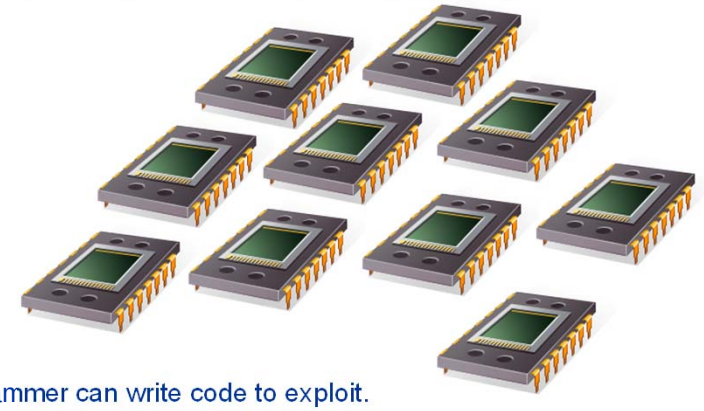


A programmer can write code to exploit.

PARALLEL HARDWARE

Flynn's Taxonomy

SISD Single instruction stream Single data stream	(SIMD) Single instruction stream Multiple data stream
MISD Multiple instruction stream Single data stream	(MIMD) Multiple instruction stream Multiple data stream

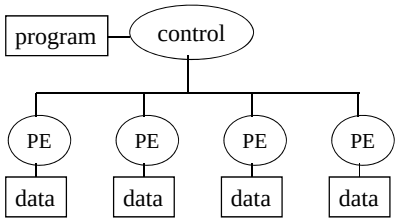
classic von Neumann

not covered

SIMD (two flavors)

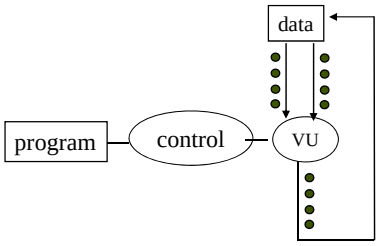
1) Synchronous, lockstep execution

All processing elements (PE) execute the same instructions on different data (data parallelism)



2) Vector processing

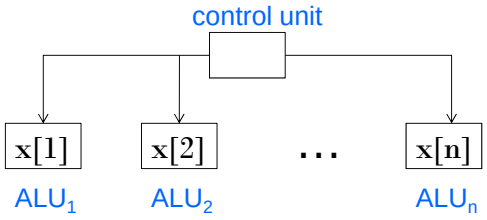
The same instruction is repeatedly executed on different data
VU = Vector unit



33

SIMD example

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



- What if we don't have as many ALUs as data items?
- Divide the work and process iteratively.
- Ex. m = 4 ALUs and n = 15 data items.

Round	ALU_1	ALU_2	ALU_3	ALU_4
1	X[0]	X[1]	X[2]	X[3]
2	X[4]	X[5]	X[6]	X[7]
3	X[8]	X[9]	X[10]	X[11]
4	X[12]	X[13]	X[14]	

34

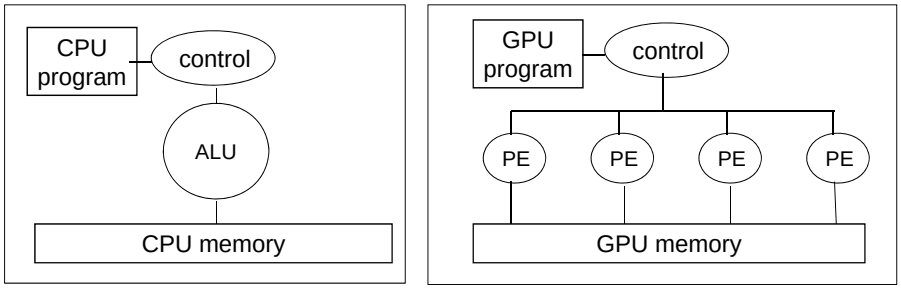
Graphics Processing Units (GPU)

- Real time graphics application programming interfaces or API's use points, lines, and triangles to internally represent the surface of an object.
- Graphics processing functions convert the internal representation into an array of pixels that can be sent to a computer screen.

- These functions (called shader functions) are implicitly parallel, since they can be applied to multiple elements in the graphics stream.

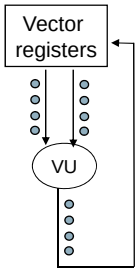
GPUs

- GPU's optimize performance by using SIMD parallelism.
- The current generation of GPU's use SIMD parallelism, but are not pure SIMD systems



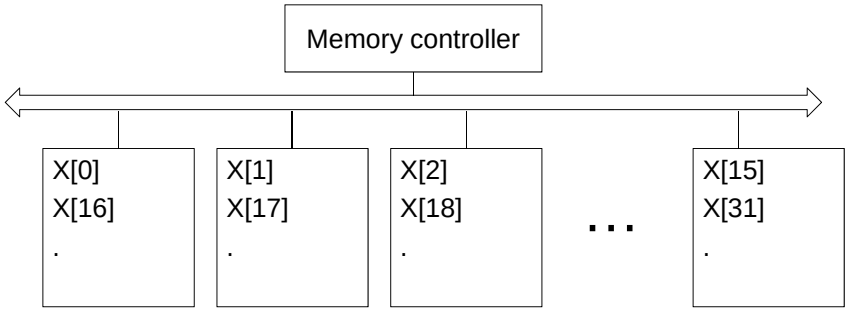
Vector processors

- Operate on arrays or vectors of data while conventional CPU's operate on individual data elements or scalars.
- Vector registers.
  - Capable of storing a vector of operands and operating simultaneously on their contents.
- Vectorized and pipelined functional units.
  - The same operation is applied to each element in the vector (or pairs of elements).
- Vector instructions.
  - Operate on vectors rather than scalars.



Interleaved memory

- Multiple "banks" of memory, which can be accessed independently.
- Distribute elements of a vector across multiple banks to reduce or eliminate delay in loading/storing successive elements.
- **Strided** memory access: the program accesses elements of a vector located at fixed distances from each other.



Vector processors

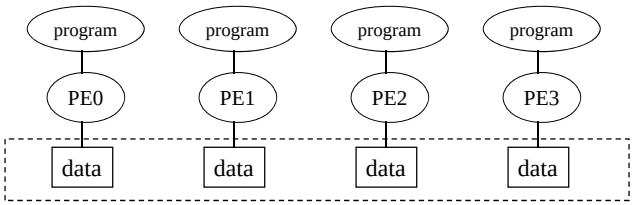
Pros

- Fast and easy to use.
- Vectorizing compilers can identify code to exploit.
- Compilers also can provide information about code that cannot be vectorized.
  - Helps the programmer re-evaluate code.
- High memory bandwidth.

Cons

- They don't handle irregular data structures as well as other parallel architectures.
- A very finite limit to their ability to handle ever larger problems. (scalability)

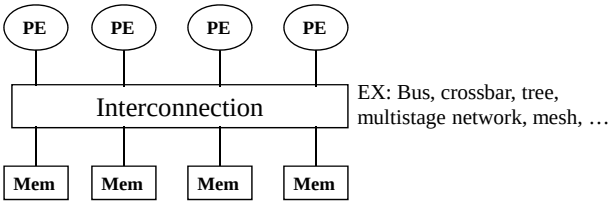
MIMD



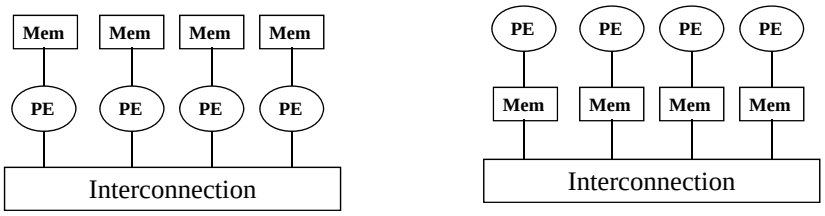
Multiple programs executing on different data – However, if all PEs are to cooperate to solve the problem (as opposed to solving different problems), there should be interaction between the programs and/or the data.

Many flavors depending on the **memory architecture** and the **address space** of each PE (the range of memory addresses that an instruction executing on the PE can access).

Physical memory Architectures



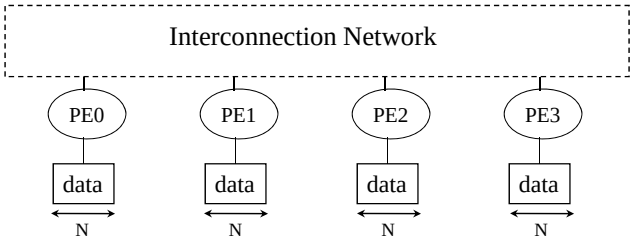
Global, shared memory architecture (Symmetric Multi-Processors – SMP)



Distributed memory architecture

41

Distributed address space

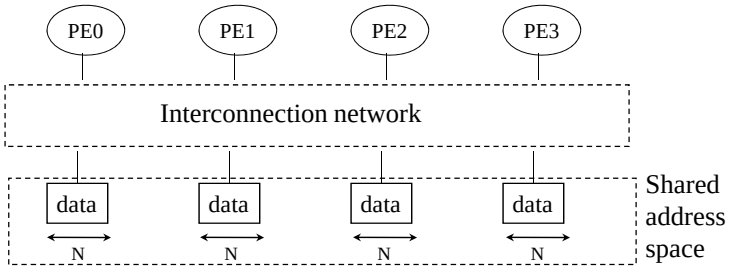


PE0 can directly access N locations with addresses 0,…,N-1
PE1 can directly access N other locations with addresses 0,…,N-1
PE2 can directly access N other locations with addresses 0,…,N-1
PE3 can directly access N other locations with addresses 0,…,N-1

- In order for PE_i to access data in the address space of PE_j, the two processors have to communicate through sending/receiving messages.

42

Shared address space

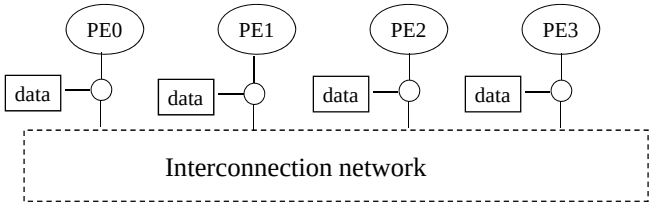


Each of PE0, PE1, PE2 and PE3 can address (directly access using load and store instructions) locations 0,…, 4N-1

- No need for message passing – communicate through shared memory locations.

43

Distributed shared memory systems



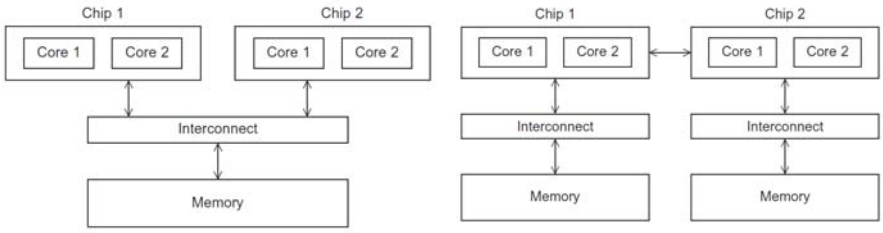
Shared address space, but physically distributed memory.

- No need for message passing – communicate through shared memory locations.
- Data is physically distributed, but a runtime system is responsible to access data that do not reside in the local memory.

Results in the so called “Non Uniform Memory Access” – NUMA (as opposed to UMA, “Uniform Memory Access”)

44

multicore systems



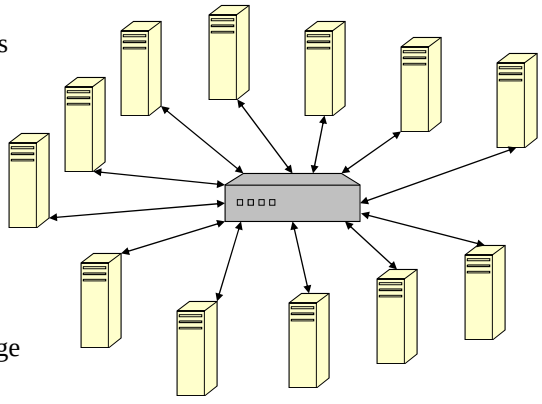
UMA multicore system

NUMA multicore system

45

Clusters

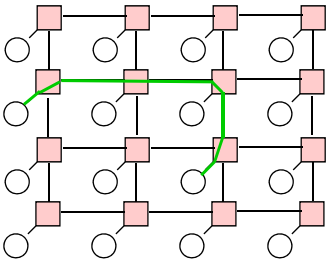
- Off the shelf computing nodes
- Commodity switches
  - Ethernet
  - Myrinet
  - Quadrics
  - Infiniband
  - Fiber Channel
- Distributed address space
- Communicate through message passing



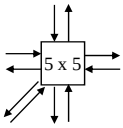
46

Interconnection networks (two classes)

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

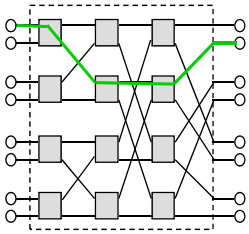


A 5x5 switch or router

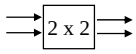


A processor-memory pair

Indirect interconnects: Switches may be connected to other switches, to a processor or to a memory module



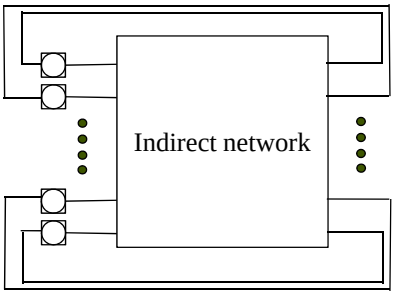
A 2x2 switch or router



Input or output ports

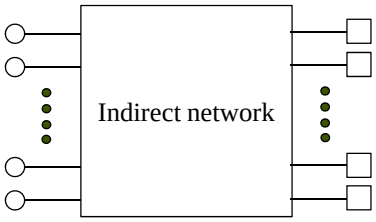
Indirect networks

Can be used for distributed memory systems to connect processor/memory nodes



A processor + memory

Can also be used for shared memory systems to connect processors to memory modules

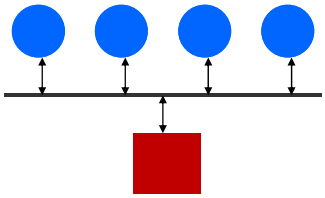


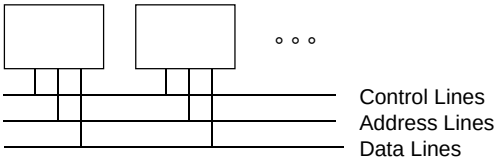
Processor

memory

Bus Interconnect

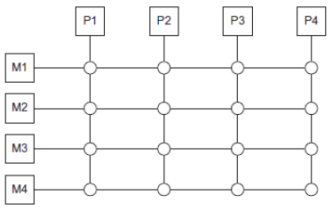
- Mostly used in shared memory systems
- A collection of parallel communication wires together with some hardware that controls access to the bus
- Communication wires are shared by the devices that are connected to it.
- As the number of devices connected to the bus increases, contention increases, and performance decreases.



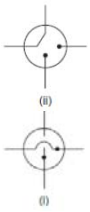


Crossbars

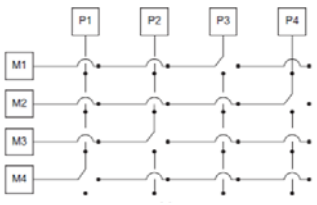
- Uses switches to control the routing of data among the connected devices.
- Allows simultaneous communication among different devices
- Faster than buses.
- But the cost of the switches and links is relatively high.



A crossbar switch connecting 4 processors (Pi) and 4 memory modules (Mj)



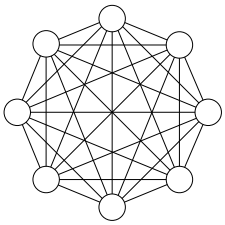
Configuration of internal switches in a crossbar



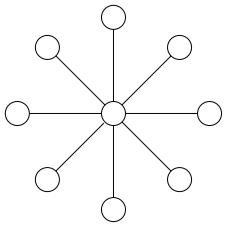
Simultaneous memory accesses by the processors

Some direct network topologies

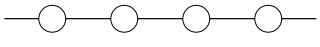
impractical



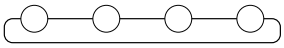
A completely-connected network



A star connected network



with no wraparound links

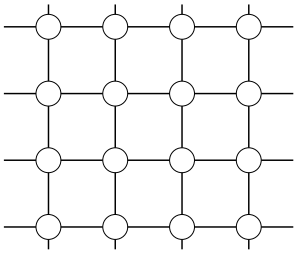


with wraparound link (ring).

Linear arrays

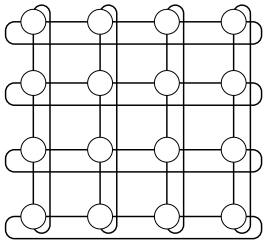
51

Two- and Three Dimensional Meshes



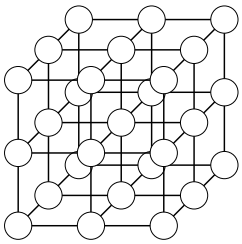
(a)

2-D mesh with no wraparound



(b)

2-D mesh with wraparound link (2-D torus)

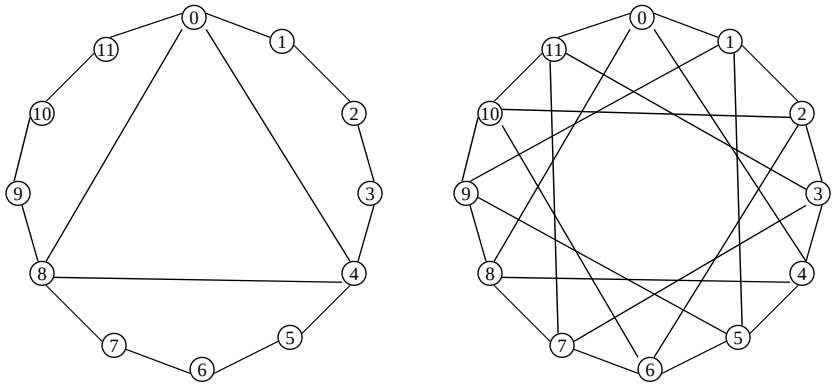


(c)

3-D mesh with no wraparound.

52

Enhanced ring networks



- Cords (in a chordal ring) may bypass any given number of nodes
- May have more than one set of chords

53

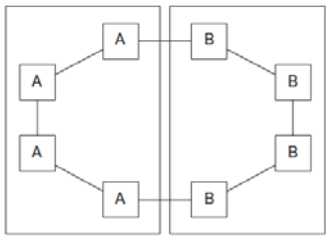
Evaluating Interconnection Network topologies

- ***Diameter****:* The distance between the farthest two nodes in the network.
- ***Average distance:*** The average distance between any two nodes in the network.
- ***Node degree:*** The number of neighbors connected to any particular node.
- ***Bisection Width:*** The minimum number of links you must cut to divide the network into two almost equal parts – Reflects the severity of the communication bottleneck.
- ***Cost:*** The number of links and switches is a meaningful measure of the cost. Other factors affecting cost are the ability to layout the network, the length of wires, etc.

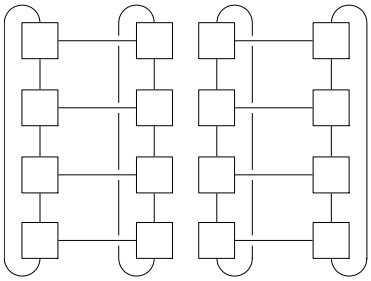
54

The bisections bandwidth

- A measure of network quality.
- Instead of counting the number of links joining the halves, it sums the bandwidth of the links.



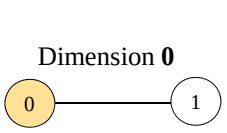
For a ring

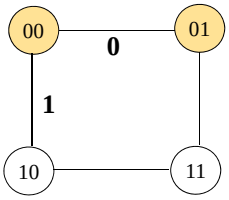


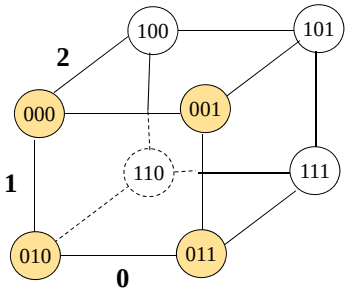
For a toroidal mesh (torus)

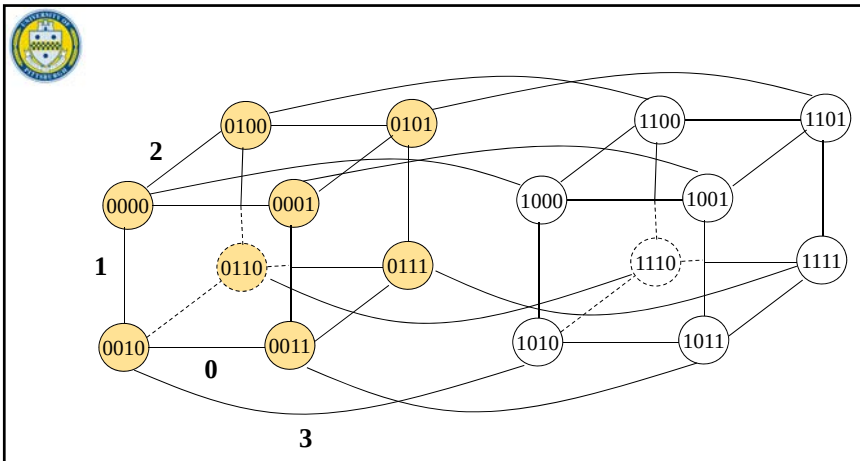
Hypercube interconnections

- An interconnection with low diameter and large bisection bandwidth.
- A *q*-dimensional hypercube is built from two *(q-1)*-dimensional hypercubes.





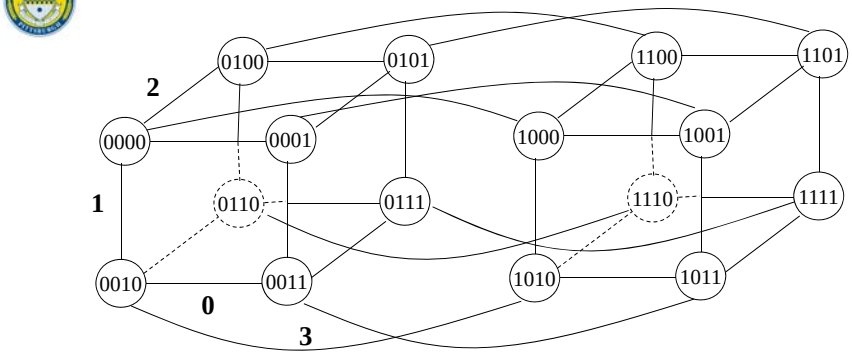




For a q dimension hypercube, calculate

- The number of nodes, n, and the number of edges, e
- The node degree
- The diameter
- The bisection bandwidth

57

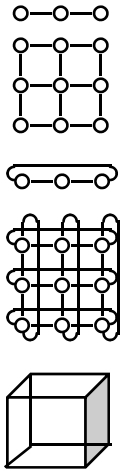


- Each node in a q-dimension hypercube has a q-bits identifier $x_{q-1}, \ldots, x_1, x_0$
- Identifiers of nodes across a dimension j differ in the j^{th} bit (have a unit **Hamming distance***)
- How do you find the route between a source, s, and a destination, d?

*) The Hamming distance between two binary numbers is the number of positions at which the bits in the two strings are different.

58

Common interconnection topologies

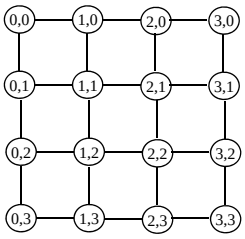


Type	Degree	Diameter	Av Dist	Bisection	N = 1024 Diam	N = 1024 Av. D
1D mesh	2	N-1	N/3	1		
2D mesh	4	$2(N^{1/2} - 1)$	$2N^{1/2} / 3$	$N^{1/2}$	63	21
3D mesh	6	$3(N^{1/3} - 1)$	$3N^{1/3} / 3$	$N^{2/3}$	~30	~10
Ring	2	N / 2	N/4	2		
2D torus	4	$N^{1/2}$	$N^{1/2} / 2$	$2N^{1/2}$	32	16
k-ary n-cube ($N = k^n$)	2n	$n(N^{1/n})$ nk/2	$nN^{1/n}/2$ nk/4	$2k^{n-1}$	15	8 (3D)
Hypercube	n	n = LogN	n/2	N/2	10	5

Routing messages in 2D mesh networks

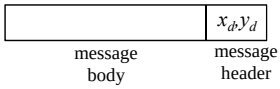
The problem:

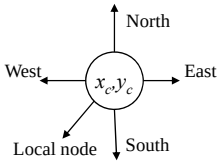
- Assume that each switch in an *n* x *n* mesh is labeled by (*x*,*y*), where $0 < x < n-1$ and $0 < y < n-1$.
- Assume also that each message has a header which contains the address, (x_d , y_d), of its destination.
- A routing algorithm determines at any intermediate node, (x_c , y_c), where to send the message next.



X-Y routing:

1) If $x_c < x_d$ then send the message to the East port
   elseif $x_c > x_d$ then send it to the West port
2) If $y_c < y_d$ then send the message to the North port
   elseif $y_c > y_d$ then send it to the South port
3) Deliver the message to the local node





Messages travel along the x-direction before traveling along the y-direction

Type of routing algorithms

X-Y routing is a minimal deterministic routing

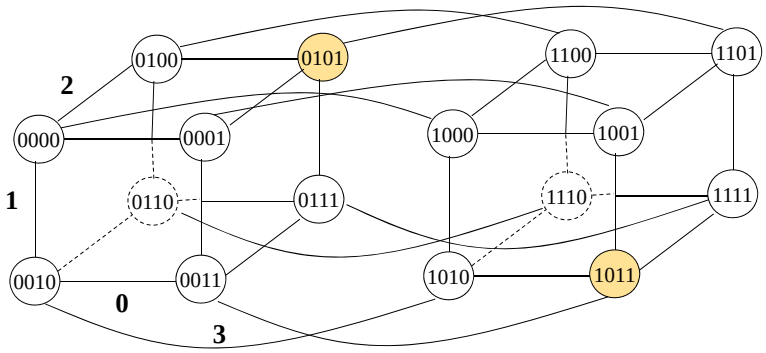
- **Minimal routing**: A message always follows one of the shortest paths between a source and a destination.
- **Deterministic routing**: a message between a source and a destination always follow the same route.

Other types of routings

- **Non-deterministic routing**: a message between a source and a destination does not have to always follow the same route.
- **Adaptive routing**: a non-deterministic routing which chooses the route based on current network condition (example congestion or faults).
- **Deflection routing**: a non-minimal adaptive routing. May follow a "non-shortest" path.

61

Routing on a hypercube

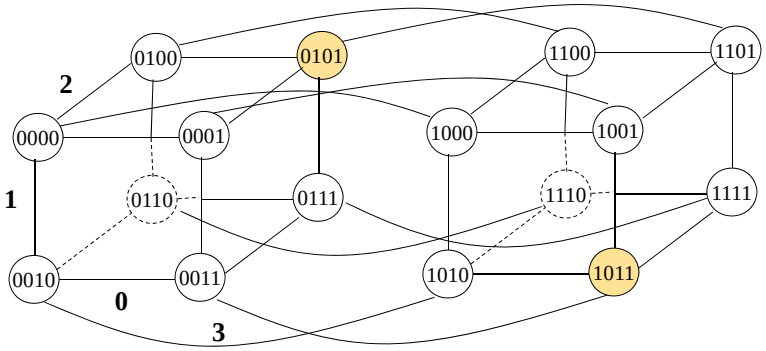


A message from a source, $s_{q-1}, \ldots, s_0$, to a destination $d_{q-1}, \ldots, d_0$ has to cross any dimension, b, for which $d_b \neq s_b$

How many distinct routes there are between any source and destination?

62

Dimension-order routing

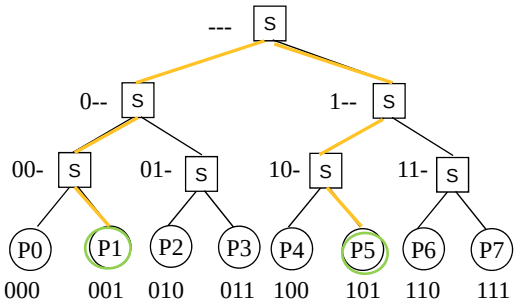


When a node, $c_{q-1}, \ldots, c_0$, receives a message for destination node $d_{q-1}, \ldots, d_0$, it executes the following

- If $d_k = c_k$ for $k = 0, \ldots, q\text{-}1,$, keep the message
- Else { Find the largest k such that $d_k \neq c_k$;
  Send the message to the neighbor across dimension k }

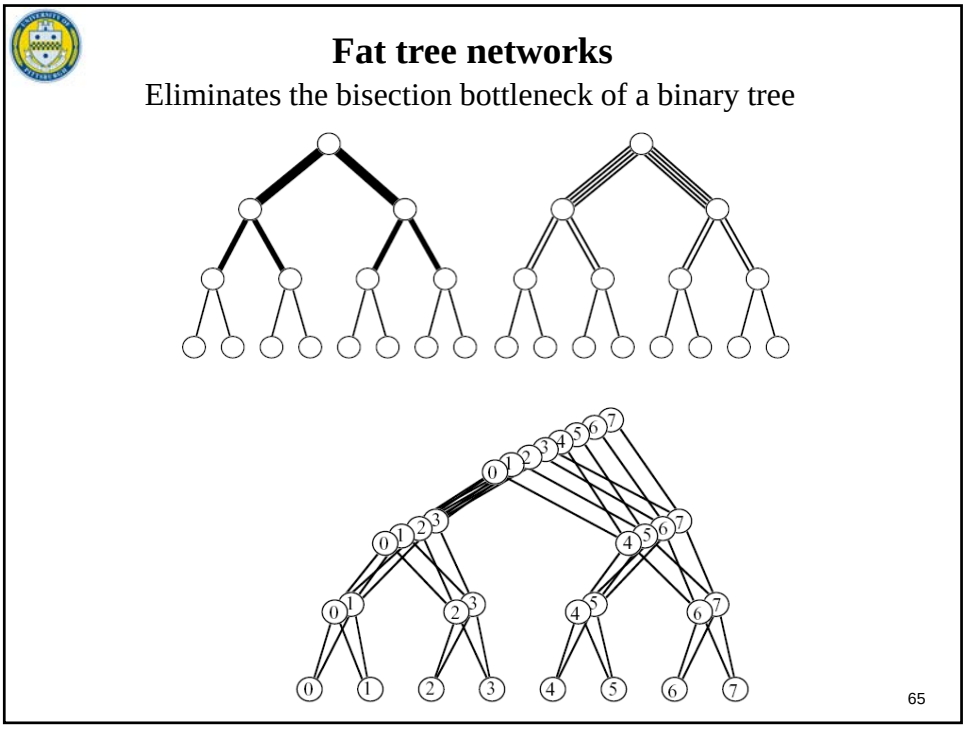
63

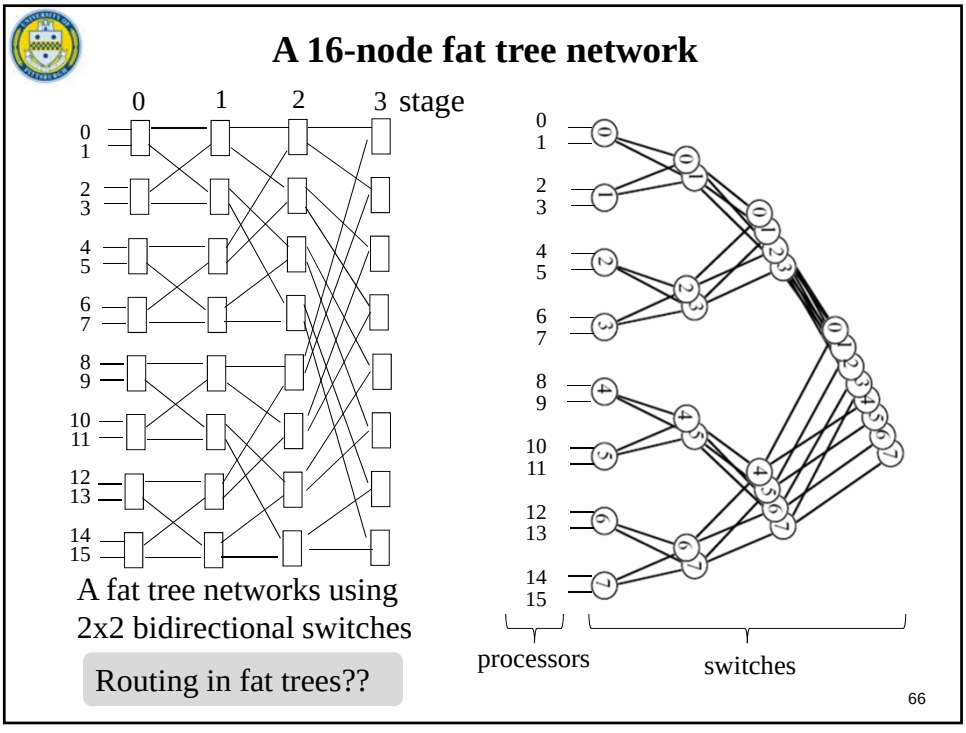
Tree Network topology



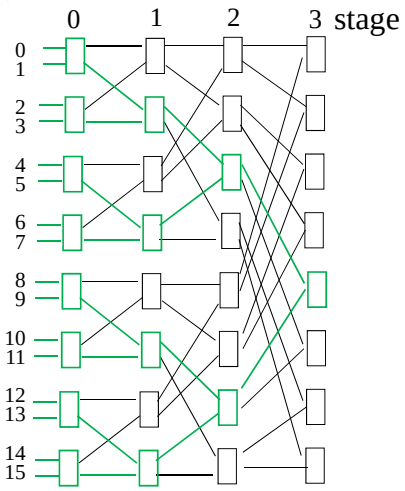
- The route between a source node, $s_{q-1}, \ldots, s_0$, and a destination node , $d_{q-1}, \ldots, d_0$, can be expressed as a sequence of up moves (U) followed by a sequence of right (R) and left (L) moves.
- Example: the route between 001 and 101 is UUURLR
- What is the bisection width of a tree with n leave nodes?

64


Fat tree networks
Eliminates the bisection bottleneck of a binary tree
65

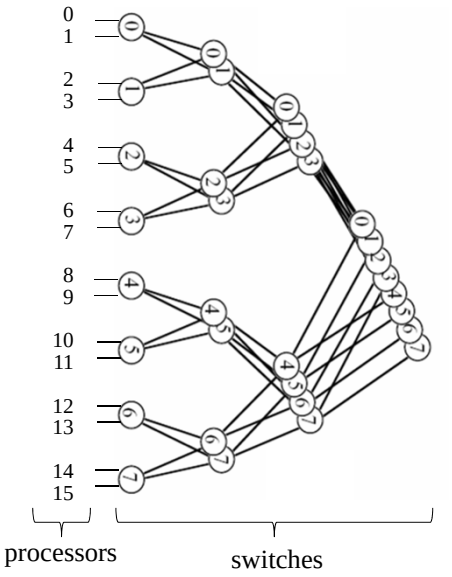

A 16-node fat tree network
0 1 2 3 stage
A fat tree networks using
2x2 bidirectional switches
Routing in fat trees??
processors
switches
66

A 16-node fat tree network



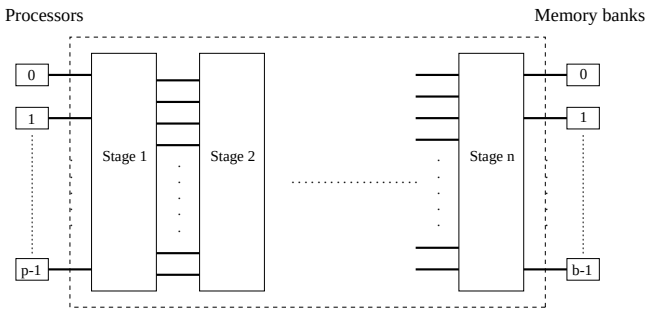
A fat tree networks using 2x2 bidirectional switches

Routing in fat trees??



67

Multistage Networks



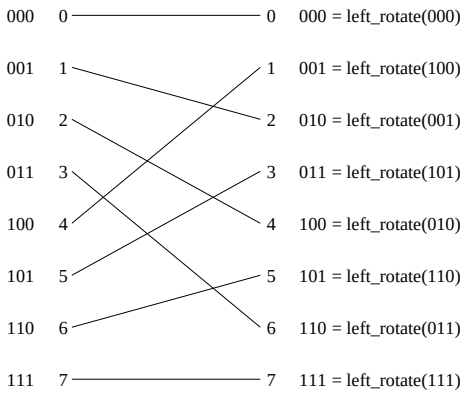
The schematic of a typical multistage interconnection network.

68

Multistage Omega Network

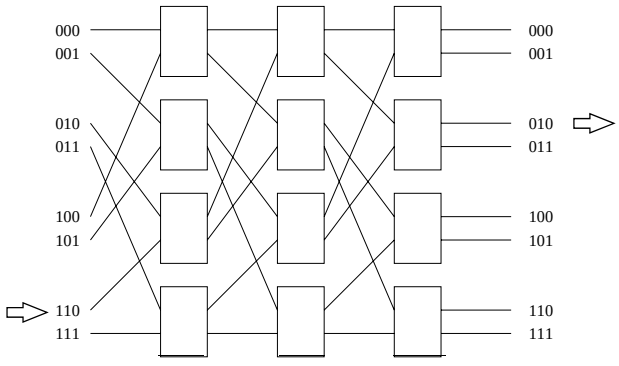
A *nxn* Omega network consists of:
- *log n* stages,
- each stage has *n/2*, 2x2 switches
- Perfect shuffle connection between stages

i connects to *2i* for *i = 0, … , n/2-1*
i connects to *2i+1-n* for *i = p/2, … , n-1*



69

Multistage Omega Network

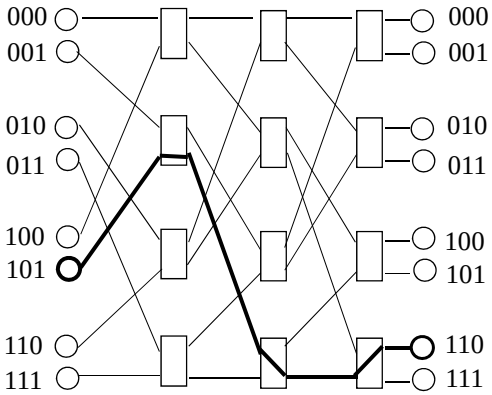


An 8x8 omega network

- cost grows as *n log n*
- Unique route between a source, *s,* and a destination, *d.*

70

Routing in an OMEGA network



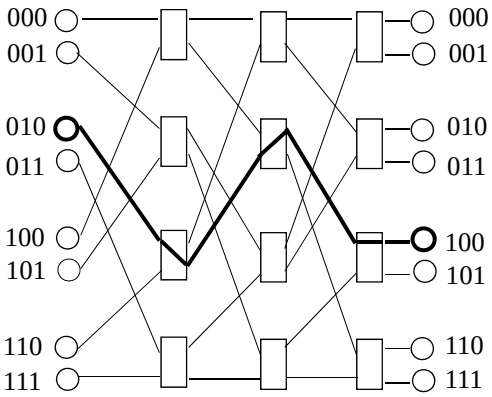
Example: to route from source 101 to destination 110 (101 xor 110 = 011)

101 → 011 → 011 → 110 → 111 → 111 → 110

shuffle, straight; shuffle, exchange cross; shuffle, exchange cross

71

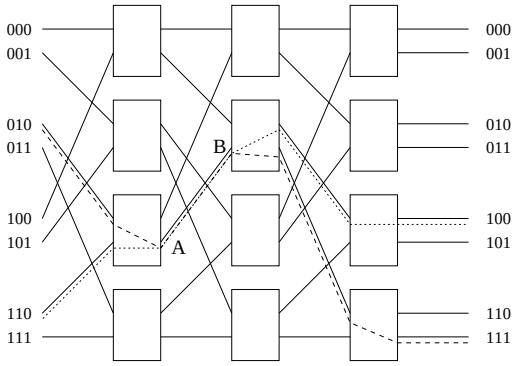
Routing in an OMEGA network



Example: to route from source 010 to destination 100
010 xor 100 = 110 = (cross, cross, straight)
Route: cross at level 0, cross at level 1, straight at level 2

72

The Omega Network is blocking

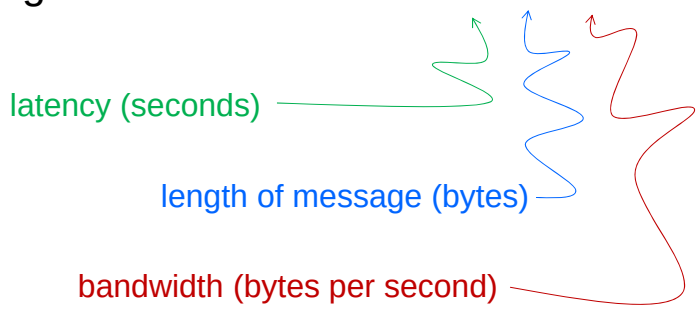


Example: one of the messages (010 to 111 or 110 to 100) is blocked at link AB.

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.

Message transmission time = l + n / b



Example: what is the transmission time for a message of length 256 Bytes if the network delay is 500 nsec and bandwidth is 100 MB/sec?