ECE 669

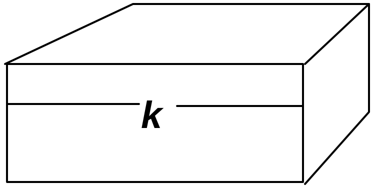
Parallel Computer Architecture

Lecture 25

Final Exam Review

Bandwidth & Latency

Direct



Latency

- **Average latency k_d:**

nk – worst case

$n\frac{(k-1)}{2}$ - *torus* - 1 dir channels

$\sim n\frac{k}{4}$ – *torus* – 2 dir channels

$\sim n\frac{k}{3}$ – no end conns – 2 dir channels

- **Bandwidth per node - more complex**
  - $\frac{1}{B}$ **if all near neighbor messages**
  - **If average travel dist then?**
  - **Let**

Avg DIST $= \frac{nk}{3}$

Msg size $= B$

Analogy

- If each student takes 8 years to graduate
- And if a Prof. can support 10 students max at any time

How many new students can Prof. take on in a year? $8x = 10$

Each year take on $\frac{10}{8}$

Similarly:

- Network has Nn channels
- # of flits it can sustain = Nn
- # of msgs it can concurrently sustain= $\frac{Nn}{B}$
- Each msg flit uses $\frac{nk}{3}$ channels to <u>dest</u>
- So

$$N \; x \; \frac{nk}{3} = \frac{Nn}{B}$$

or

$$BW \text{ per node} = x = \frac{3}{kB}$$

Another way of getting *BW* is:

- Max # msgs in net at any time $= \frac{Nn}{B}$
- These take $= \frac{kn}{3}$ cycles to get delivered, during which time no new mgs can get in
- I.e. we can inject $\frac{Nn}{B}$ msgs every $\frac{kn}{3}$ cycles

  or # injected per node per cycle

$$= \frac{Nn}{B} \cdot \frac{3}{kn} \cdot \frac{1}{N}$$

$$= \frac{3}{Bk}$$

- Note: We have not considered contention thus far.
- In practice, latency shoots up much before we achieve the theoretical due to contention.

Deriving *U* is not easy

- **Notice**

  m = **Probability of a message on a** useful **processor cycle**

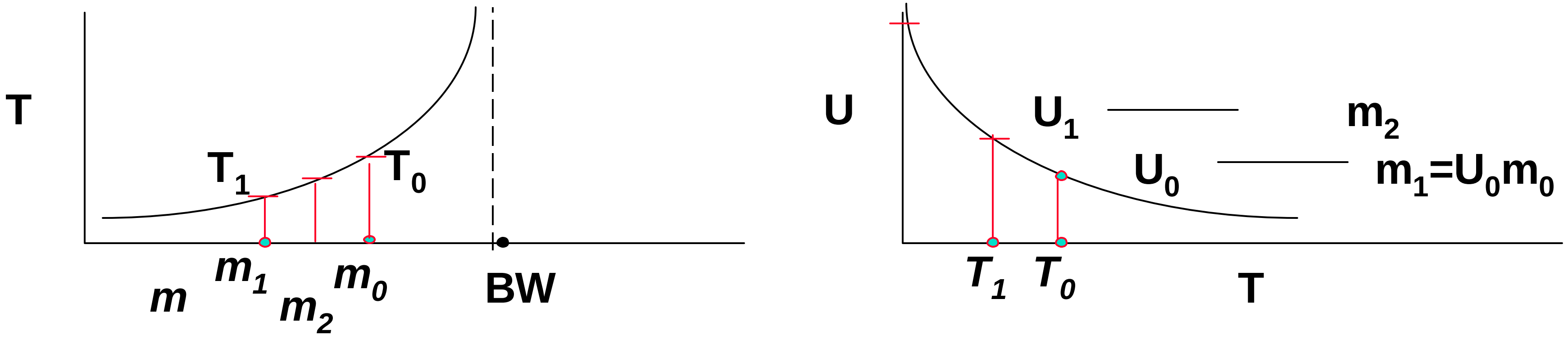
  m_{eff} = *m* • *U* = probability of msg on any cycle

  T = *f* (m_{eff}) = network delay as a function of *m*

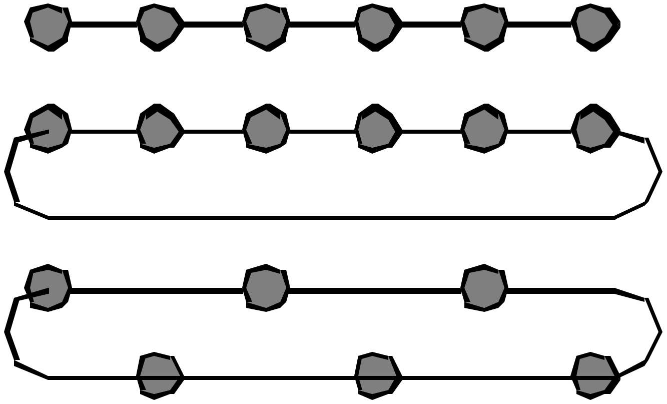
$$U = \frac{1}{1 + mT}$$

  or # of useful processor cycles depends on *T* and m_{eff}

- **Cyclic dependence!**



Linear Arrays and Rings



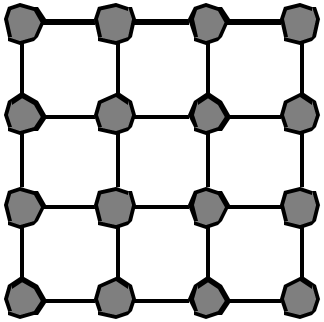
Linear Array

Torus

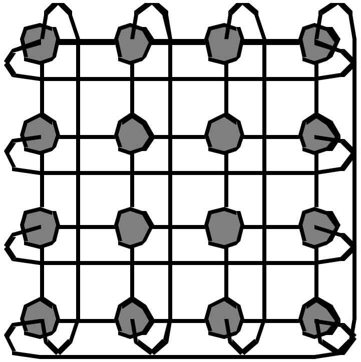
Torus arranged to use short wires

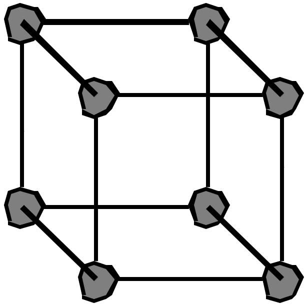
- **Linear Array**
  - **Diameter?**
  - **Average Distance?**
  - **Bisection bandwidth?**
  - **Route A -> B given by relative address R = B-A**
- **Torus?**
- **Examples: FDDI, SCI, KSR1**

Multidimensional Meshes and Tori



2D Grid

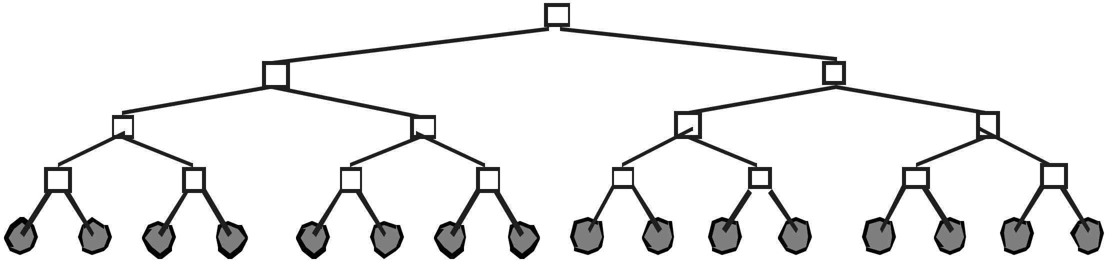


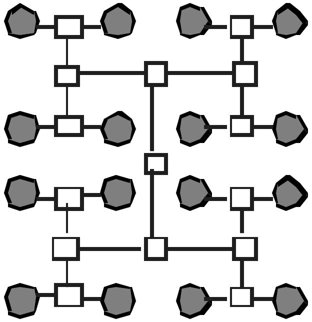


3D Cube

- ***n*-dimensional *k*-ary mesh: N = k^n**
    - **k = ^{n}ÖN**
    - **described by *n*-vector of radix k coordinate**
- ***n*-dimensional *k*-ary torus (or *k*-ary *n*-cube)?**

Trees

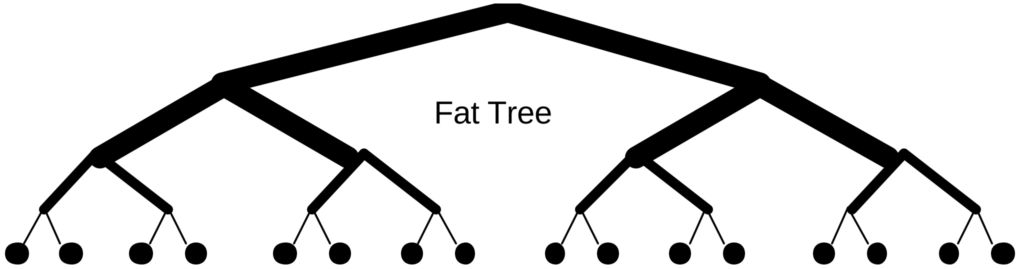


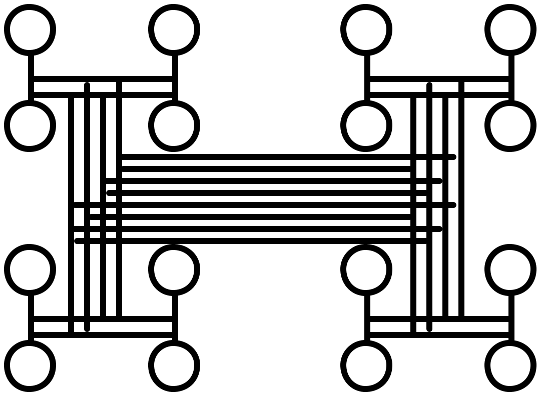


- Diameter and ave distance logarithmic
  - k-ary tree, height d = $\log_k$ N
  - address specified d-vector of radix k coordinates describing path down from root
- Fixed degree
- H-tree space is O(N) with O(ÖN) long wires
- Bisection BW?

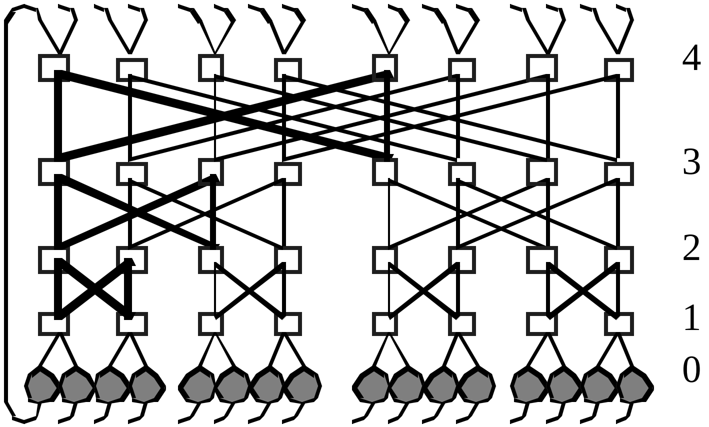
Fat-Trees

- Fatter links (really more of them) as you go up, so bisection BW scales with N

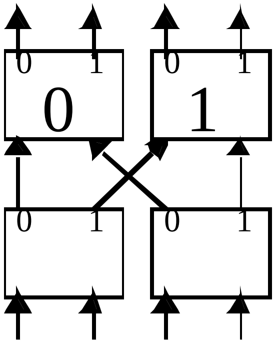




Butterflies



16 node butterfly



building block

- Tree with lots of roots!
- N log N   (actually N/2 x logN)
- Exactly one route from any source to any dest
- Bisection N/2

Hypercubes

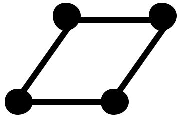
- Also called binary n-cubes.   # of nodes = N = 2^n.
- O(logN) Hops
- Good bisection BW
- Complexity
  - Out degree is n = logN

correct dimensions in order

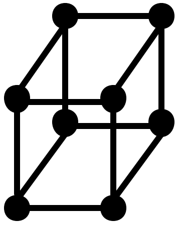
- with random comm. 2 ports per processor

0-D

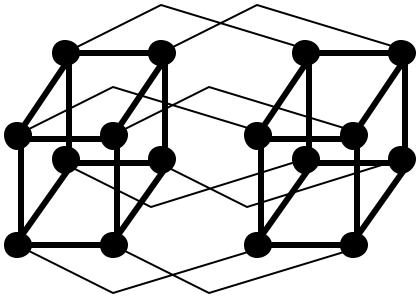
1-D



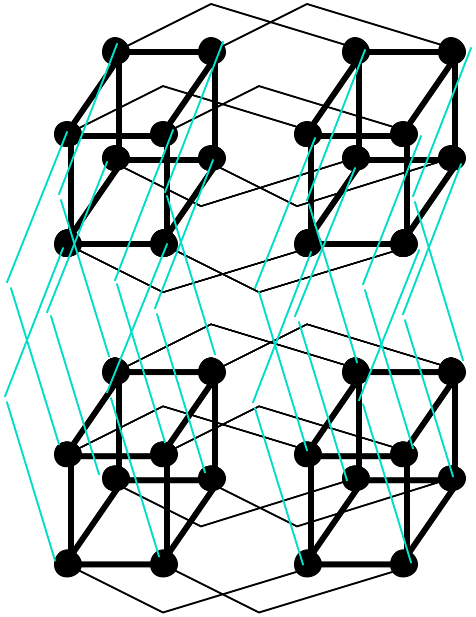
2-D



3-D



4-D



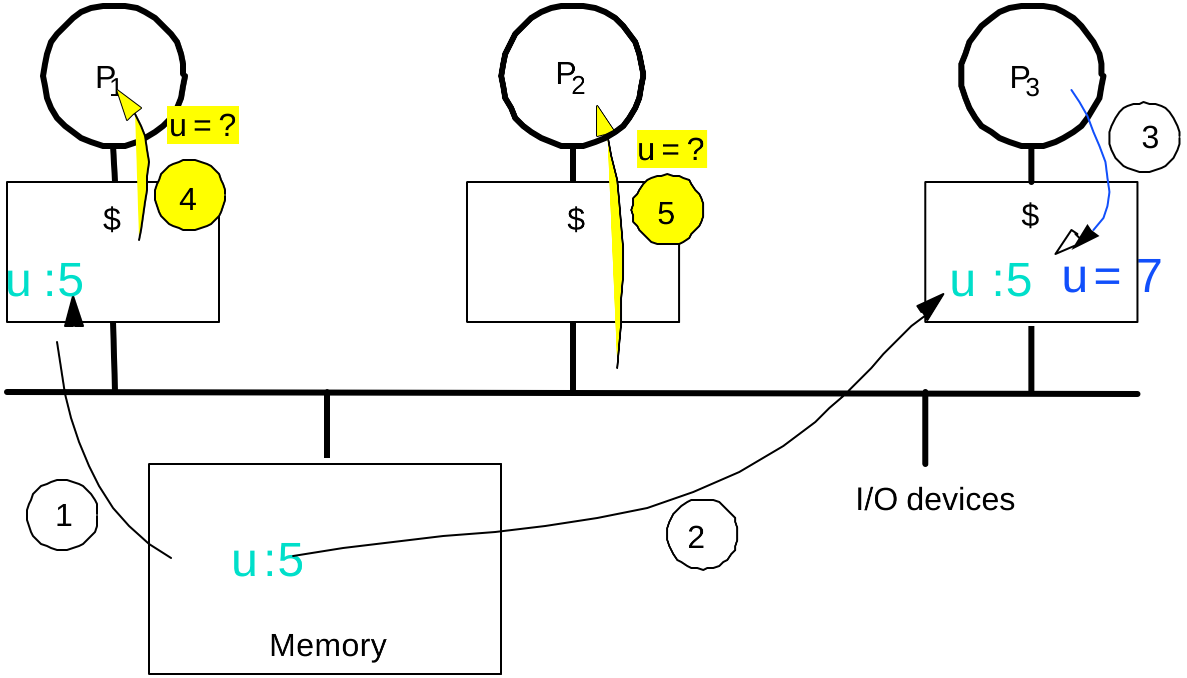
5-D !

Toplology Summary

Topology	Degree	Diameter	Ave Dist	Bisection	D (D ave) @ P=1024
1D Array	2	N-1	N / 3	1	huge
1D Ring	2	N/2	N/4	2	
2D Mesh	4	2 ($N^{1/2}$ - 1)	2/3 $N^{1/2}$	$N^{1/2}$	63 (21)
2D Torus	4	$N^{1/2}$	1/2 $N^{1/2}$	$2N^{1/2}$	32 (16)
k-ary n-cube	2n	nk/2	nk/4	nk/4	15 (7.5) @n=3
Hypercube	n =log N		n	n/2	N/2 10 (5)

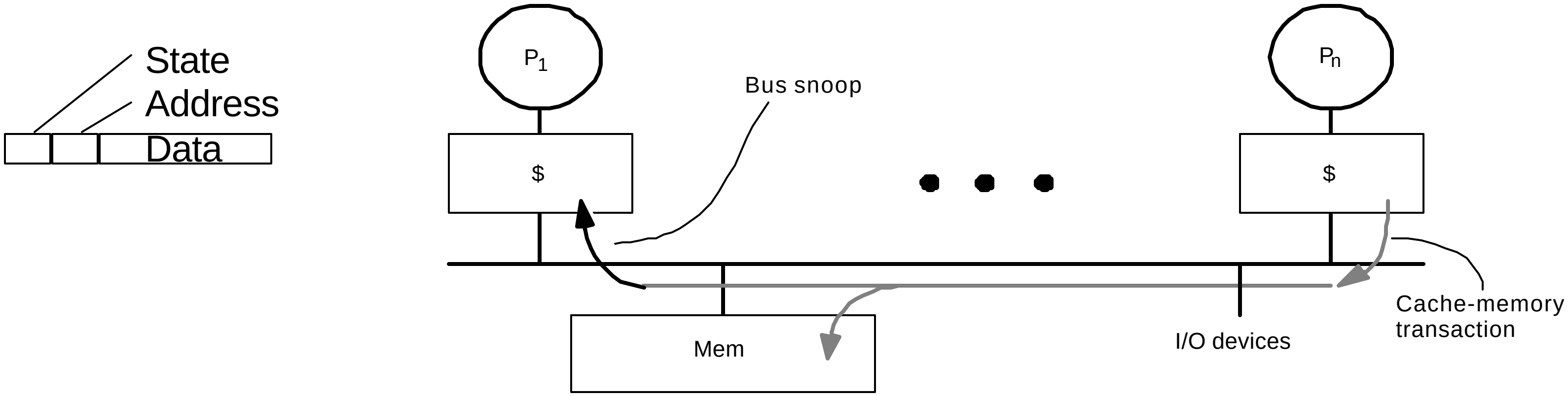
- All have some “bad permutations”
  - many popular permutations are very bad for meshs (transpose)
  - ramdomness in wiring or routing makes it hard to find a bad one!

Example Cache Coherence Problem



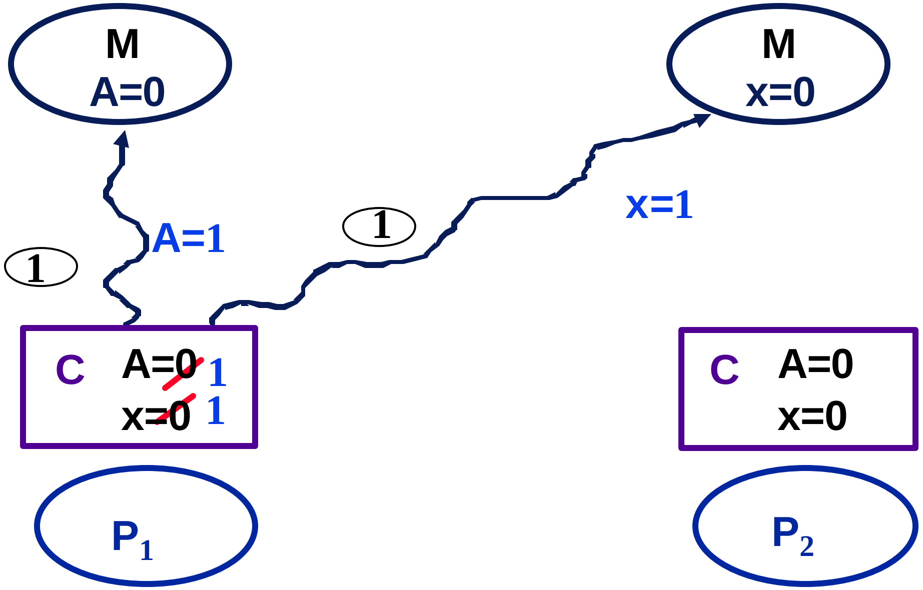
- **Processors see different values for u after event 3**
- **With write back caches, value written back to memory depends on happenstance of which cache flushes or writes back value when**
    - **Processes accessing main memory may see very stale value**
- **Unacceptable to programs, and frequent!**

Snoopy Cache-Coherence Protocols



- Bus is a broadcast medium & Caches know what they have
- Cache Controller "snoops" all transactions on the shared bus
  - relevant transaction if for a block it contains
  - take action to ensure coherence
    - invalidate, update, or supply value
  - depends on state of the block and the protocol

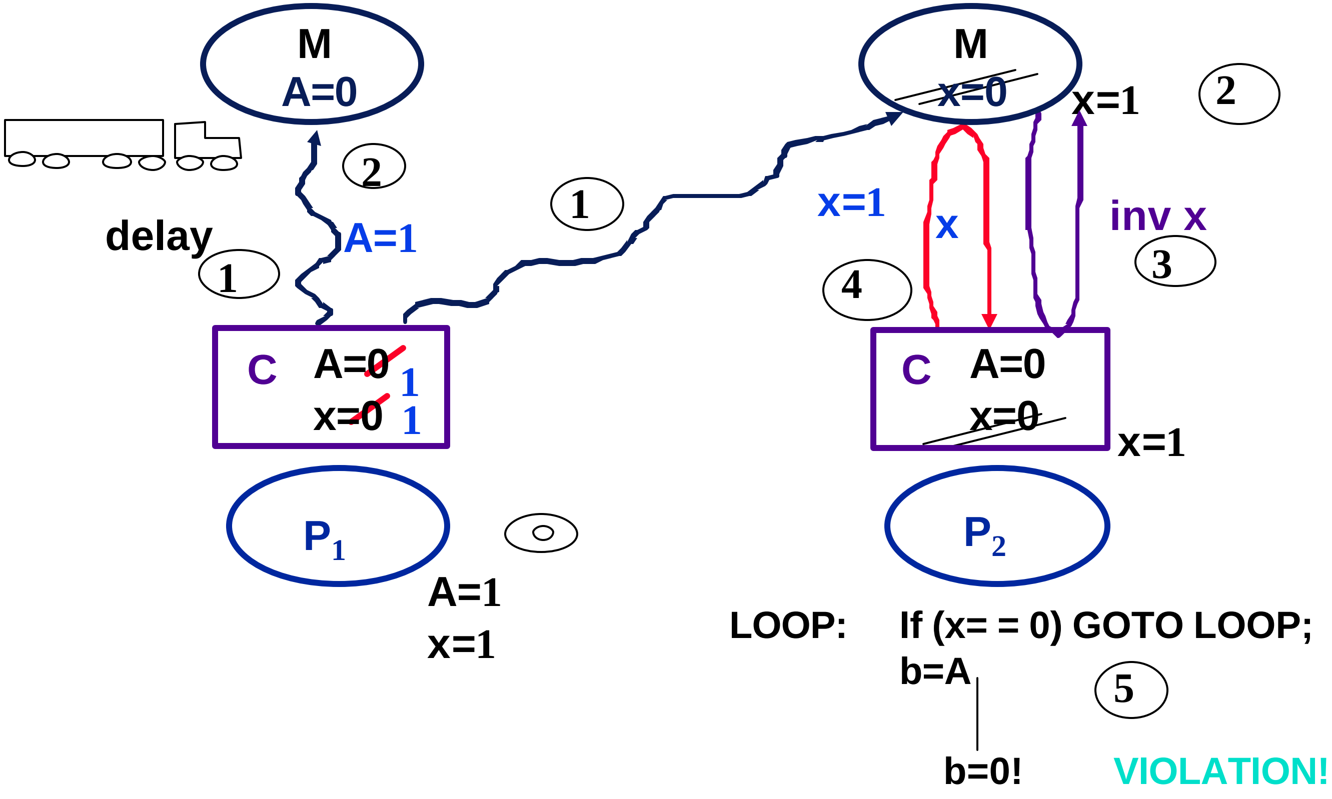
Does caching violate this model?



LOOP: If (x= = 0) GOTO LOOP;
b=A

If b = = 0 at the end, sequential consistency is violated

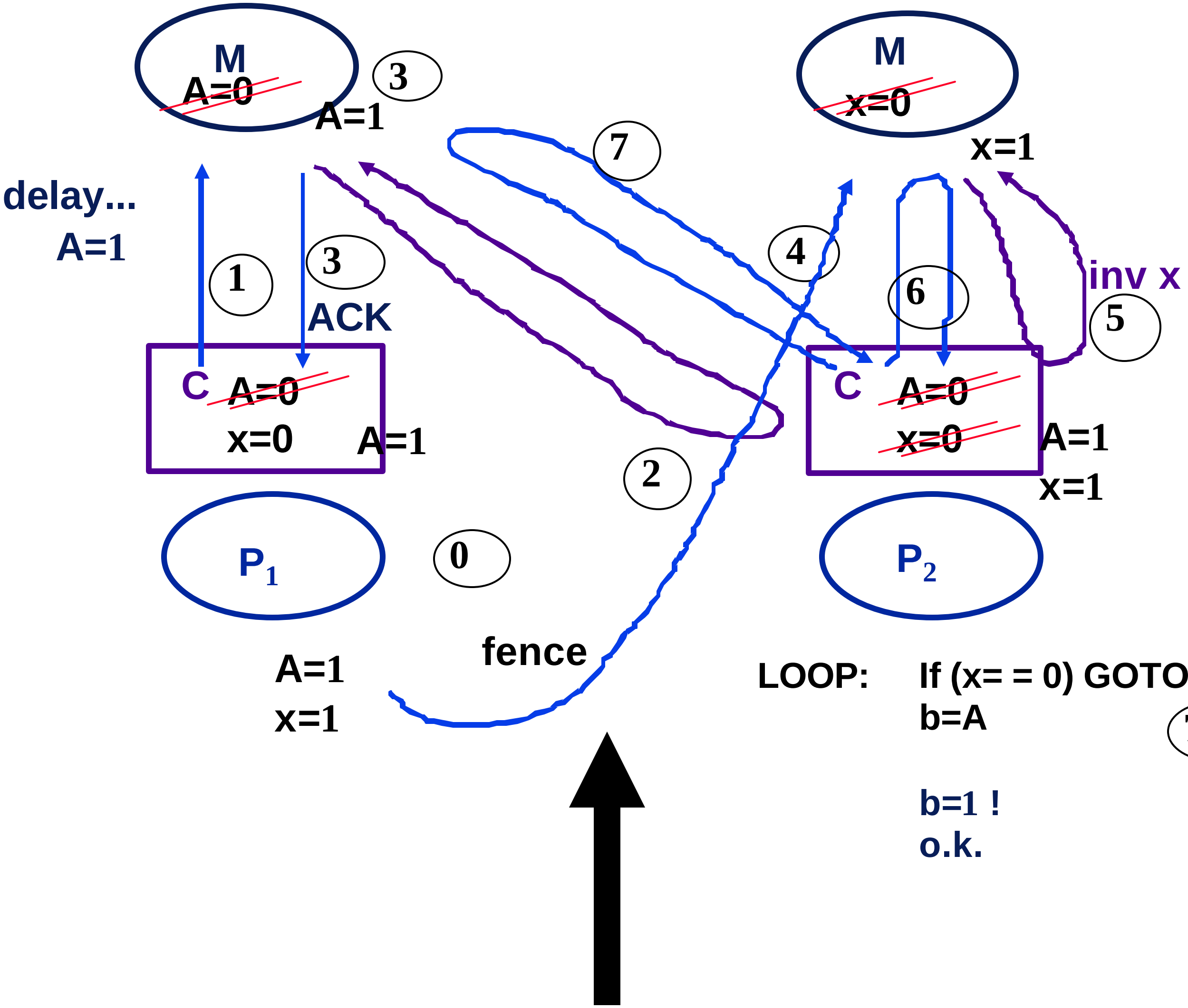
Does caching violate this model?



```
LOOP:    If (x= = 0) GOTO LOOP;
         b=A
```

If b = = 0 at the end, sequential consistency is violated

Does caching violate this model?



M

~~A=0~~

A=1

3

delay...

A=1

1

3

ACK

C ~~A=0~~

x=0

A=1

P1

0

A=1

x=1

fence

2

7

4

M

~~x=0~~

x=1

6

inv x

5

C ~~A=0~~

~~x=0~~

A=1

x=1

P2

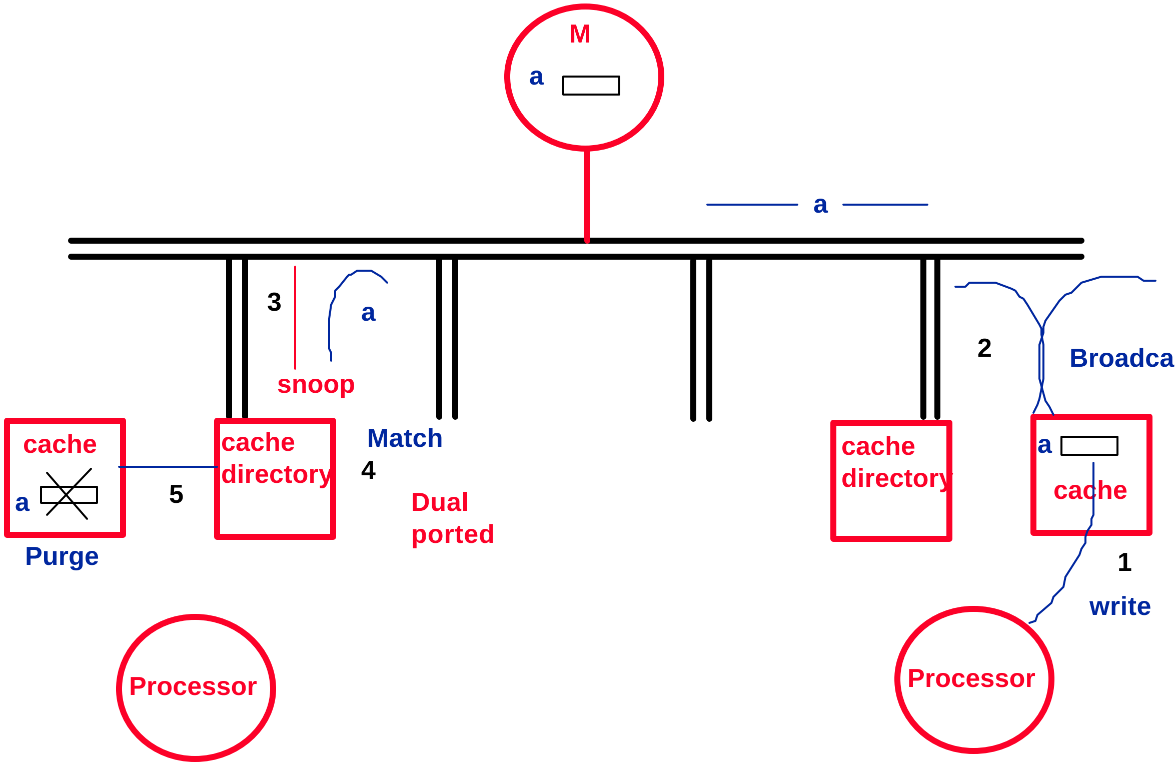
```
LOOP:   If (x= = 0) GOTO LOOP;
        b=A
```

7

b=1 !

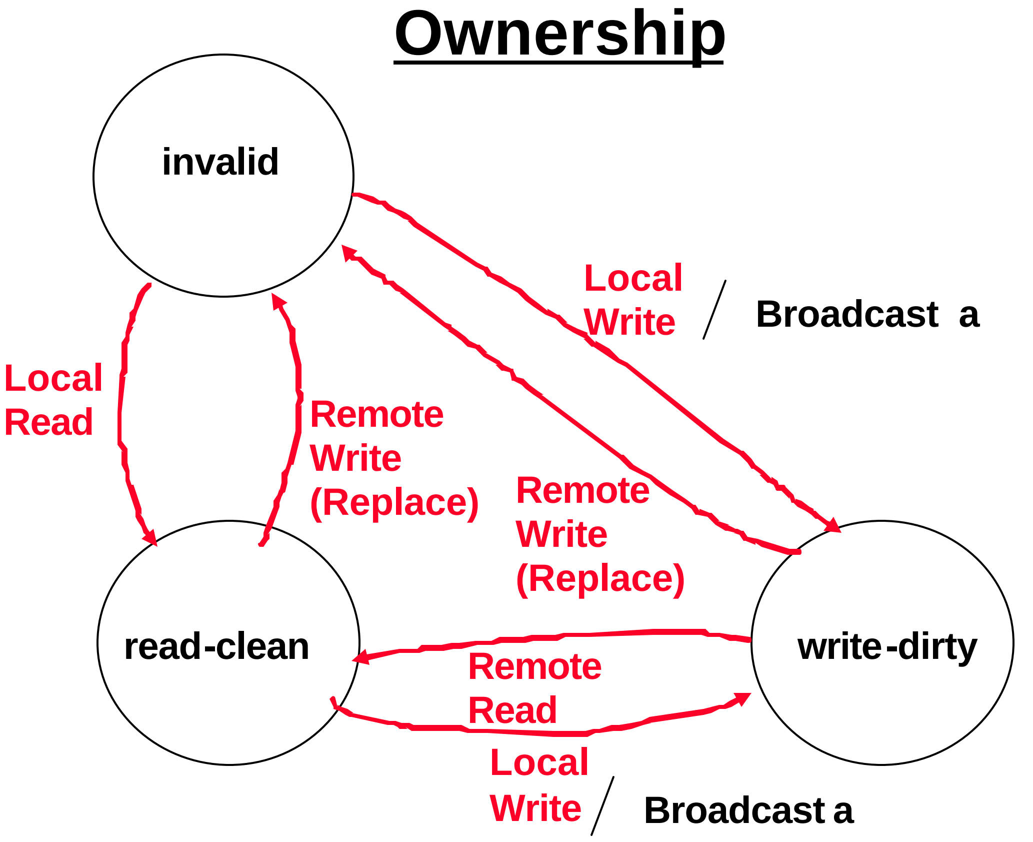
o.k.

Coherence in small machines: Snooping Caches

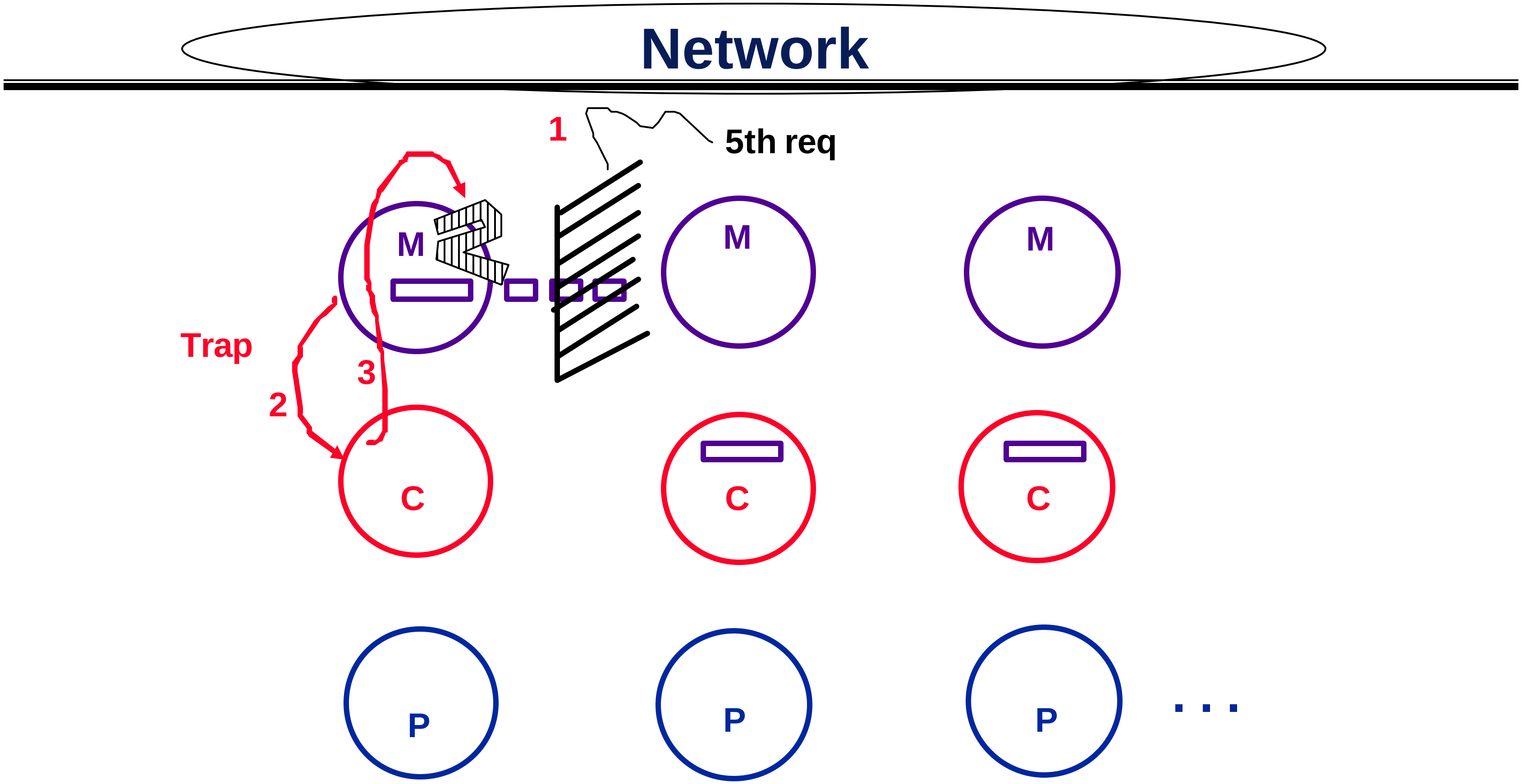


- Broadcast address on shared write
- Everyone listens (snoops) on bus to see if any of their own addresses match
- How do you know when to broadcast, invalidate...
    - State associated with each cache line

State diagram for ownership protocols



- **In ownership protocol: writer owns exclusive copy**
- For each shared data cache block

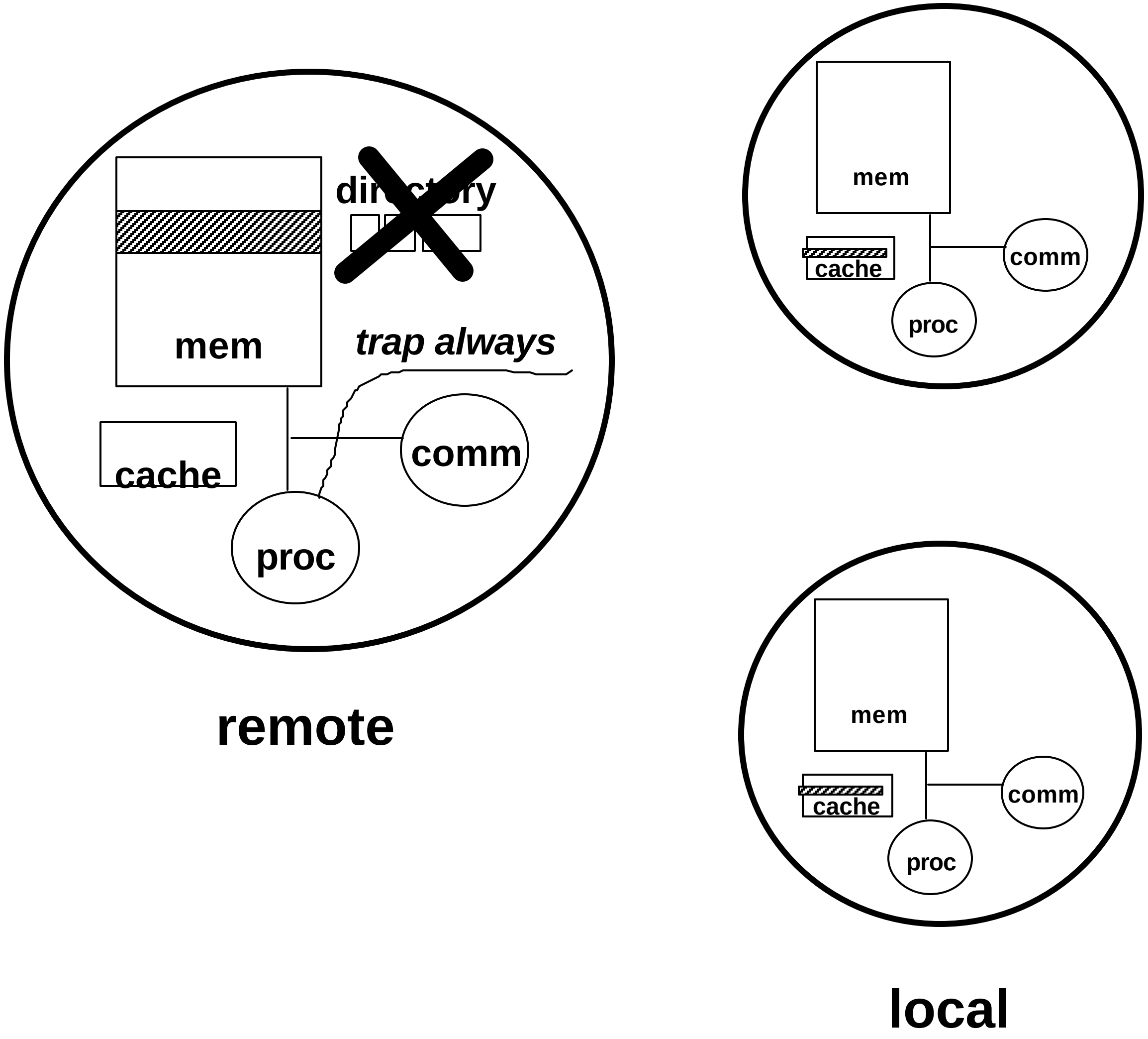


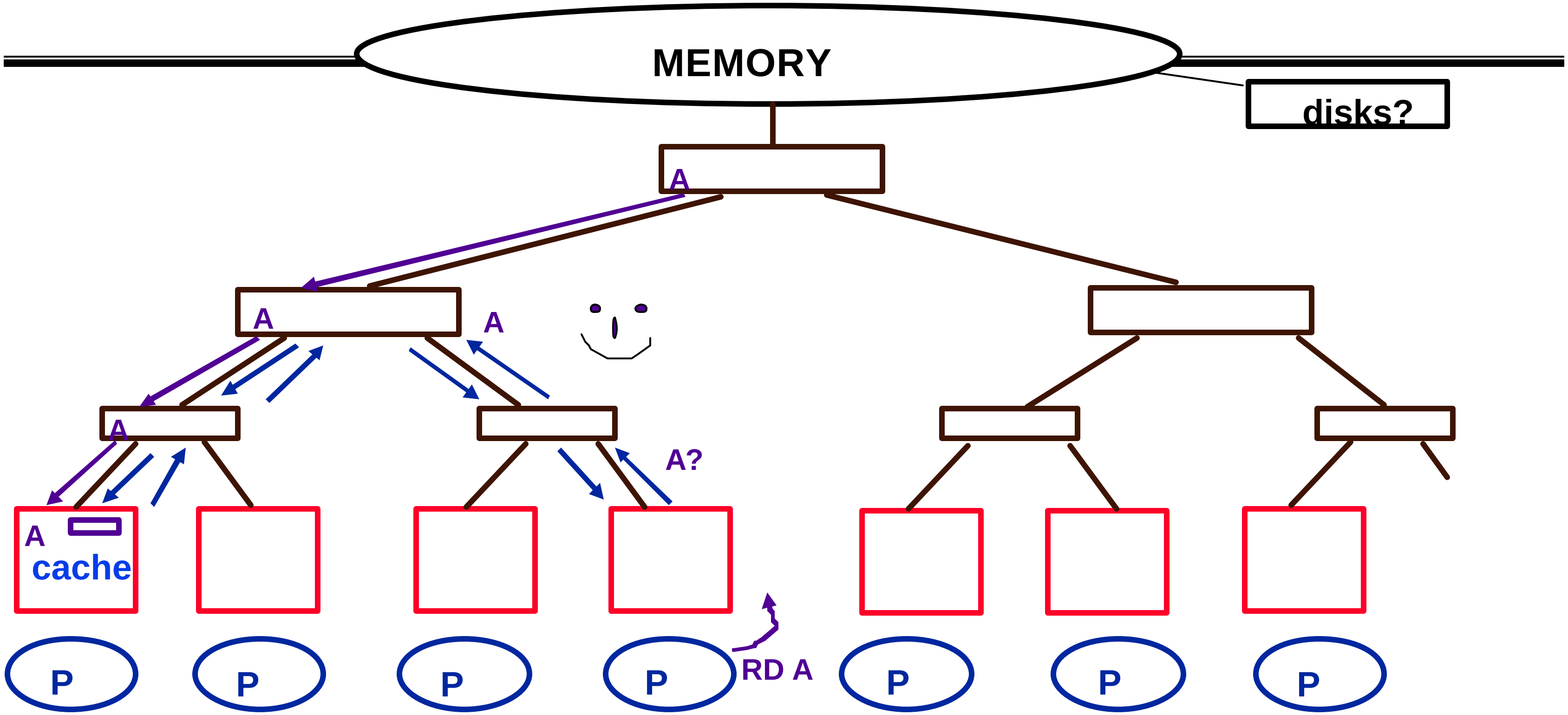
- **LimitLESS directories:**

  Limited directories Locally Extended through Software Support

  - **Trap processor when 5th request comes**
  - **Processor extends directory into local memory**

Zero pointer LimitLESS: All software coherence

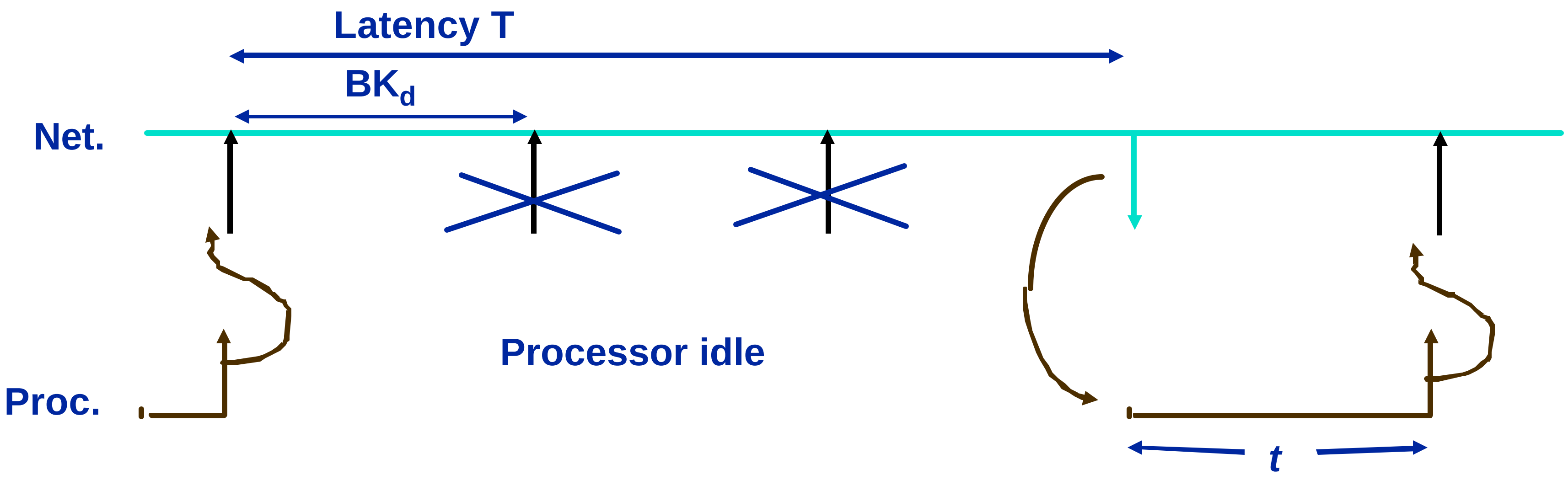




Hierarchical - E.g. KSR (actually has rings...)

Include network latency

- Each request suffers T cycles of latency



$$\frac{t}{t+T} = \frac{1}{1+mT}$$

$$\left(t = \frac{1}{m}\right)$$

- Processor utilization =

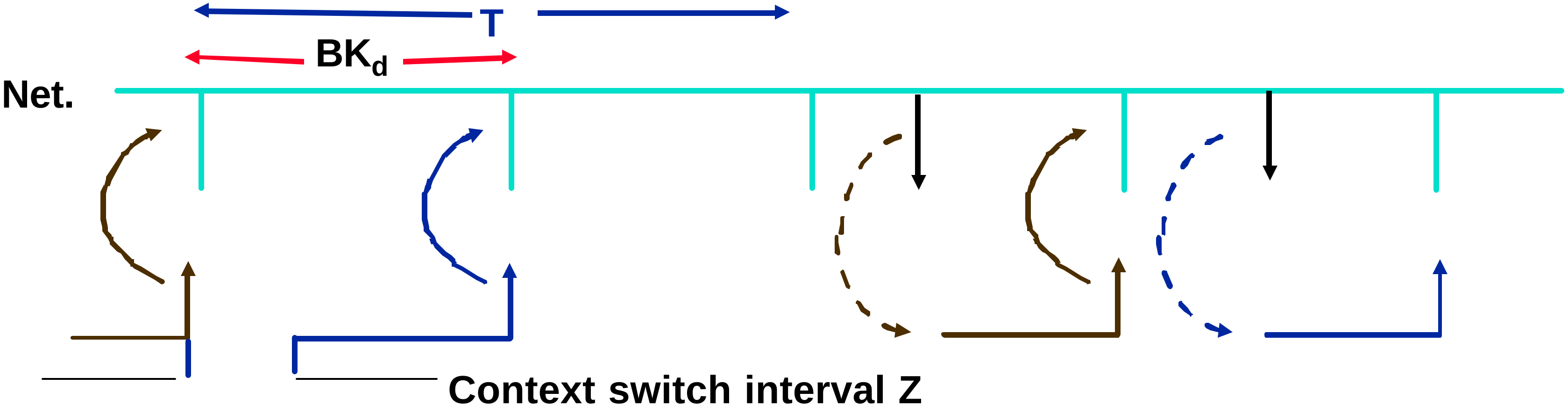
  Processor utilization |

  Network bandwidth also wasted because of lost issue opportunities!

  FIX?

One solution

Overlap communication with computation.



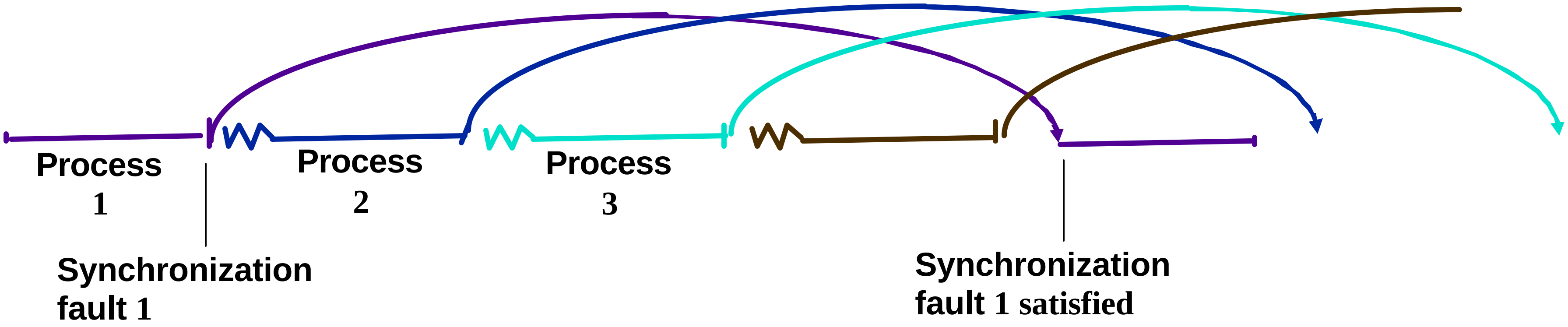
- Multithread" the processor

$$\text{Processor utilization} = \frac{pt}{t+T} \quad \text{if } pt < (t+T)$$

$$\text{or} = \frac{t}{t+Z} \quad \text{otherwise}$$

- And/or allow multiple outstanding requests -- non-blocking memory

Synchronization delays



Process 1

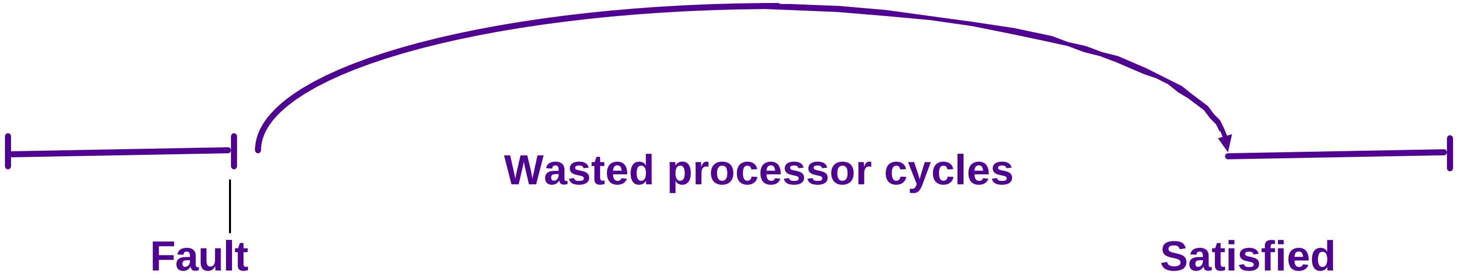
Process 2

Process 3

Synchronization fault 1

Synchronization fault 1 satisfied

- If no multithreading



Wasted processor cycles

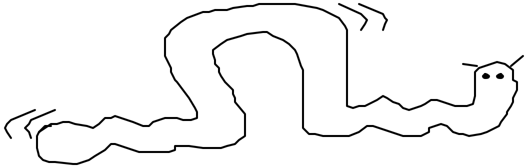
Fault

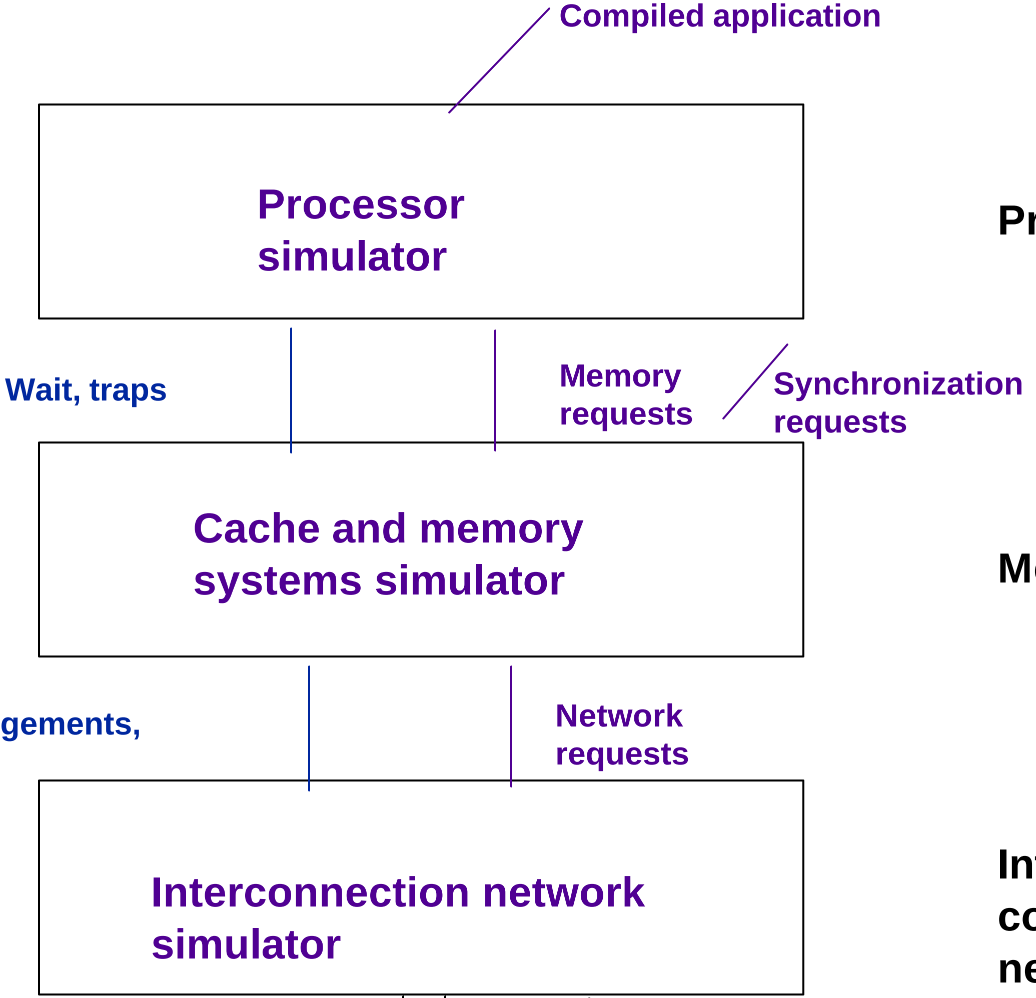
Satisfied

Full system simulation (coupled)

4000x slowdown per node

Very accurate



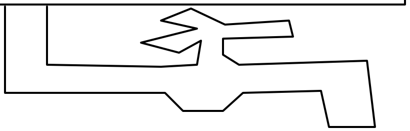


Acknowledgements, responses

Processor

Memory

Inter-connection network



Measures:

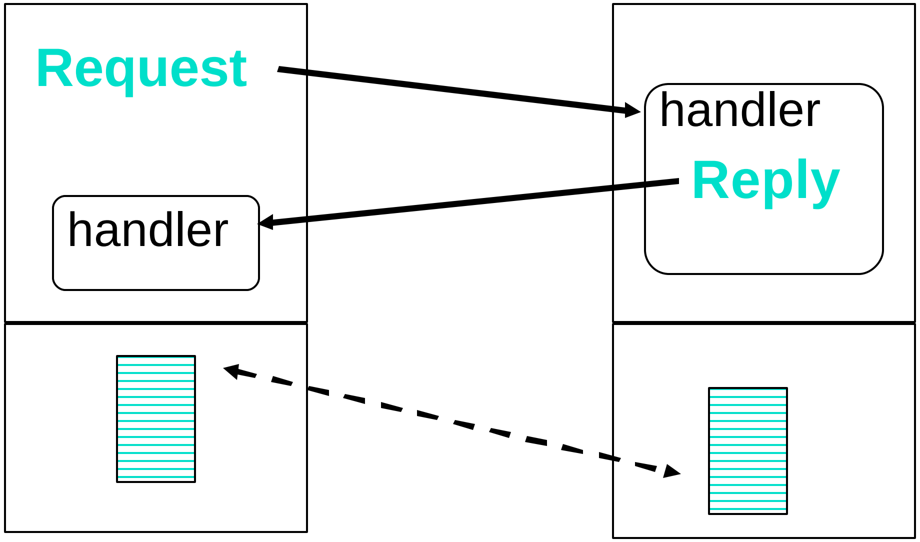
- Speedup, runtime
- proc. util.
- net latency
- cache miss rate

.

.

.

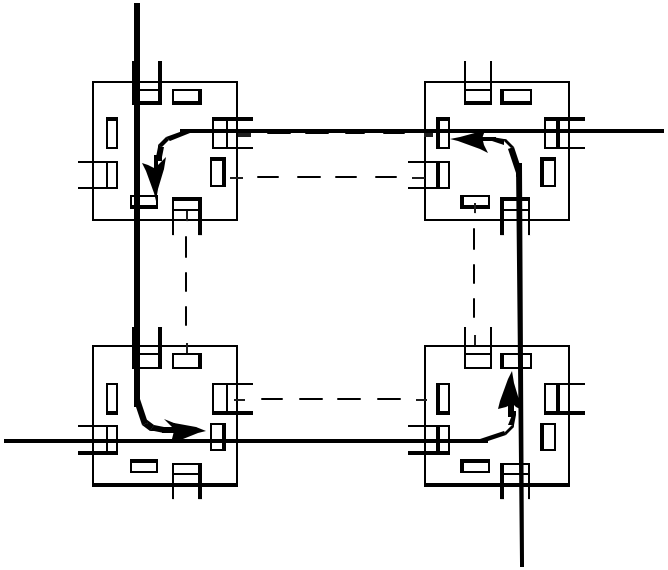
Active Messages



- User-level analog of network transaction
  - transfer data packet and invoke handler to extract it from the network and integrate with on-going computation
- Request/Reply
- Event notification: interrupts, polling, events?
- May also perform memory-to-memory transfer

Deadlock Freedom

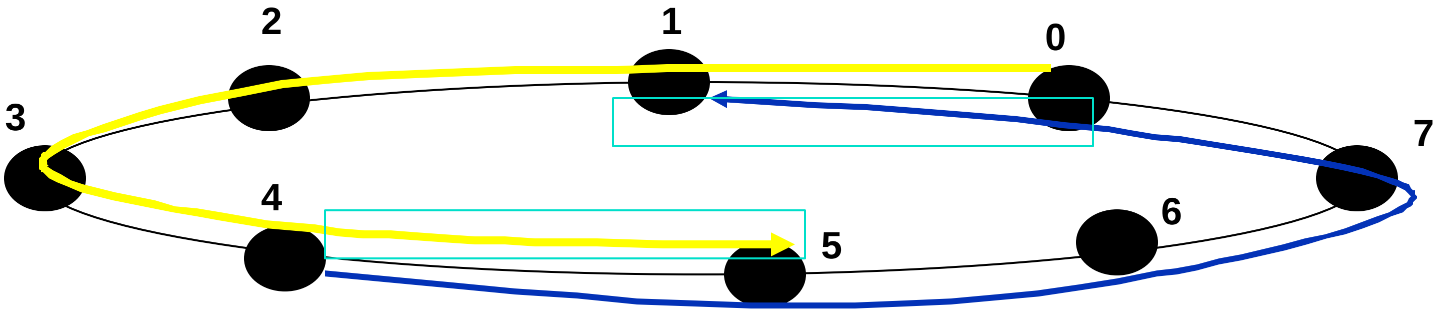
- **How can it arise?**
  - necessary conditions:
    - shared resource
    - incrementally allocated
    - non-preemptible
  - think of a channel as a shared resource that is acquired incrementally
    - source buffer then dest. buffer
    - channels along a route



- **How do you avoid it?**
  - constrain how channel resources are allocated
  - ex: dimension order
- **How do you prove that a routing algorithm is deadlock free**

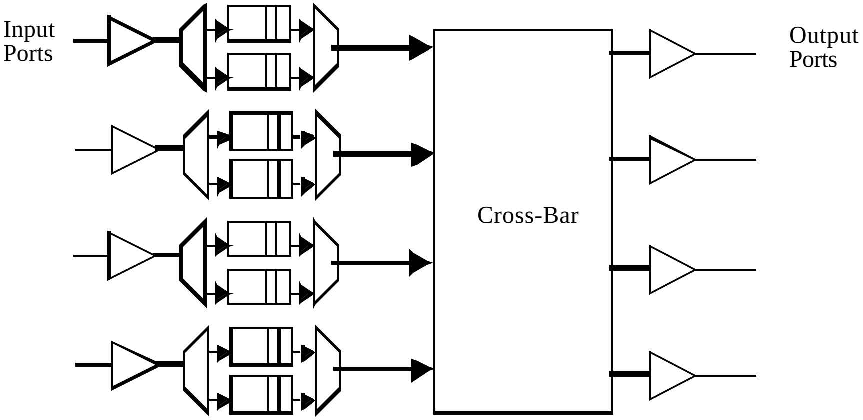
More examples:

- Consider other topologies
  - butterfly?
  - tree?
  - fat tree?
- Any assumptions about routing mechanism? amount of buffering?
- What about wormhole routing on a ring?



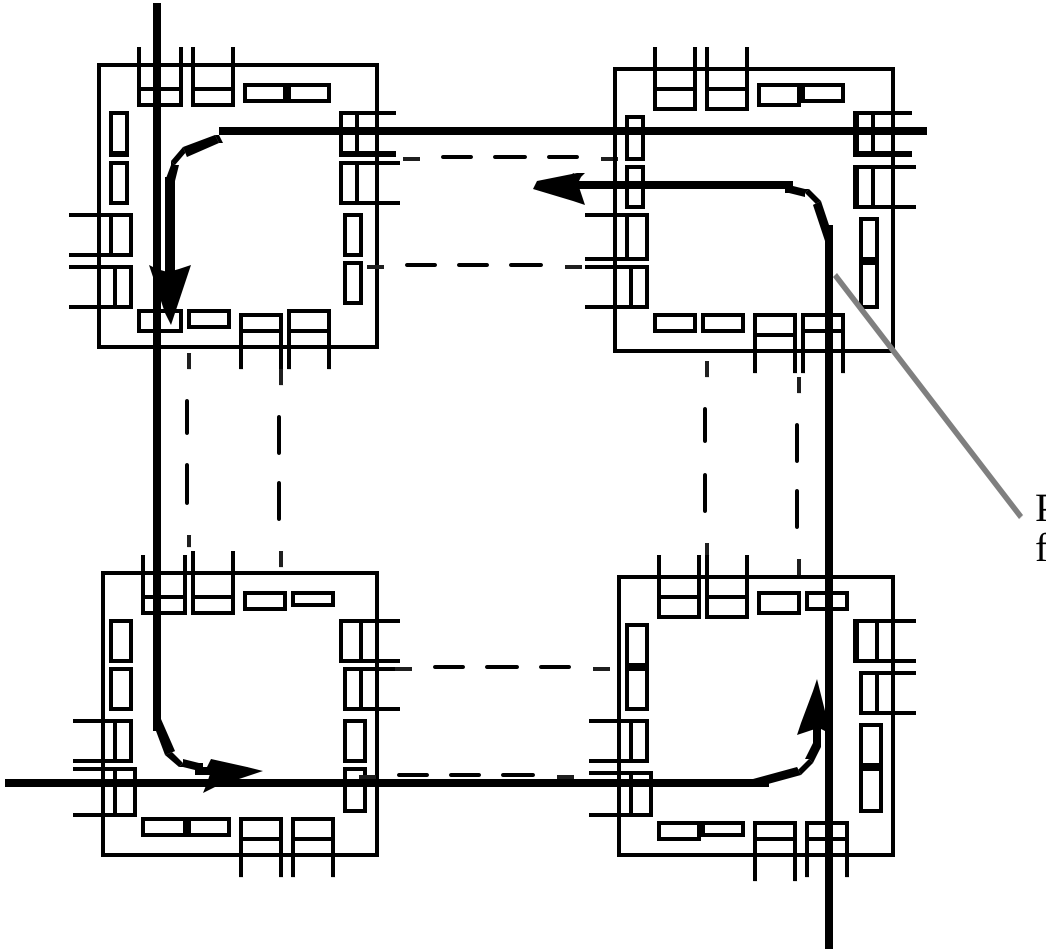
Deadlock free wormhole networks?

- **Basic dimension order routing techniques don't work for k-ary n-cubes**
  - only for k-ary n-arrays (bi-directional)
- **Idea: add channels!**
  - provide multiple "virtual channels" to break the dependence cycle
  - good for BW too!



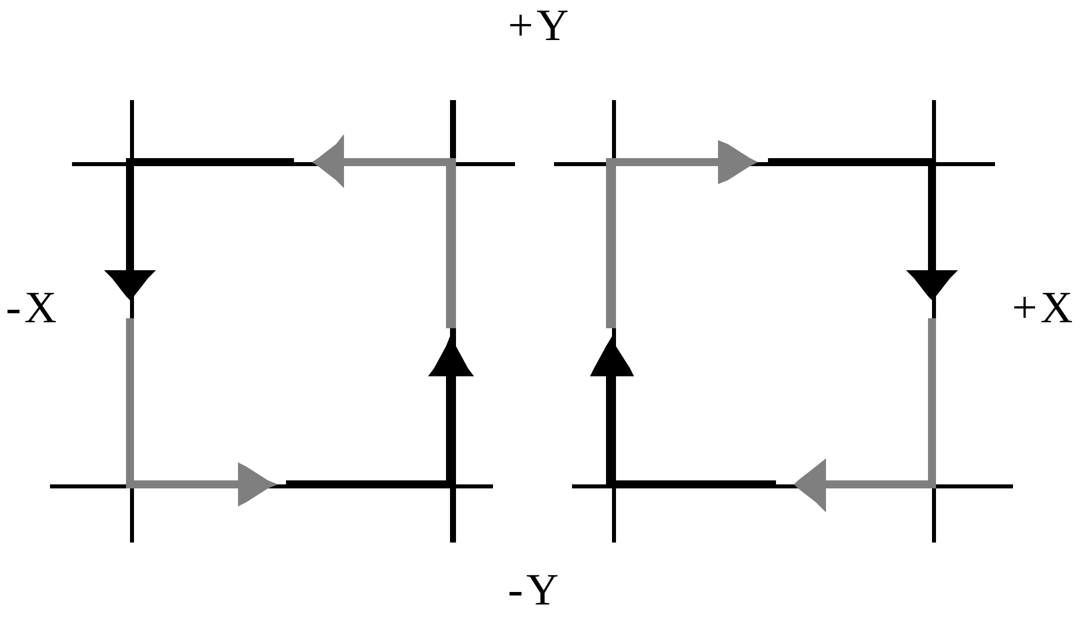
  - Do not need to add links, or xbar, only buffer resources

Breaking deadlock with virtual channels



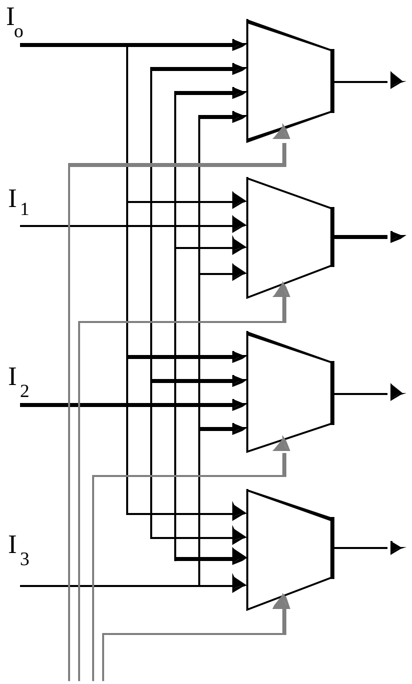
Packet switches
from lo to hi channel

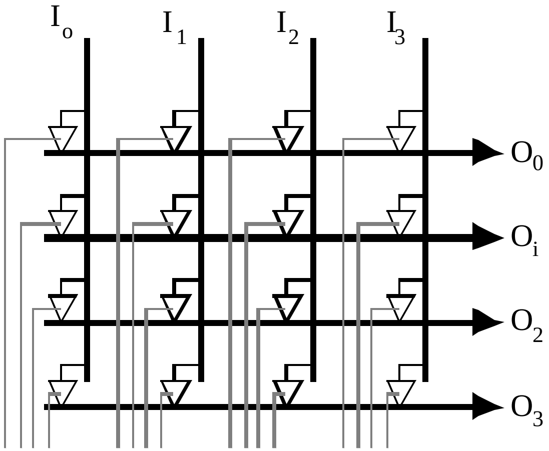
Turn Restrictions in X,Y

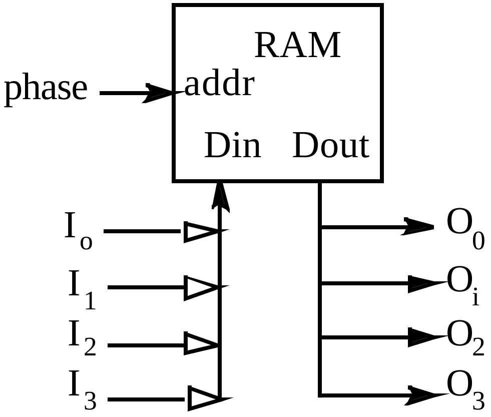


- XY routing forbids 4 of 8 turns and leaves no room for adaptive routing
- Can you allow more turns and still be deadlock free

How do you build a crossbar

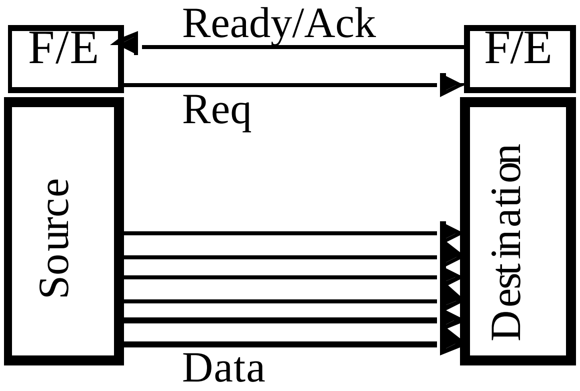




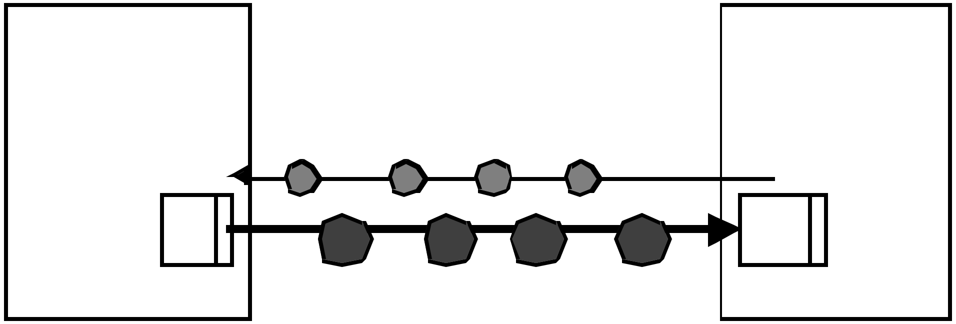


Examples

- **Short Links**

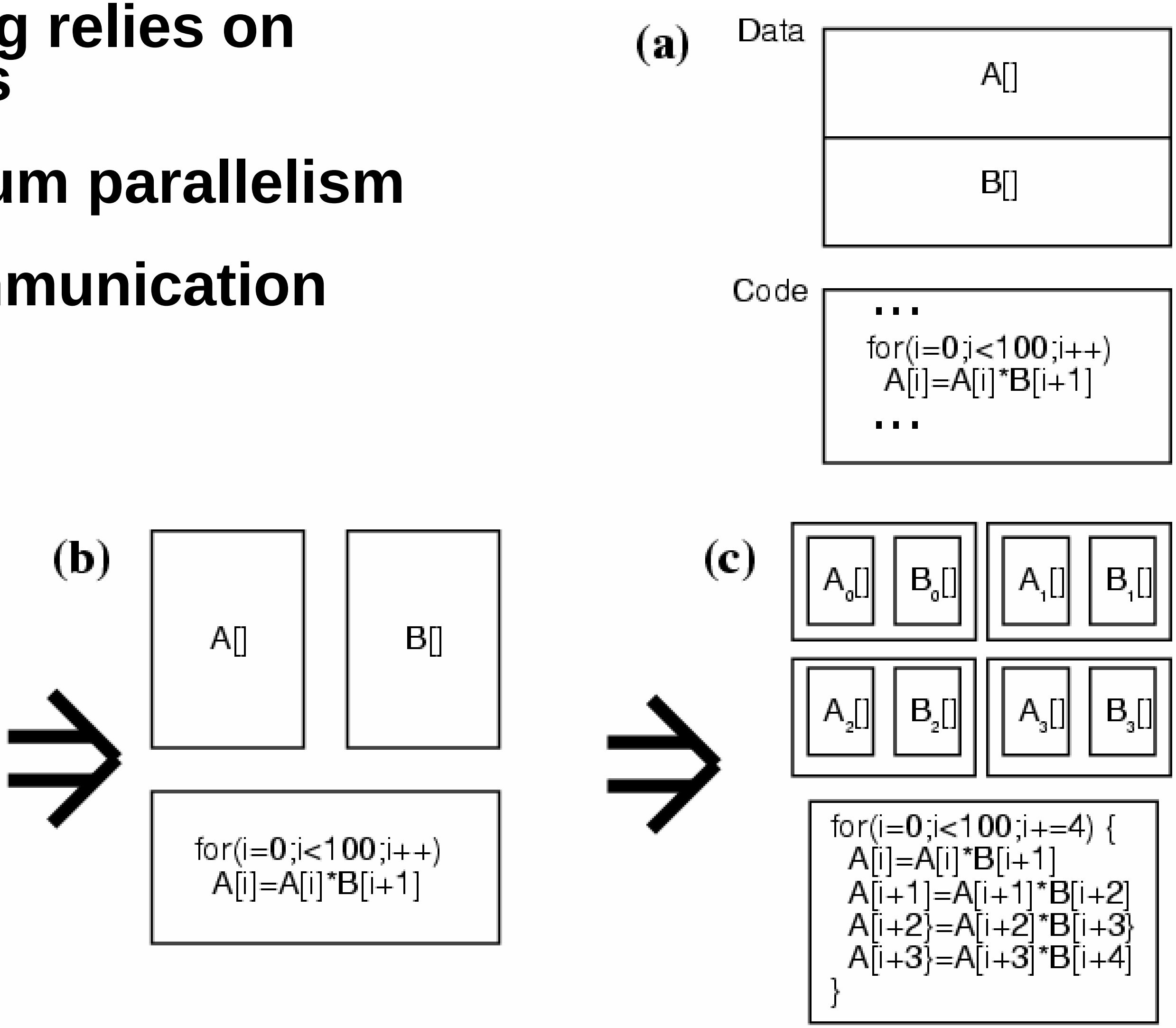


- **long links**
  - **several flits on the wire**



Modulo Unrolling – Smart Memory

- Loop unrolling relies on dependencies
- Allow maximum parallelism
- Minimize communication



```
...
for(i=0;i<100;i++)
 A[i]=A[i]*B[i+1]
...
```

```
for(i=0;i<100;i++)
 A[i]=A[i]*B[i+1]
```

```
for(i=0;i<100;i+=4) {
  A[i]=A[i]*B[i+1]
  A[i+1]=A[i+1]*B[i+2]
  A[i+2}=A[i+2]*B[i+3}
  A[i+3}=A[i+3]*B[i+4]
}
```