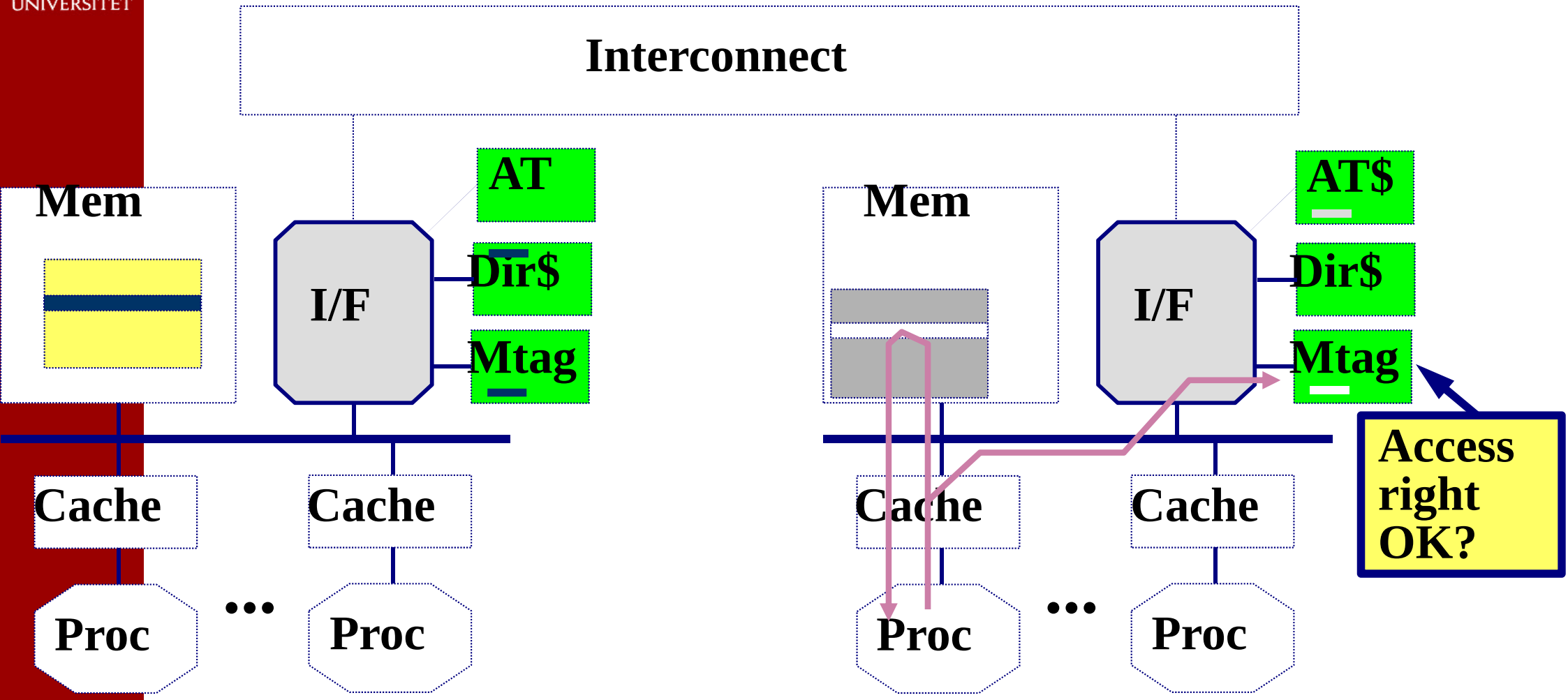



Detecting a page to migrate/replicate
UPPSALA UNIVERSITET
Data w/ E-miss-bit
Associative Counters
Mem
I/F
Dir$
Mtag
Mem
addr
I/F
Dir$
Mtag
Cache
Cache
Cache
Cache
Proc
Proc
Proc
Proc
First time an excess page is detected: Migrate
If problems remains: Replicate
AVDARK 2013
Dept of Information Technology| www.it.uu.se
Scalable SM 72
© Erik Hagersten| user.it.uu.se/~eh

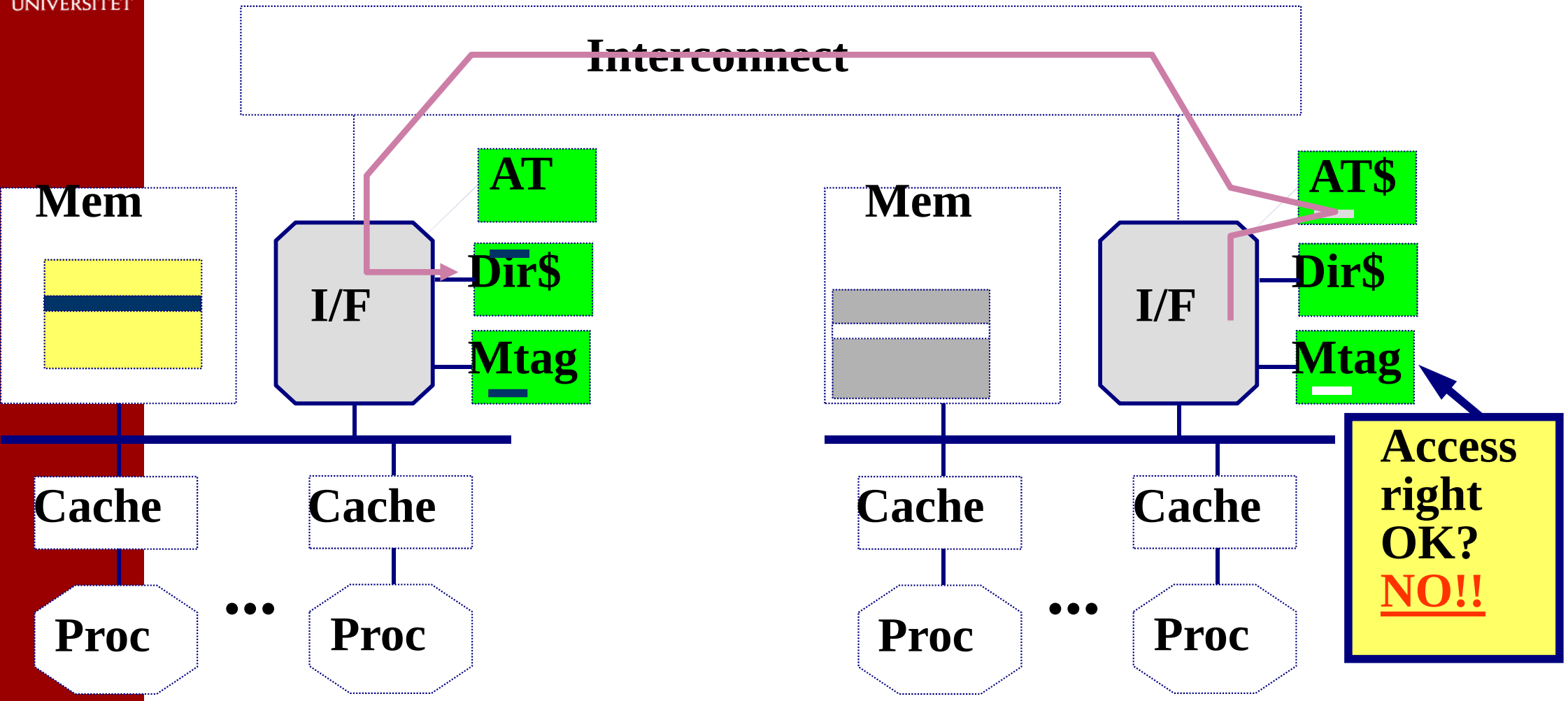
OS replicates a CMR page



An access to a CMR page

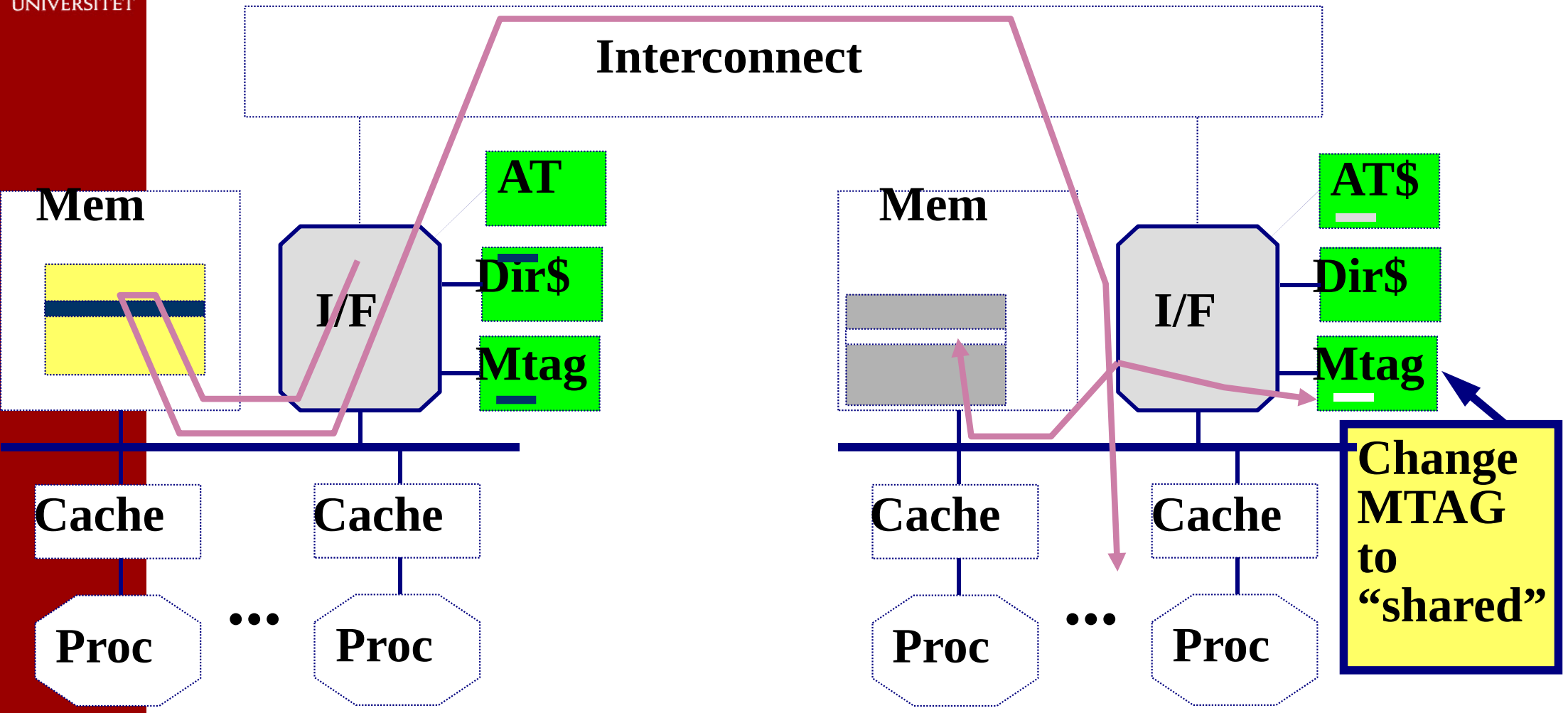


An access to a CMR page (miss)

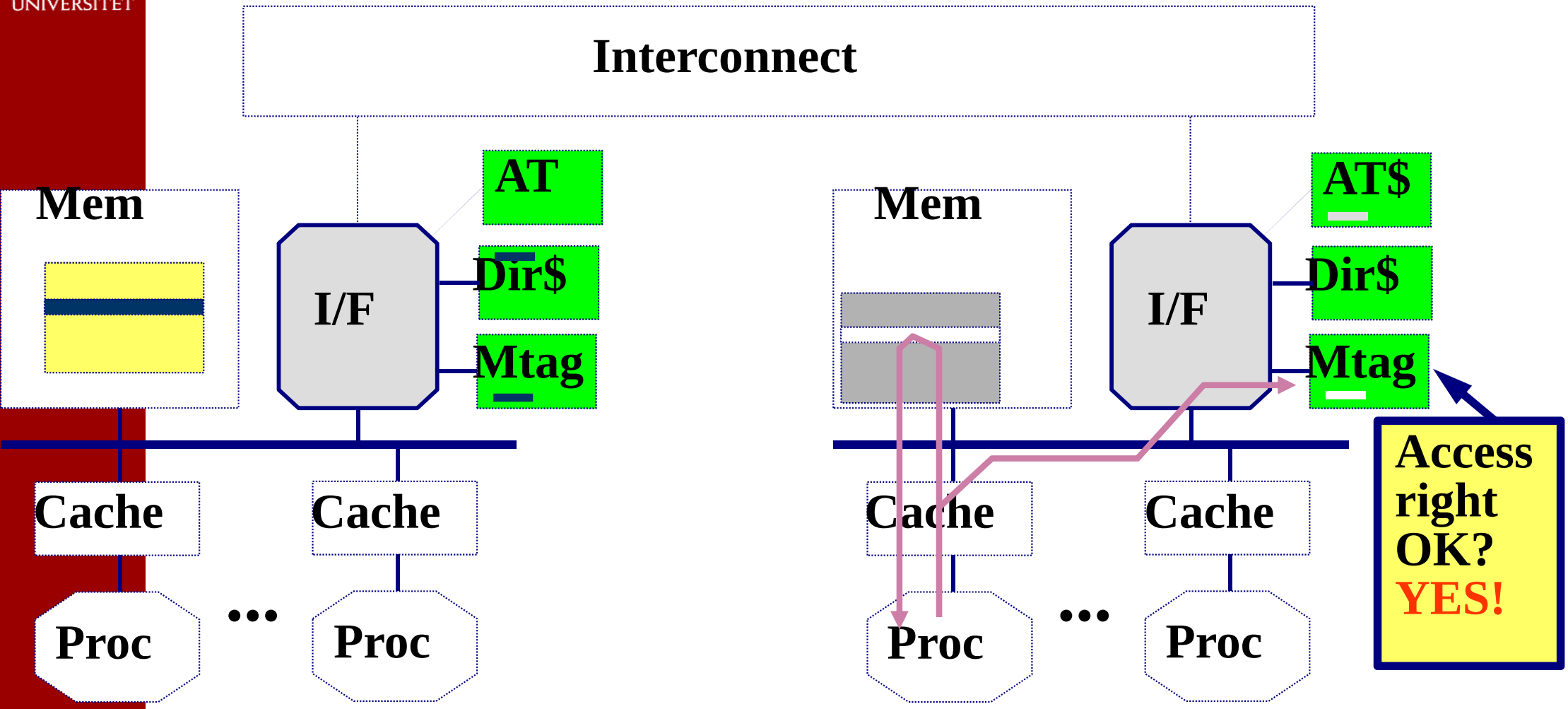


Address Translation (AT) overhead = 8B/8kB = 0.1%
No extra latency added

Data updating CMR and cache



An access to a CMR page (hit)



Does migration/replication help?

NAS parallel Benchmark Study (Execution time in seconds)

[M. Bull, EPCC at EWOMP 2002]

Shallow

	No Initial Plac.		Initial Placement	
	No migr	Migr	NoMigr	Migr
No Repl	**26**	5.9	**6.1**	6.1
Repl	7.2	**6.2**	6.1	**6.1**

Unopt. **HWopt.** **SWopt.** **SW+ HW opt.**

BT

No Initial Plac.		Initial Placement	
No migr	Migr	NoMigr	Migr
960	610	**620**	600
590	**580**	580	**580**

FT

	No Initial Plac.		Initial Placement	
	No migr	Migr	NoMigr	Migr
No Repl	**520**	330	**380**	260
Repl	250	**260**	190	**200**

SP

No Initial Plac.		Initial Placement	
No migr	Migr	NoMigr	Migr
1540	780	**760**	780
670	**680**	670	**670**

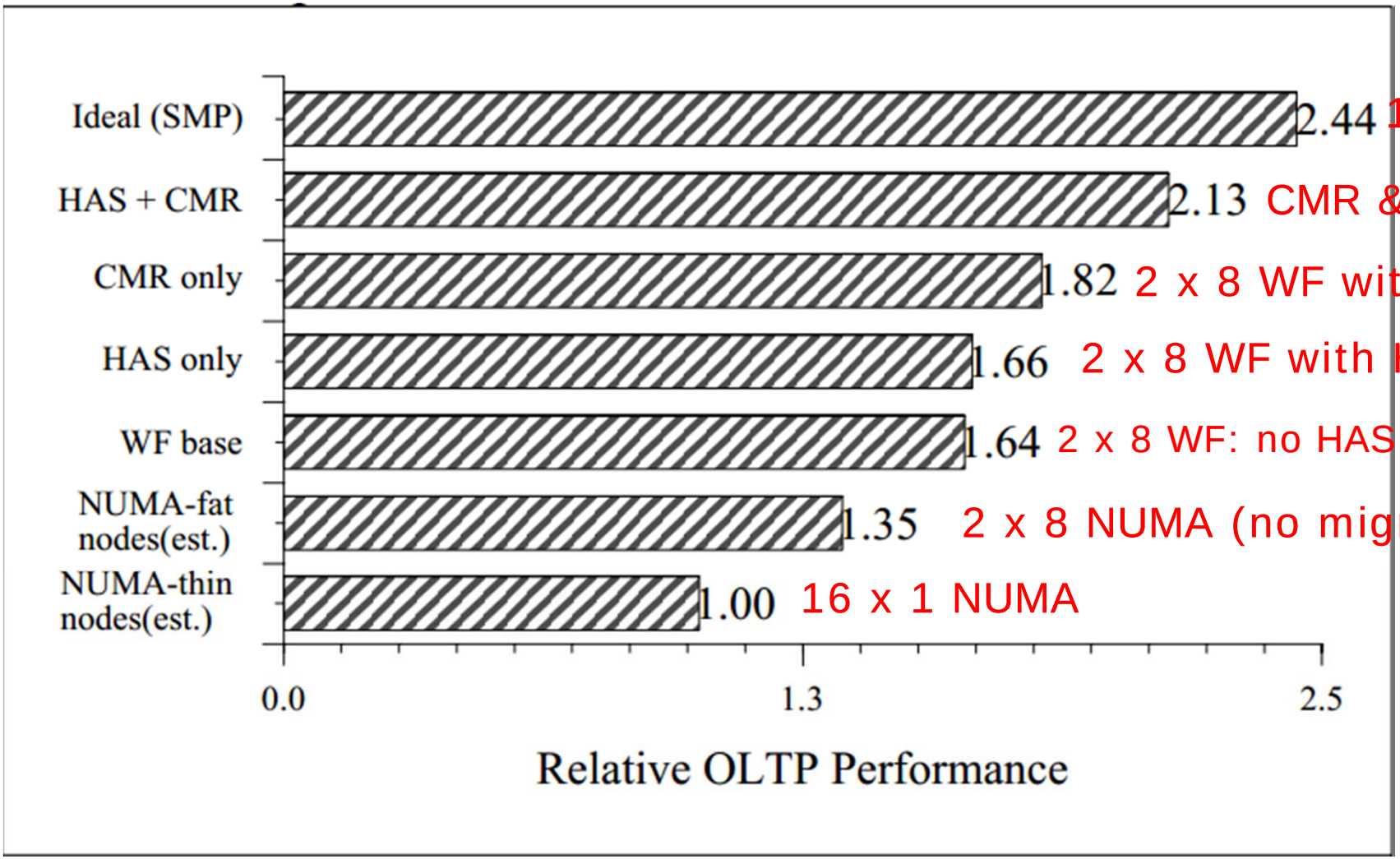
MG

	No Initial Plac.		Initial Placement	
	No migr	Migr	NoMigr	Migr
No Repl	**230**	230	**240**	230
Repl	220	**220**	220	**220**

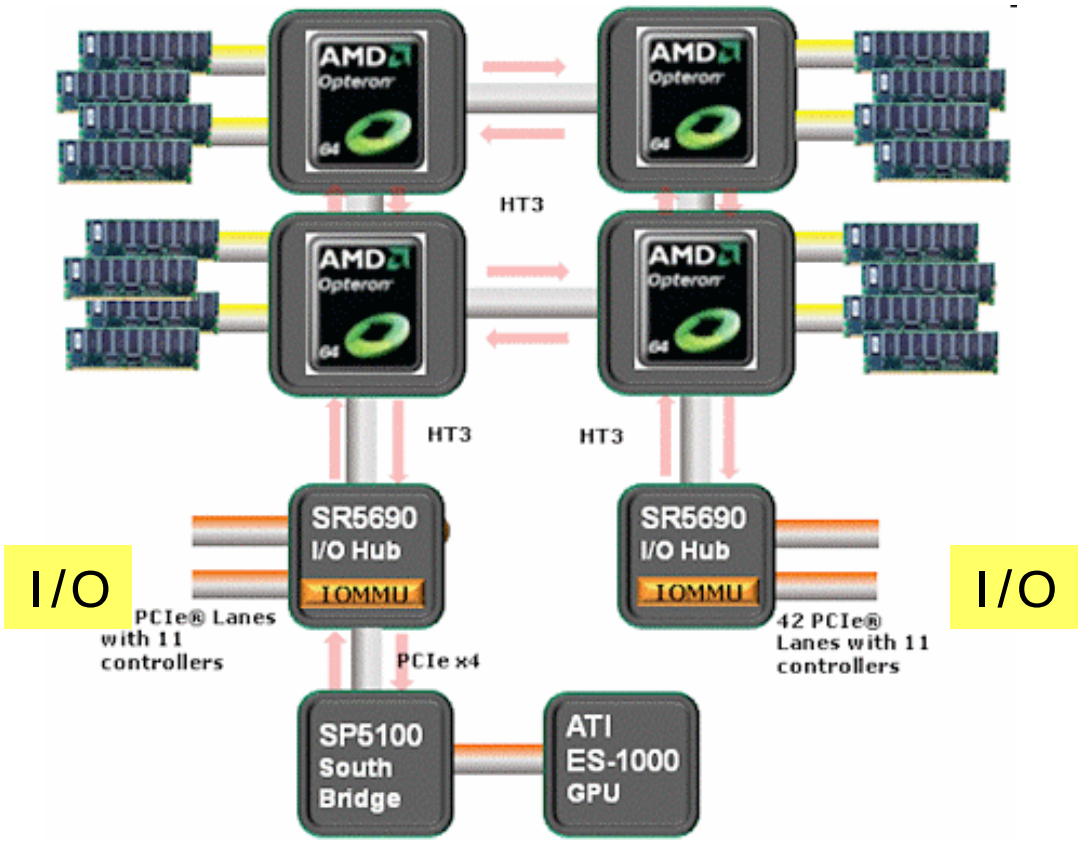
CG

No Initial Plac.		Initial Placement	
No migr	Migr	NoMigr	Migr
1060	700	**940**	700
300	**280**	300	**290**

Commercial database applications, 16 threads



x86/Linux NUMA multisocket



Typically, no support for CMR
Read-only pages may be replicated (but this is rare)
First-touch NUMA placement of data
Experiments with ”next-touch” in Linux 2009
There are large last-level caches though (LLC)

UPPSALA UNIVERSITET

What are the physical limitations of servers?

- Door frame size
- Cargo hold size
- Size and weight of CPU boards
- Chip energy (< 150W per x86 CPU chip, 300W for GPUs)
- Flow of air (e.g., front-to-back, bottom-to-top)
- How many time can you fold the air?
- Performance/energy (Google)

IBM Power 6

2009 Supercomputing

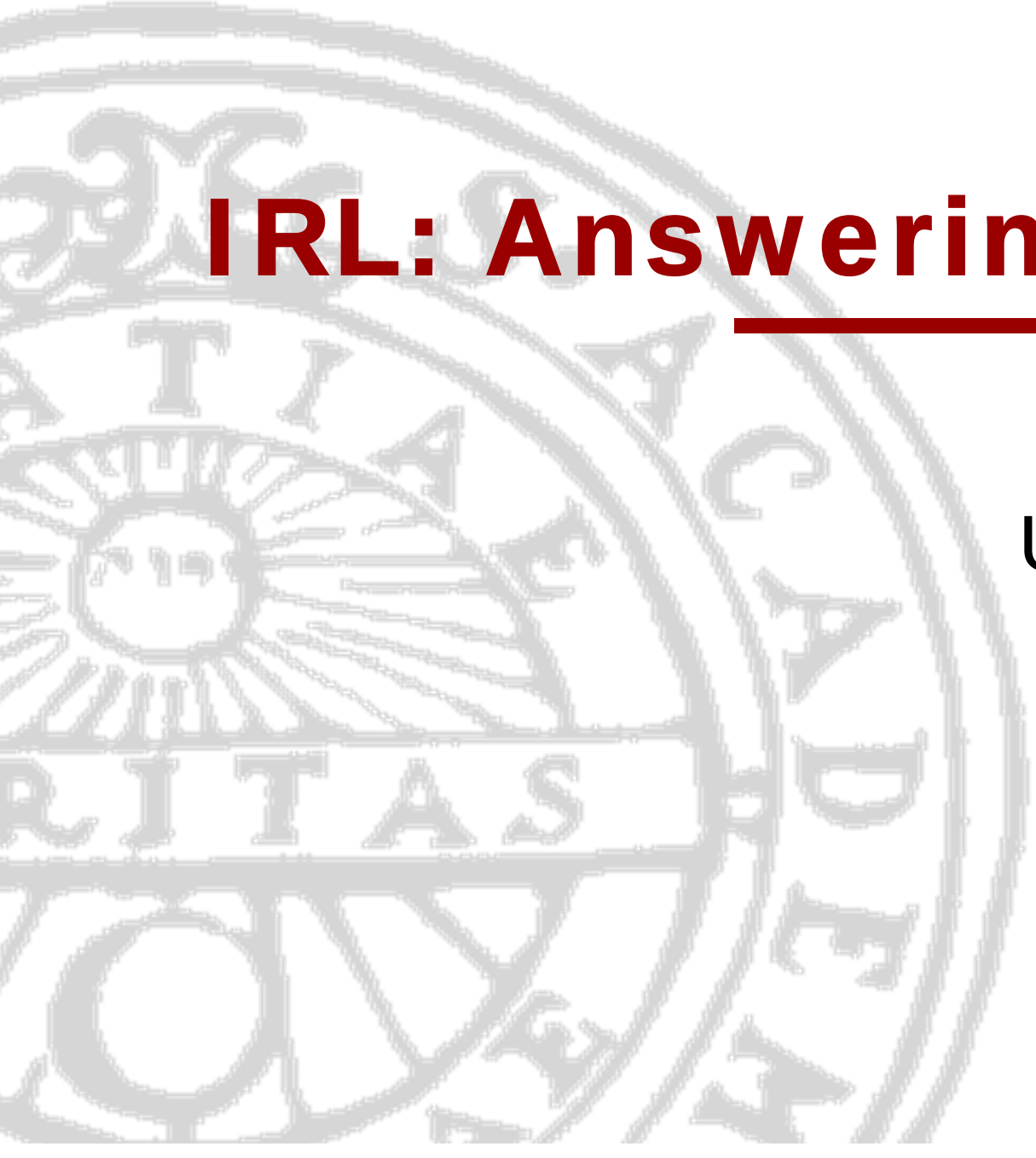
16 CPU chips (Power 6)

6 kW

120cmx100cm

Water cooling

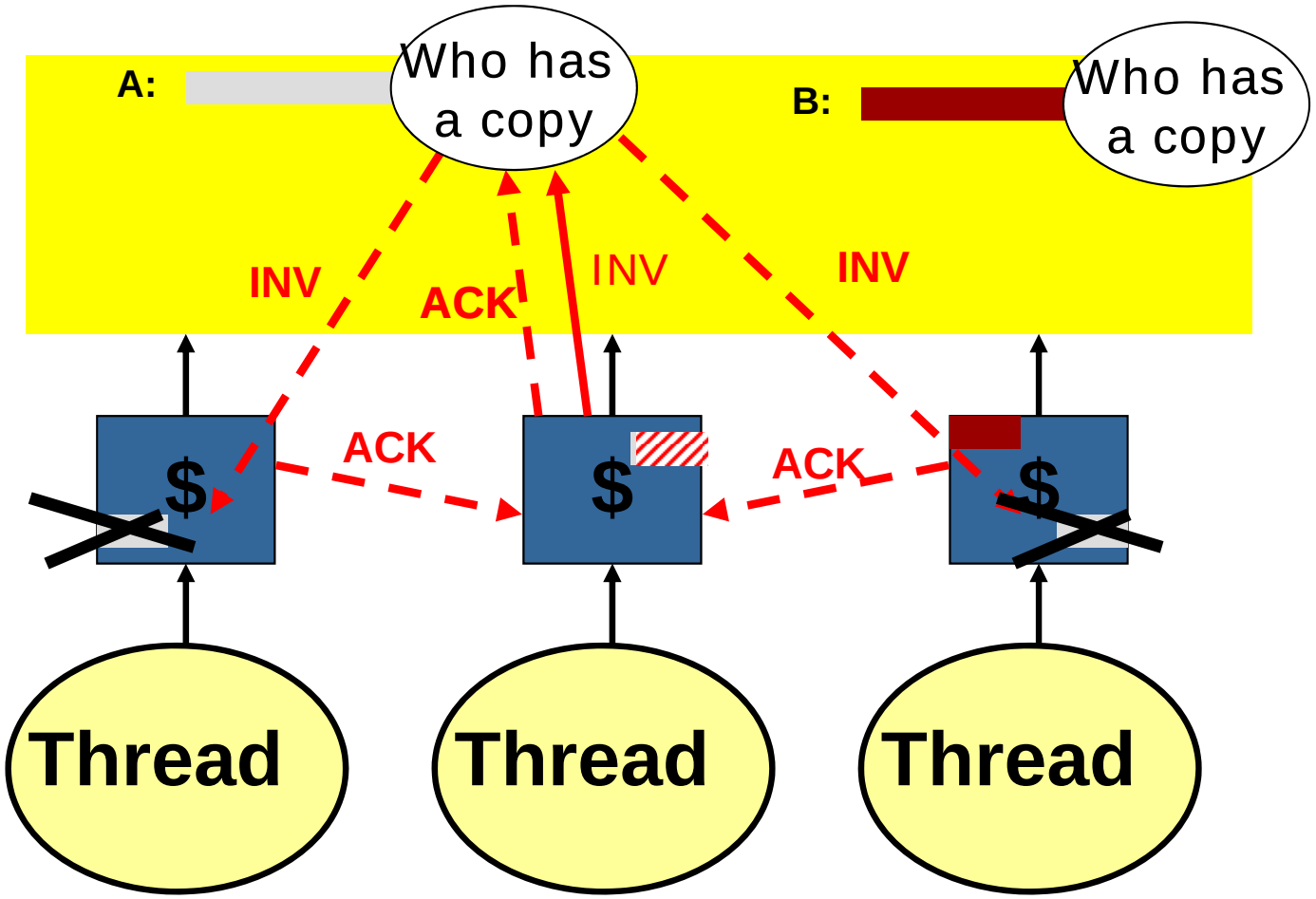
200kg



IRL: Answering questions

Erik Hagersten
Uppsala University

Q: How do you know how many ACKs to collect?

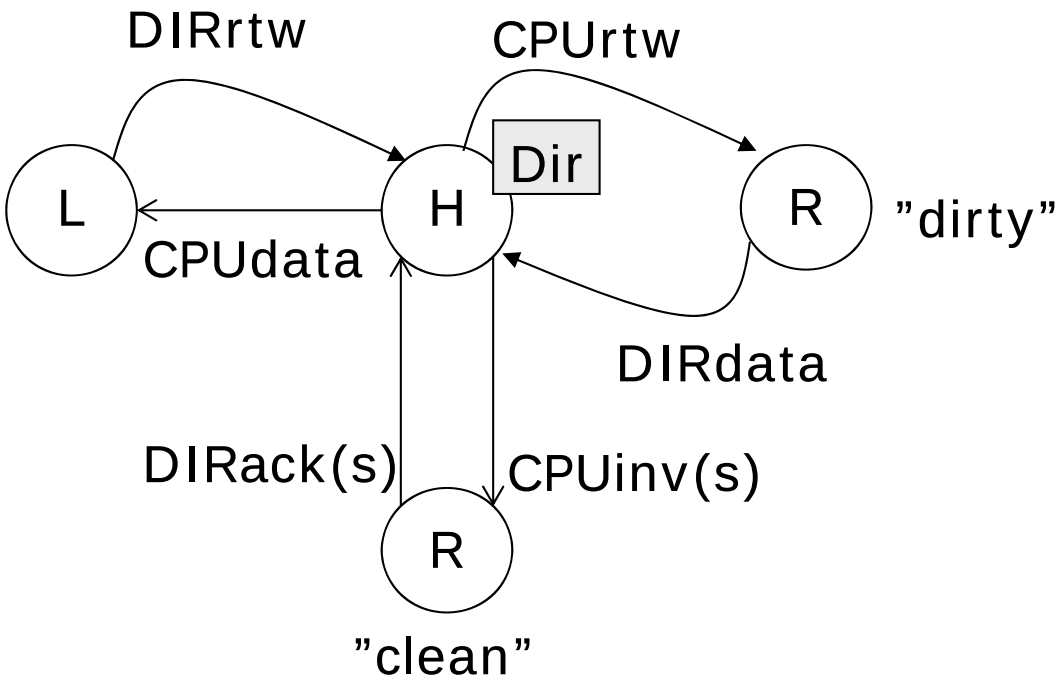


UPPSALA UNIVERSITET

As Suggested by Stanford-Dash

- **Example: RTW, dirty in a remote cache**

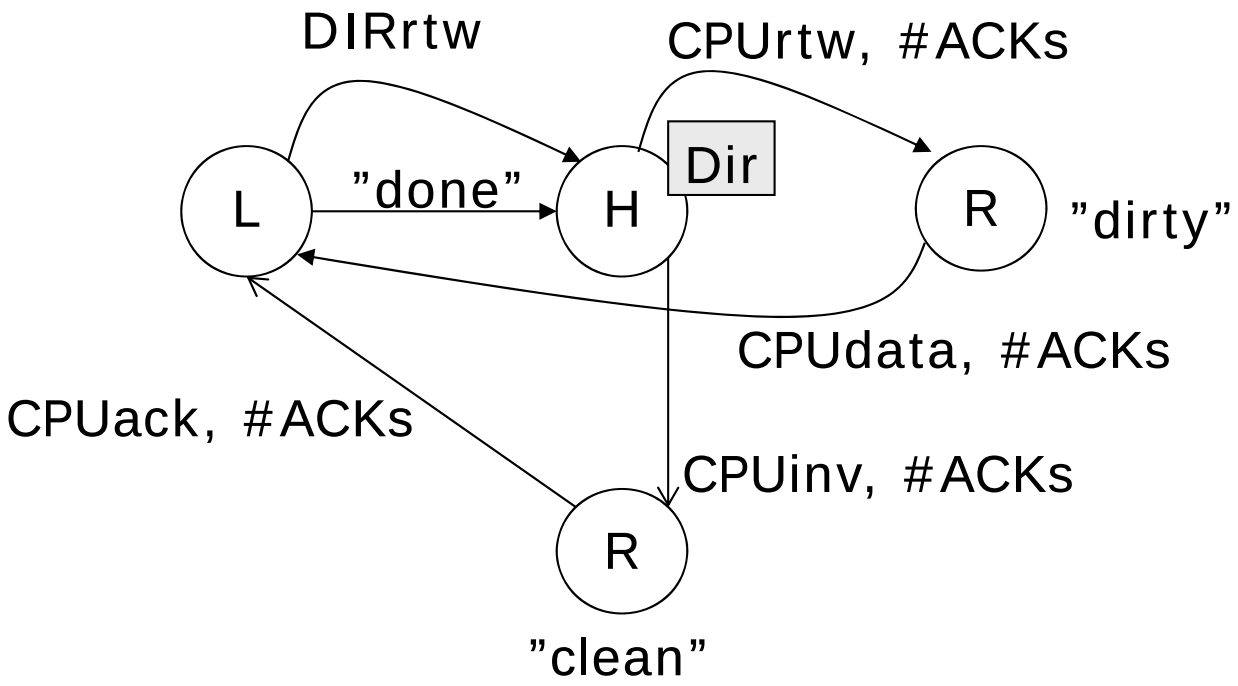
Home collects all the ACKs before returning the data



AVDARK 2013

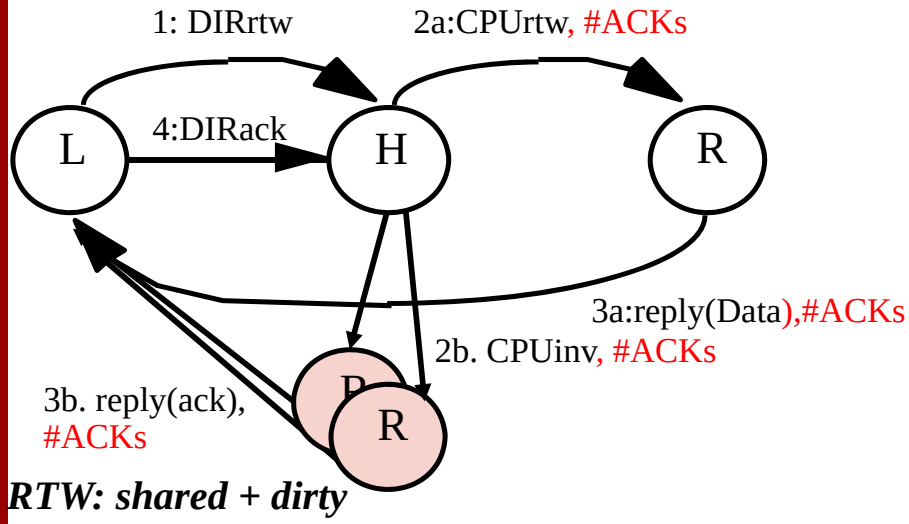
Slightly more optimized

- **Here RTW, dirty in a remote cache**



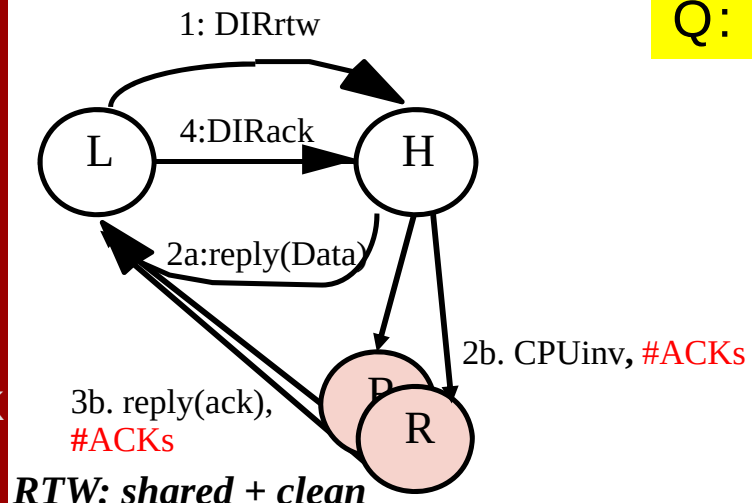
In the deterministic protocol

- L: The local ”requesting” CPU
- H: The home node with the directory
- R: A remote CPU node with a cached copy



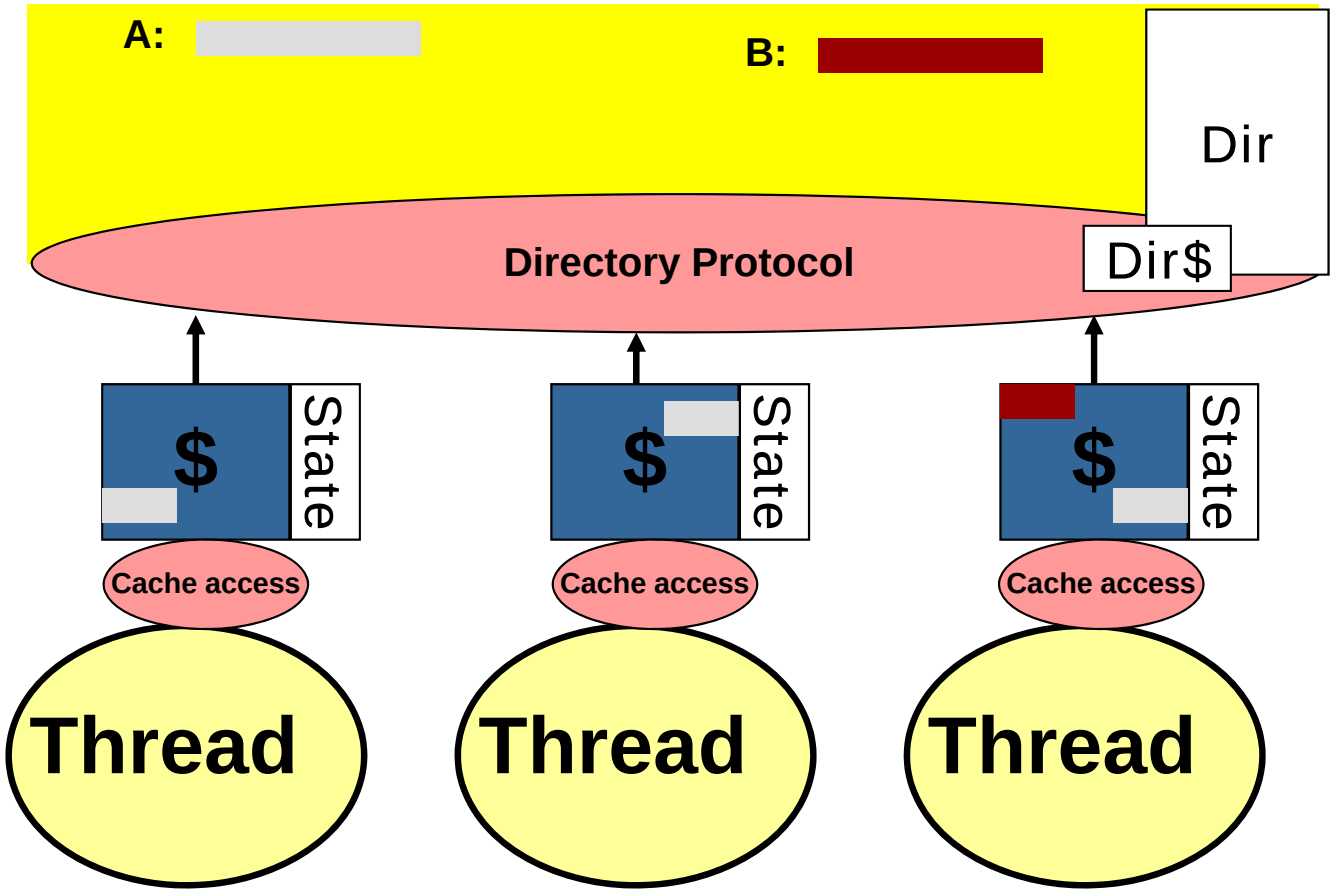
RTW: shared + dirty

Q: Is this a sequential consistency model?



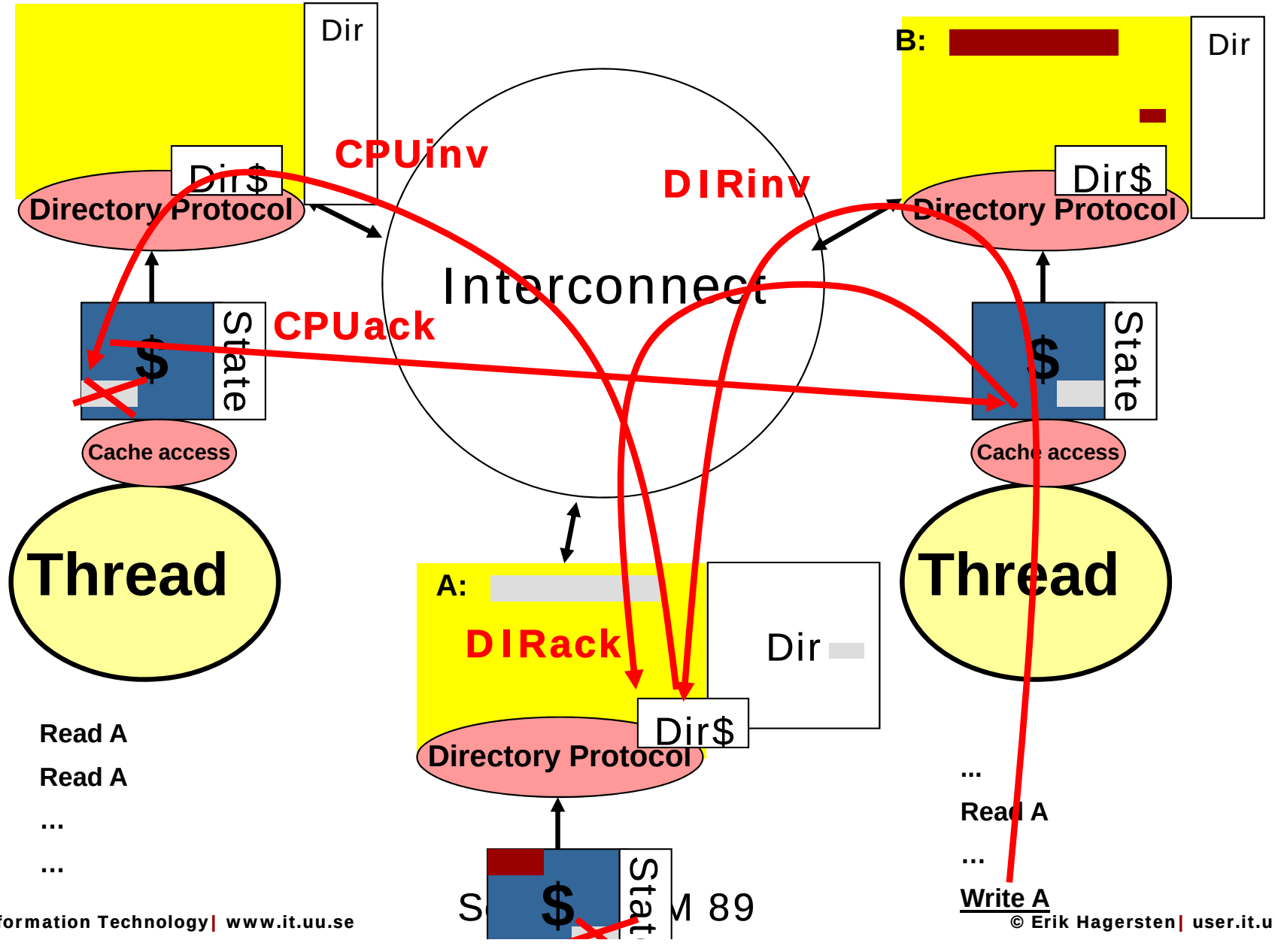
RTW: shared + clean

Q:What is a directory cache?

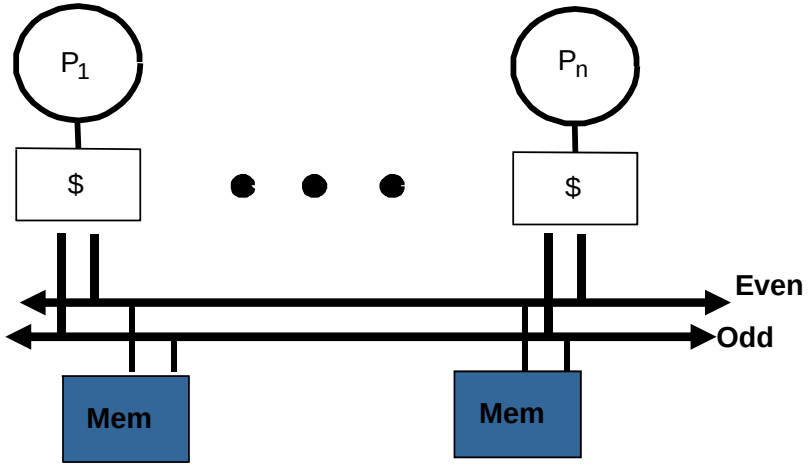


Q: Are GPUs generally NUMA structured?

Does only one node have a directory?
How do you know the middle node is the home?
What about the shared copy in the home?



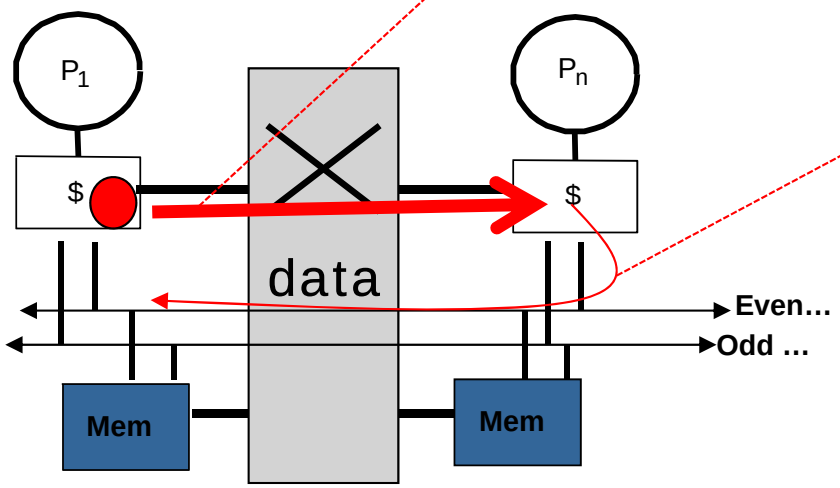
Q: What does “striped” mean?



- Multiple busses, “striped” on some address bit
- (Potentially with central logic bus implementation)
- ++ More scalable
- + Memory ordering model fix: odd < even
- - Each cache to snoop several busses
- Ex: Sun Dragon SC2000 (2 busses), CRAY CS64000 (4 busses)

How does this work?

<destination=#CPU, Uid, Data>



RTS, Addr(EVEN), ID
ID= <#CPU, uID>

- One or many address busses ...
- A separate crossbar switch for data (p2p, ordering of data replies do not matter)
- ++ Higher bandwidth
- ++ Fewer pins (better use of the wide datapaths)
- -- Longer Latency
- Ex: Sun E10000 StarFire, E15000, E6800

Discussion topics

UPPSALA UNIVERSITET

- Directory representation
  - Fully mapped
  - DIRiB
  - DIRiNB
  - DIRiCV
- Protocols
  - MSI
  - MOSI
  - MESI

AVDARK 2013

UPPSALA UNIVERSITET

What is a good implementation? (Space / efficiency / complexity)

UMA: Mem access latency = 2x latency of cache-to-cache accesses

A. 16 caches
B. 128 caches

1. Migratory sharing: (ALL: x = x + 1)
2. Producer consumer, P1: x := ..., P2: ... := x
3. One writer, 8 readers (writes are 0,2% of all mem accesses)
4. Massive read-only sharing
5. Many independent applications

	MSI	MESI	MOSI
DIRiB		B4, B5	B2, B1
DIRiNB			
DIRiCV			B3
Fully mapped		A4,A5	A3, A2, A1