

Distributed memory interconnects

Two groups

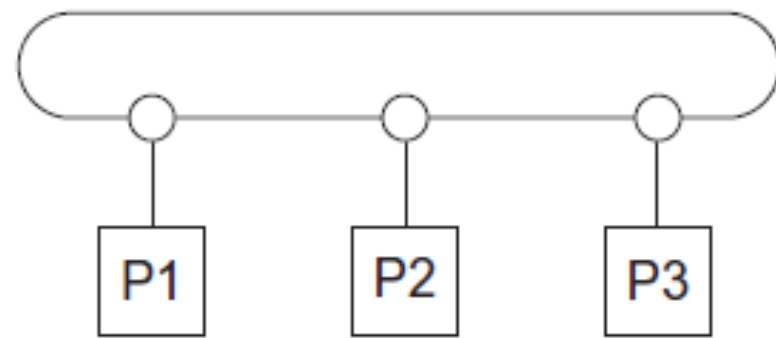
Direct interconnect

Each switch is directly connected to a processor memory pair, and the switches are connected to each other.

Indirect interconnect

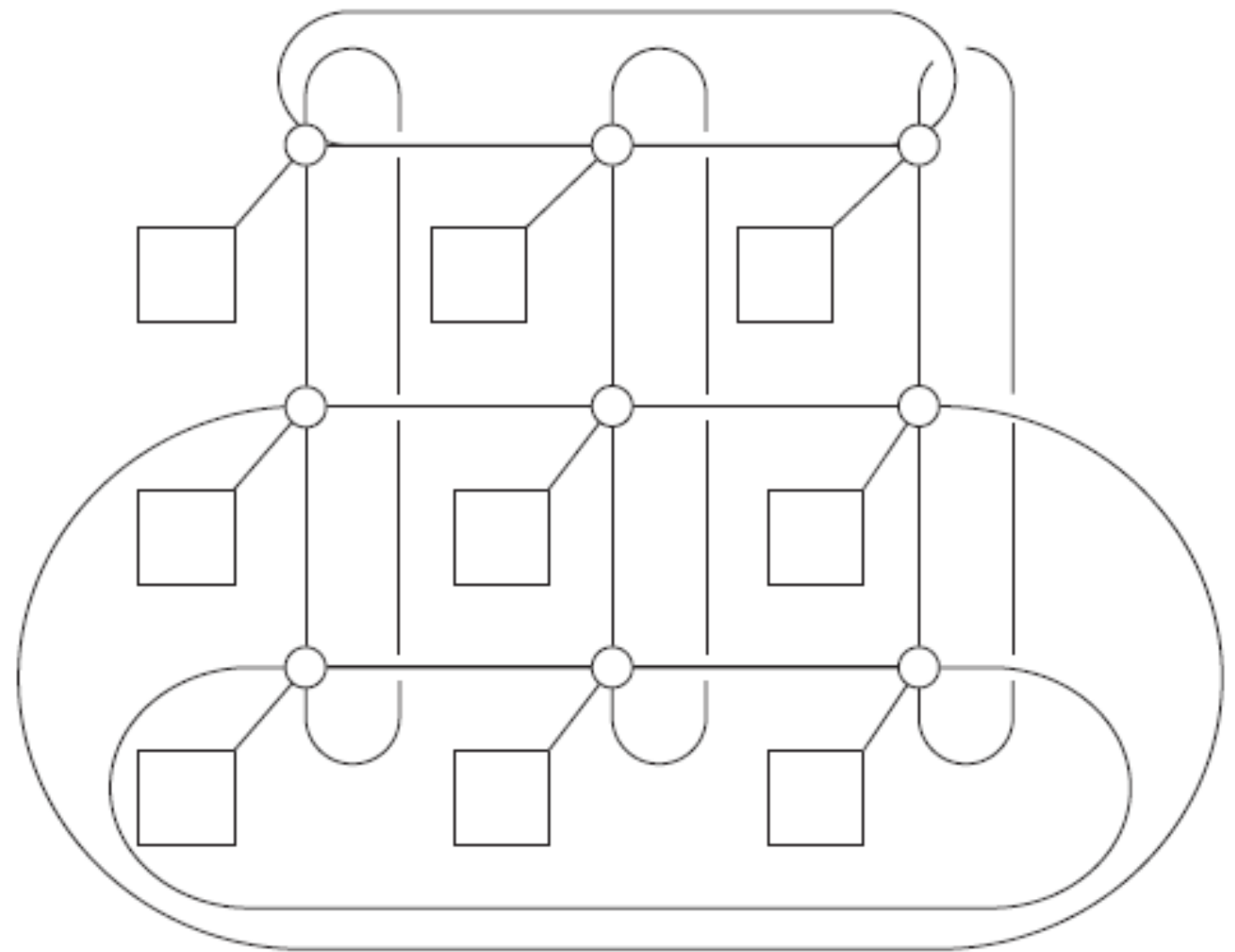
Switches may not be directly connected to a processor.

Direct interconnect



(a)

ring



(b)

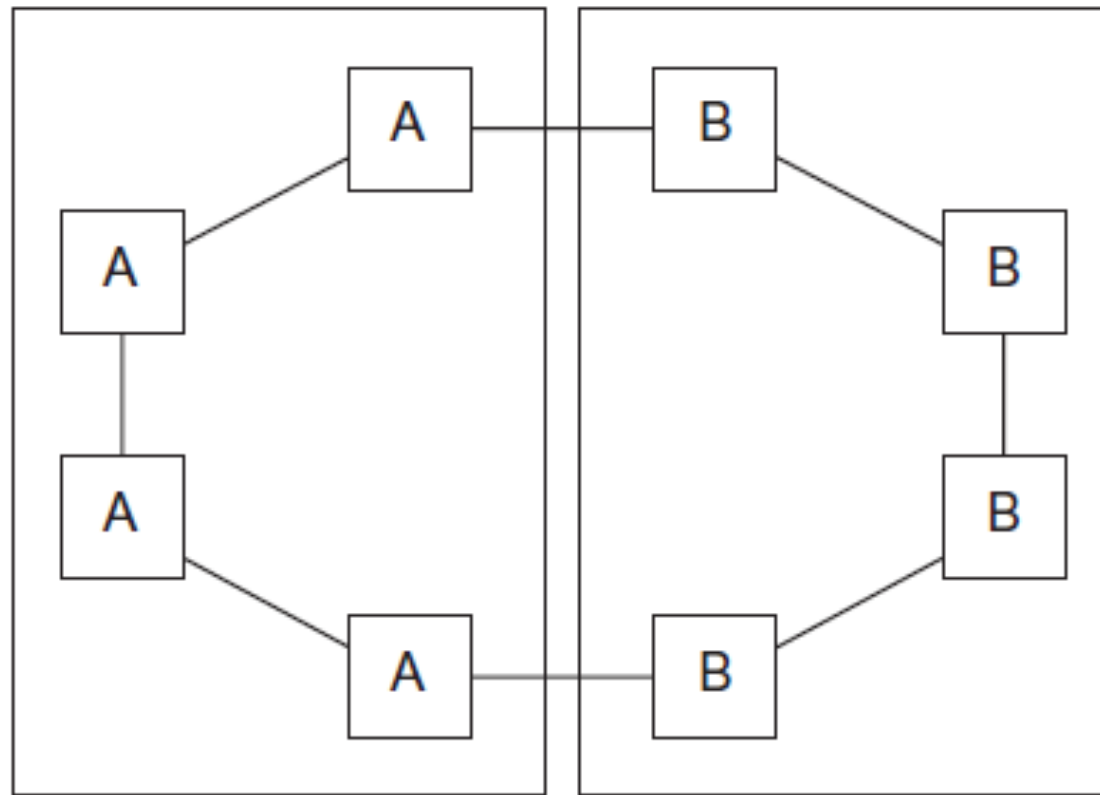
toroidal mesh

Bisection width

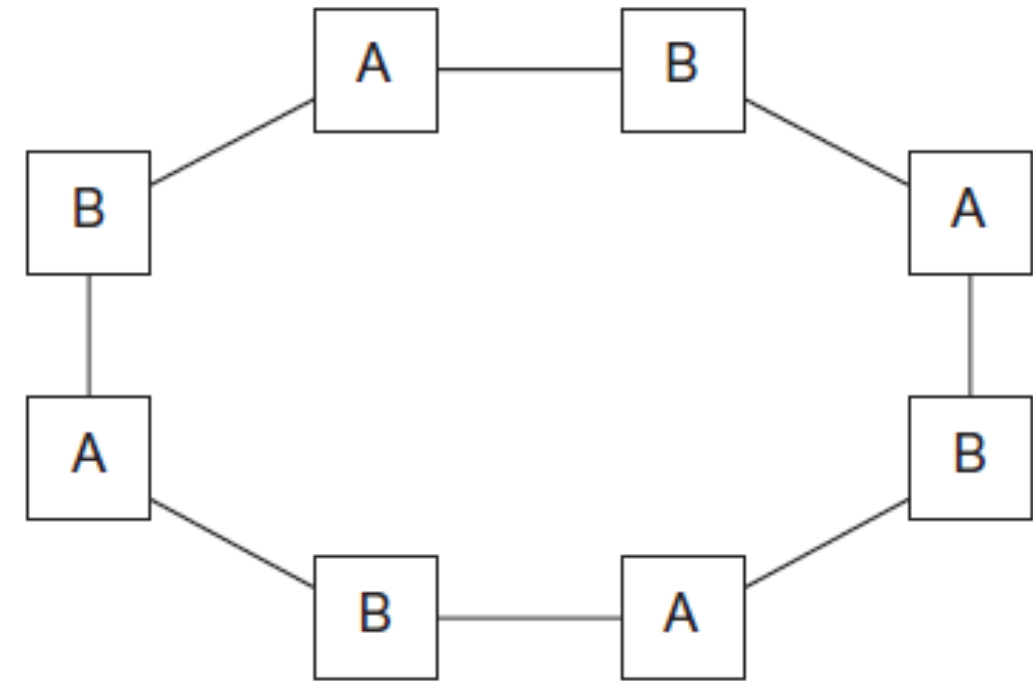
A measure of “number of simultaneous communications” or “connectivity”.

How many simultaneous communications can take place “across the divide” between the halves?

Two bisections of a ring



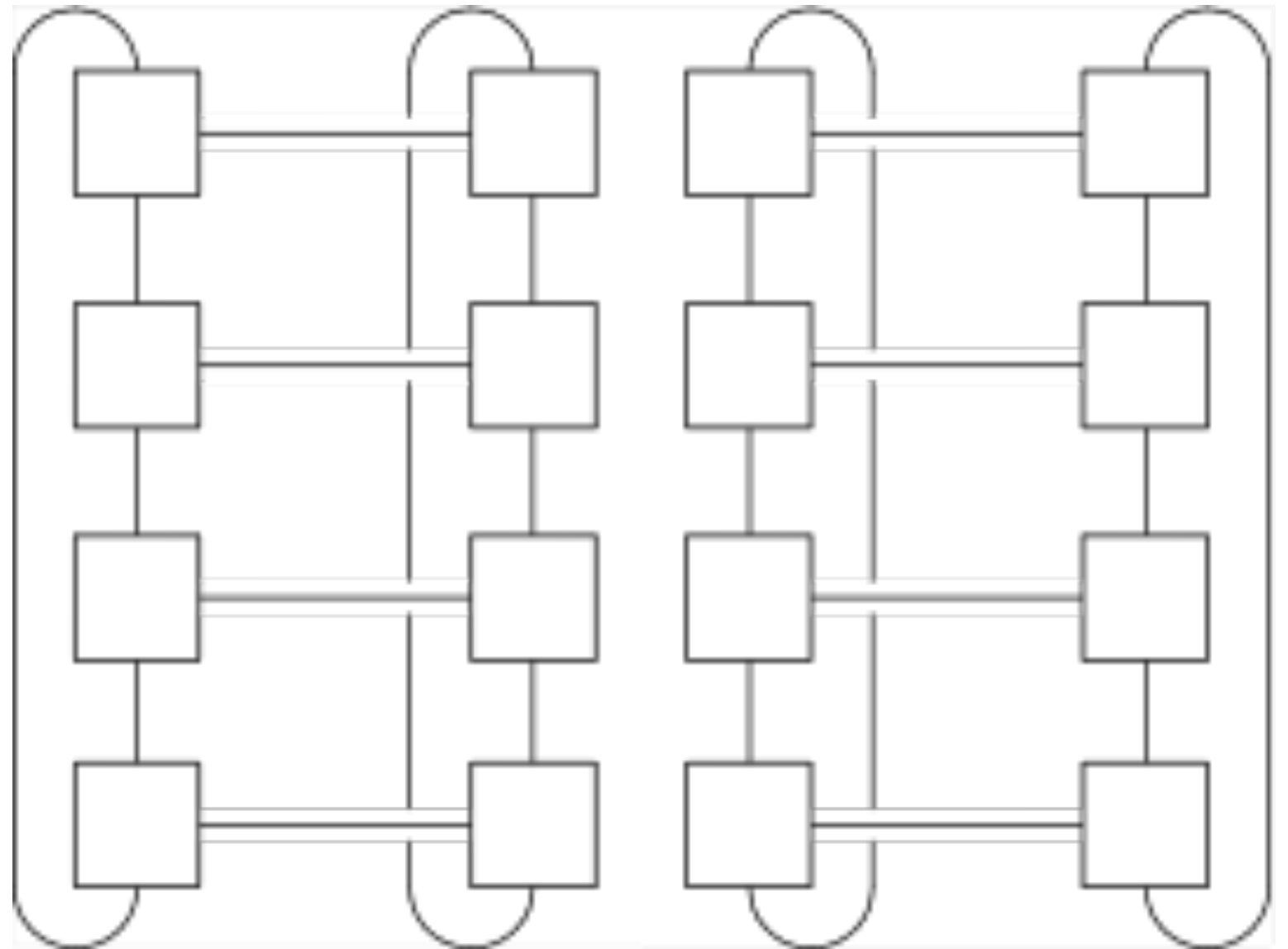
(a)



(b)

A bisection of a toroidal mesh

remove the
minimum number of
links needed to split
the set of nodes into
two equal halves



p : the number of processors

bisection width : $2\sqrt{p}$

Definitions

Bandwidth

The rate at which a link can transmit data.

Usually given in megabits or megabytes per second.

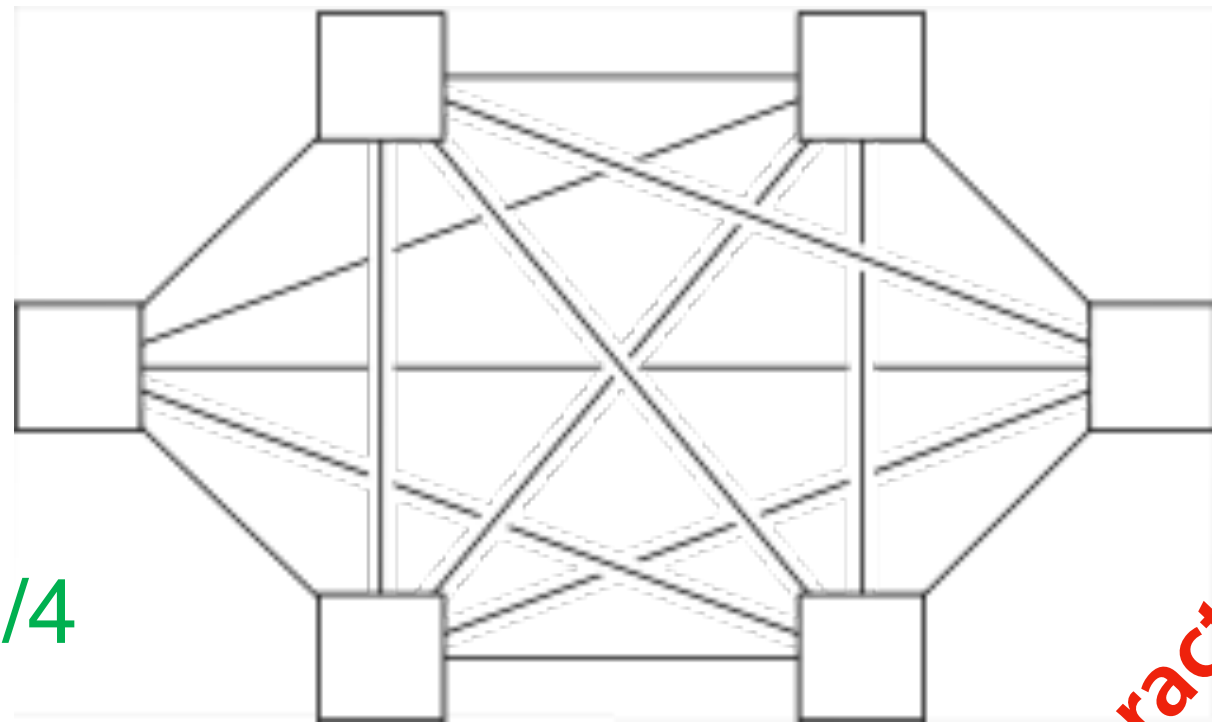
Bisection bandwidth

A measure of network quality.

Instead of counting the number of links joining the halves, it sums the bandwidth of the links.

Fully connected network

Each switch is directly connected to every other switch.



bisection width = $p^2/4$

impractical

total link = $p(p-1)/2$

Hypercube

Highly connected direct interconnect.

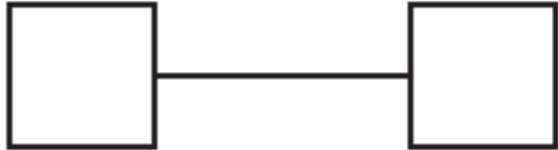
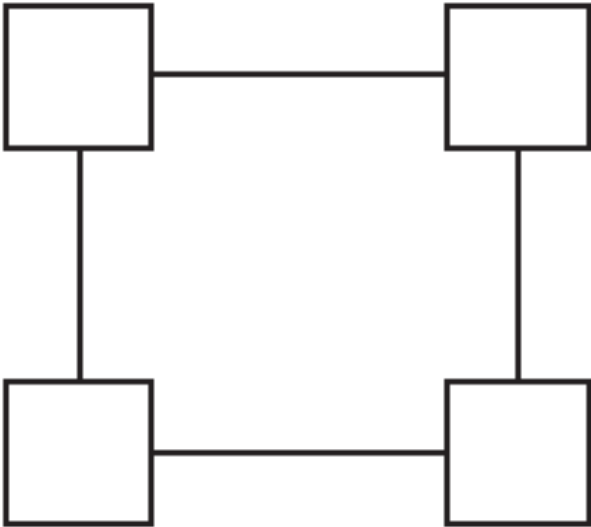
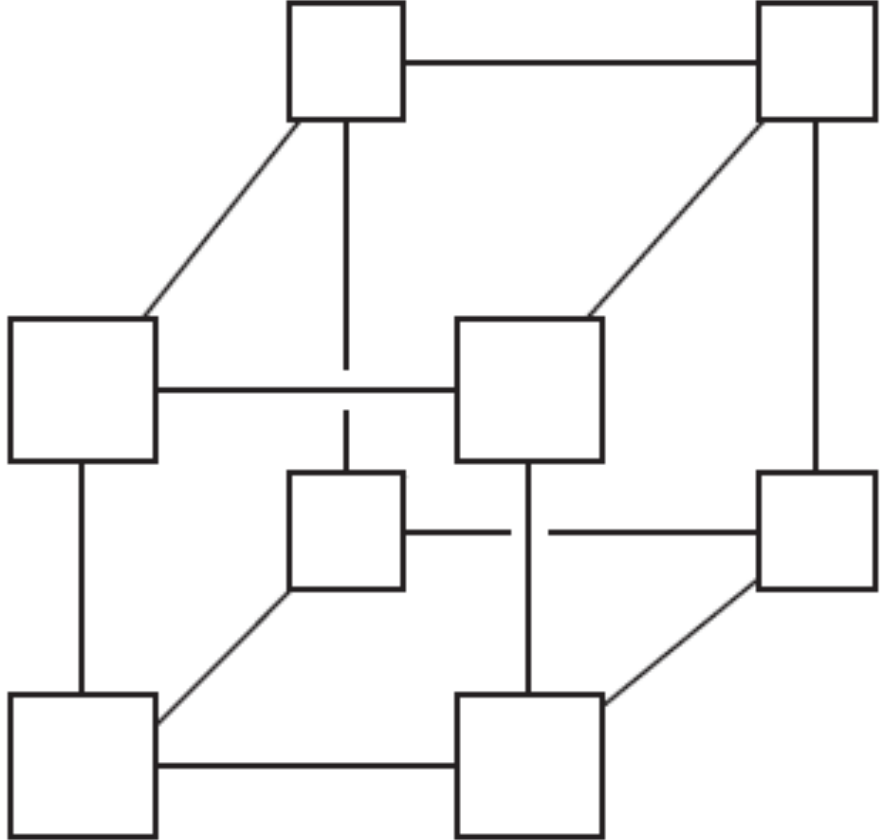
Built inductively:

A **one-dimensional** hypercube is a fully-connected system with two processors.

A **two-dimensional** hypercube is built from two one-dimensional hypercubes by joining “corresponding” switches.

Similarly a **three-dimensional** hypercube is built from two two-dimensional hypercubes.

Hypercubes

	d=1	d=2	d=3
	 <p>one-dimensional</p>	 <p>two-dimensional</p>	 <p>three-dimensional</p>
BW	1	2	4

$$p = 2^d$$

$$\text{bisection width : } \frac{p}{2} = 2^{d-1}$$

Indirect interconnects

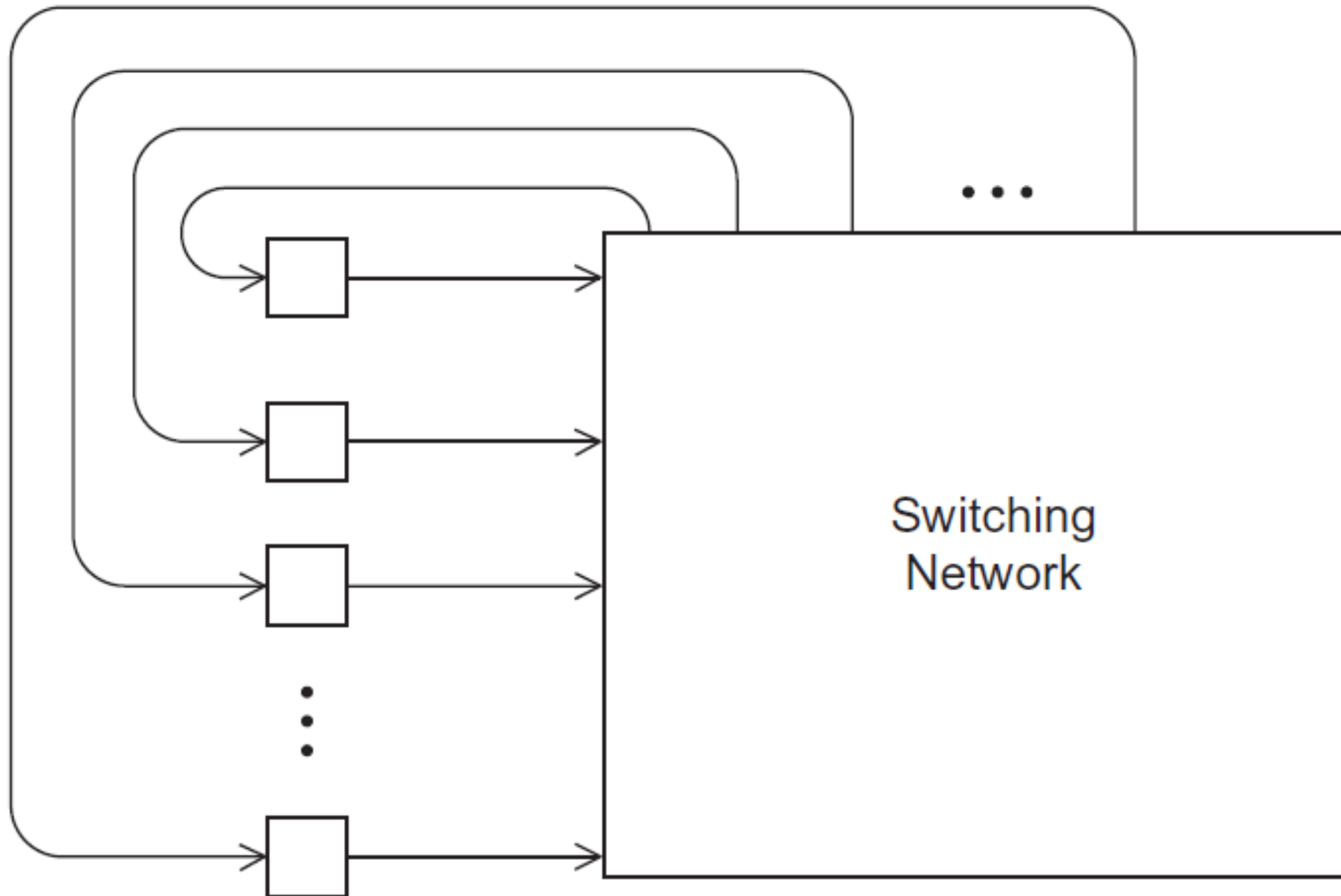
Simple examples of indirect networks:

Crossbar

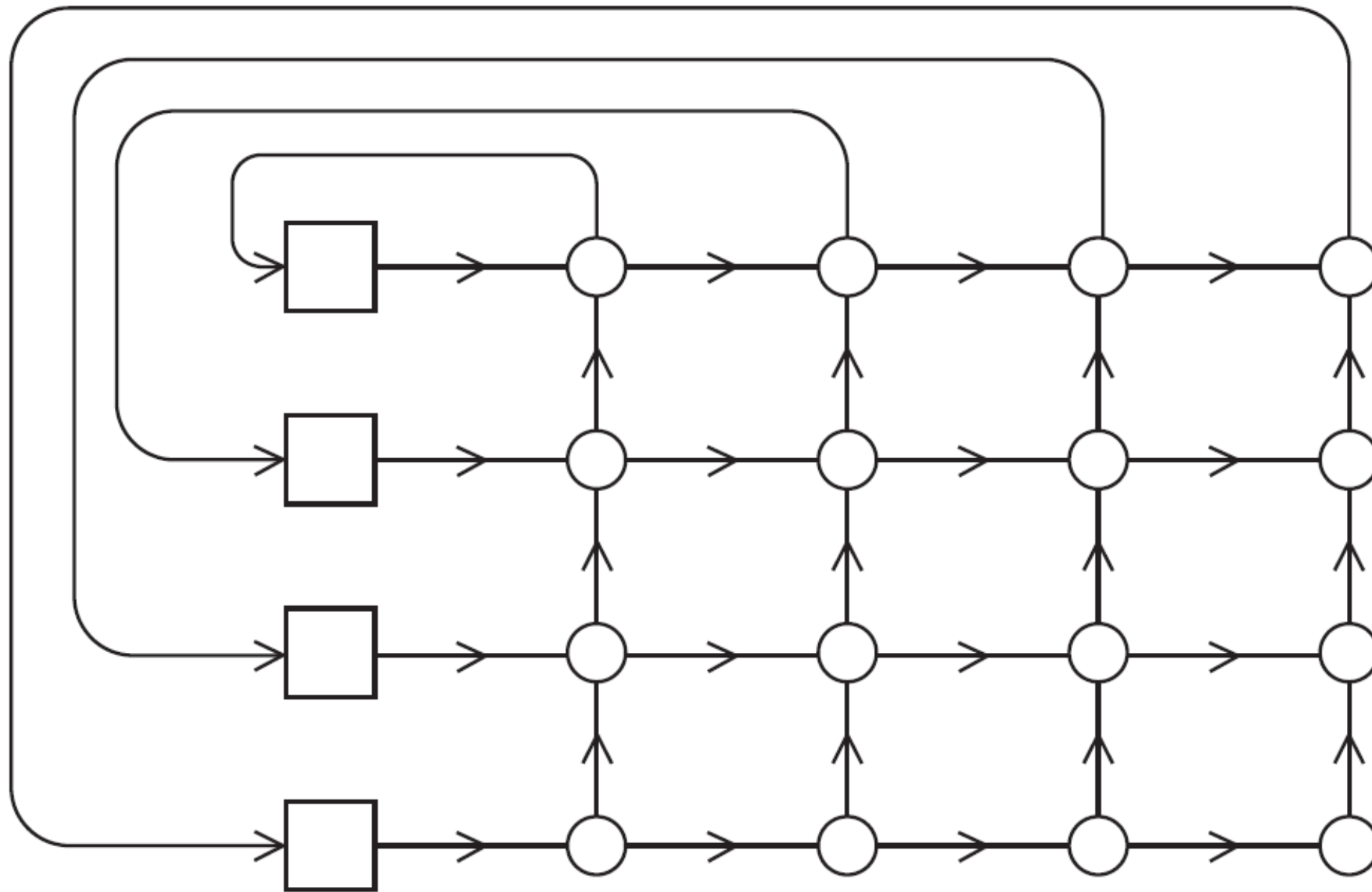
Omega network

Often shown with unidirectional links and a collection of processors, each of which has an outgoing and an incoming link, and a switching network.

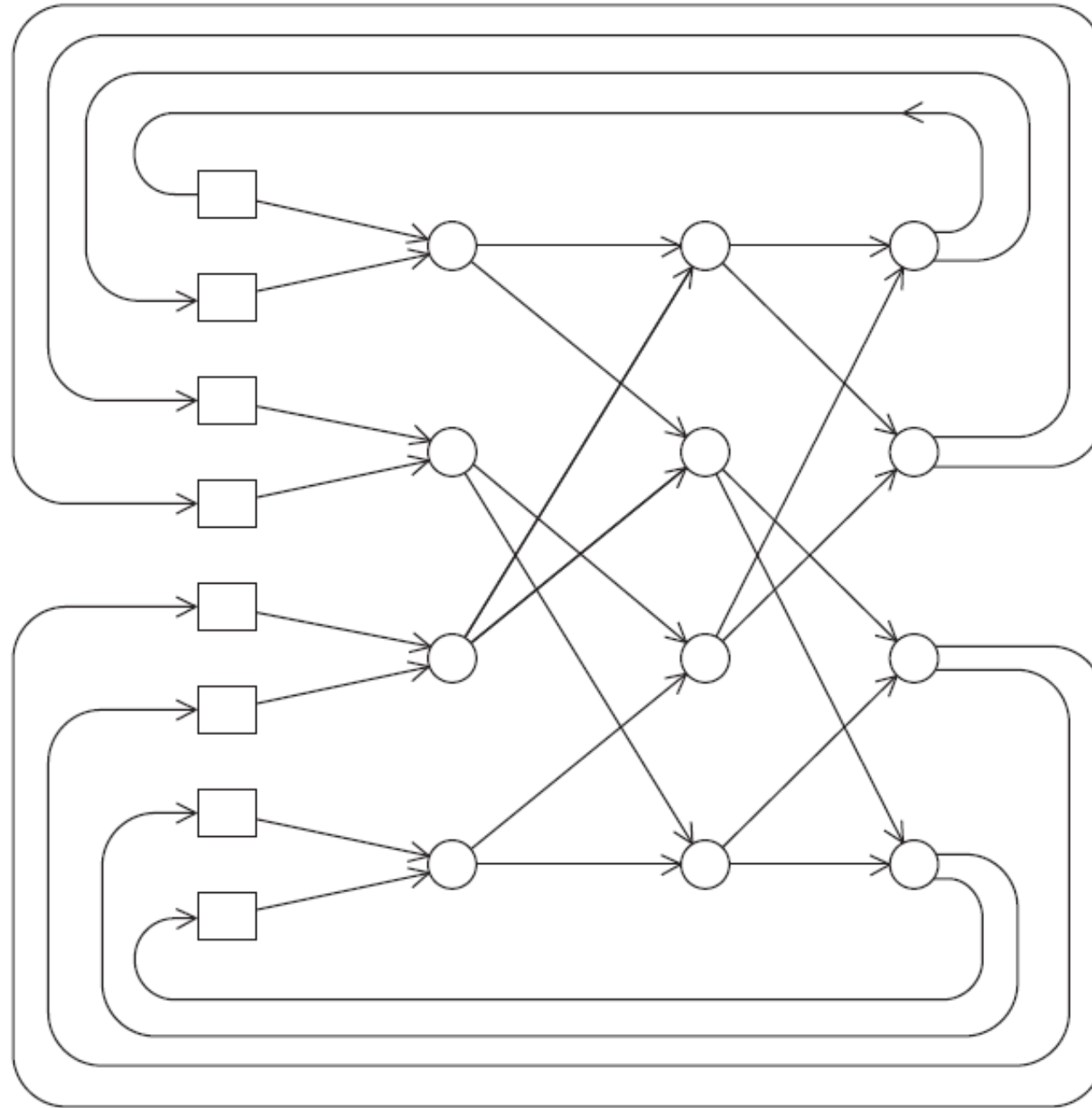
A generic indirect network



Crossbar interconnect for distributed memory

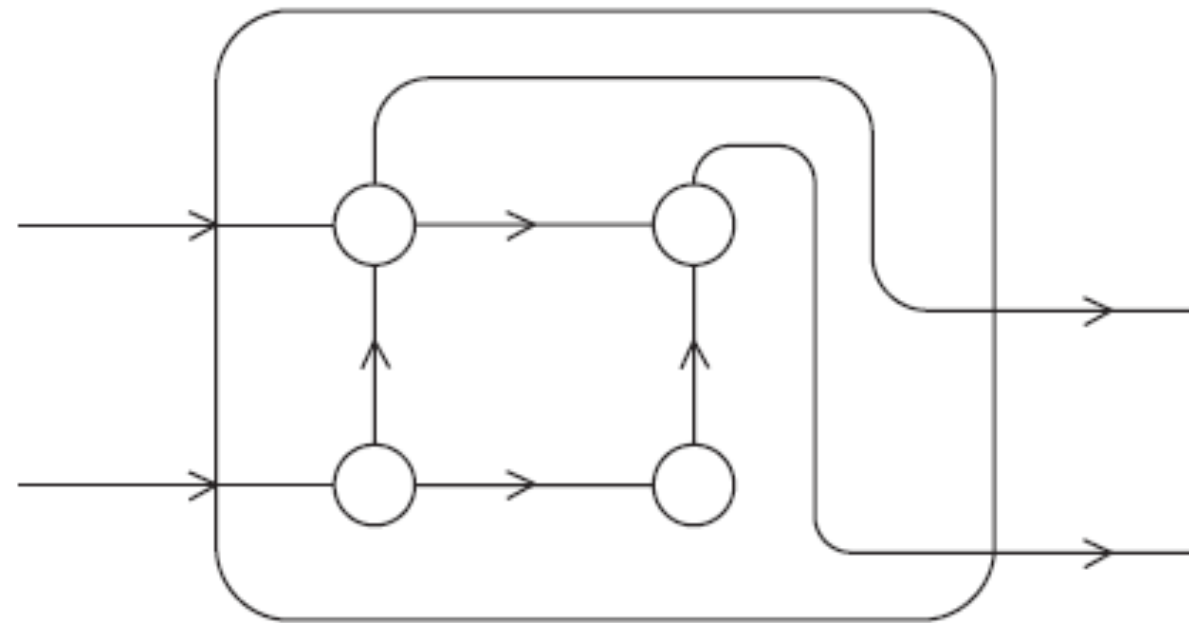


An omega network



$\frac{1}{2}p \log_2 p$ crossbar switches

A switch in an omega network



More definitions

Any time data is transmitted, we're interested in how long it will take for the data to reach its destination.

Latency

The time that elapses between the source's beginning to transmit the data and the destination's starting to receive the first byte.

Bandwidth

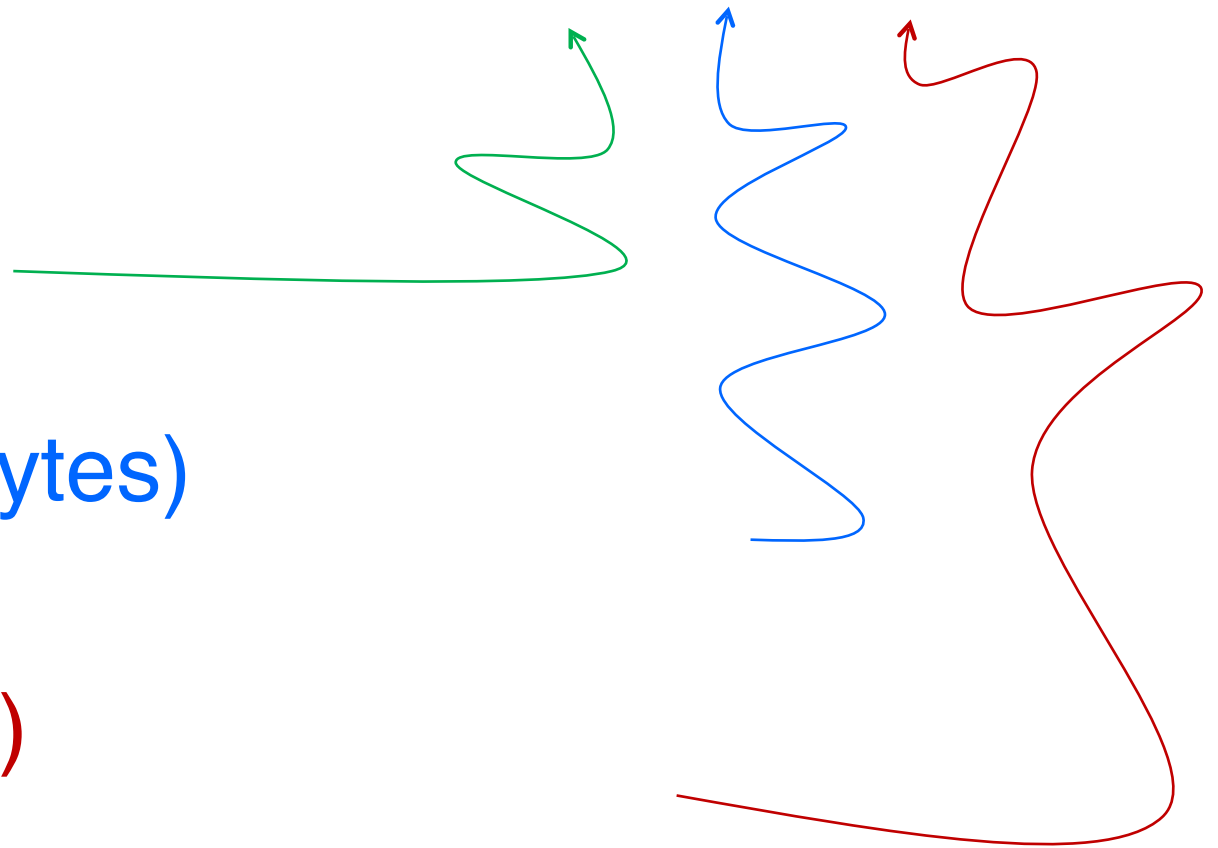
The rate at which the destination receives data after it has started to receive the first byte.

$$\text{Message transmission time} = l + n / b$$

latency (seconds)

length of message (bytes)

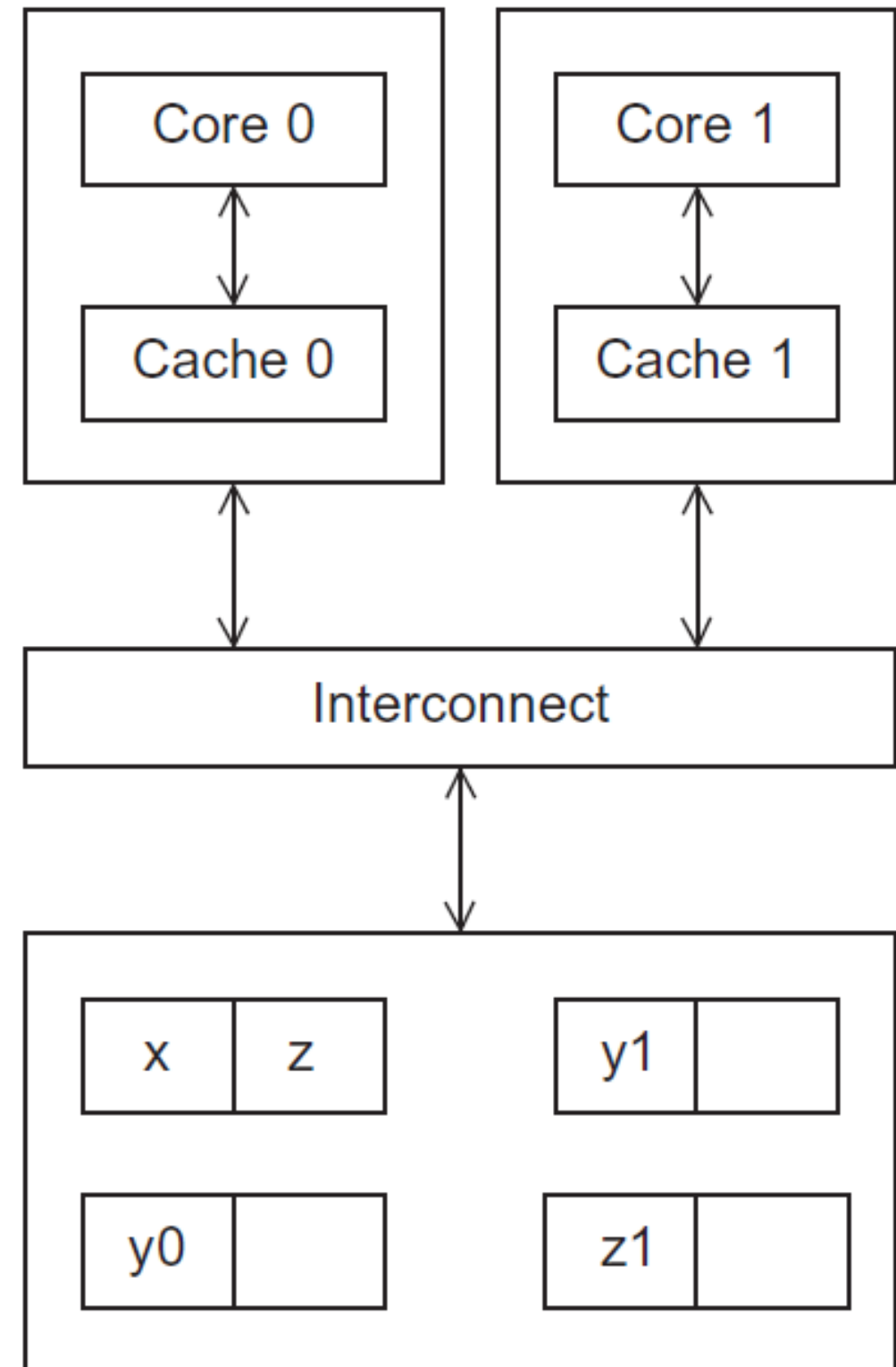
bandwidth (bytes per second)



Cache coherence

Programmers have no control over caches and when they get updated.

A shared memory system with two cores and two caches



Cache coherence

y0 privately owned by Core 0

y1 and z1 privately owned by Core 1

x = 2; /* shared variable */

Time	Core 0	Core 1
0	y0 = x;	y1 = 3*x;
1	x = 7;	Statement(s) not involving x
2	Statement(s) not involving x	z1 = 4*x;

y0 eventually ends up = 2

y1 eventually ends up = 6

z1 = ???

Snooping Cache Coherence

The cores share a bus .

Any signal transmitted on the bus can be “seen” by all cores connected to the bus.

When core 0 updates the copy of x stored in its cache it also broadcasts this information across the bus.

If core 1 is “snooping” the bus, it will see that x has been updated and it can mark its copy of x as invalid.

Directory Based Cache Coherence

Uses a data structure called a **directory** that stores the status of each cache line.

When a variable is updated, the directory is consulted, and the cache controllers of the cores that have that variable's cache line in their caches are invalidated.

Parallel Software

The burden is on software

Hardware and compilers can keep up the pace needed.

From now on...

In shared memory programs:

- Start a single process and fork threads.

- Threads carry out tasks.

In distributed memory programs:

- Start multiple processes.

- Processes carry out tasks.

SPMD – single program multiple data

A SPMD programs consists of a single executable that can behave as if it were multiple different programs through the use of conditional branches.

```
if (I'm thread/process i)
    do this;
else
    do that;
```

Writing Parallel Programs

1. Divide the work among the processes/threads

(a) so each process/thread gets roughly the same amount of work

(b) and communication is minimized.

```
double x[n], y[n];  
...  
for (i = 0; i < n; i++)  
    x[i] += y[i];
```

2. Arrange for the processes/threads to synchronize.

3. Arrange for communication among processes/threads.

These last two problems are often interrelated.