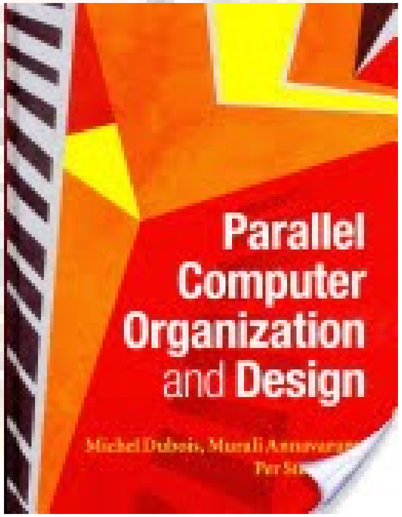
Embedded Computer Architecture
5SAI0

Interconnection Networks

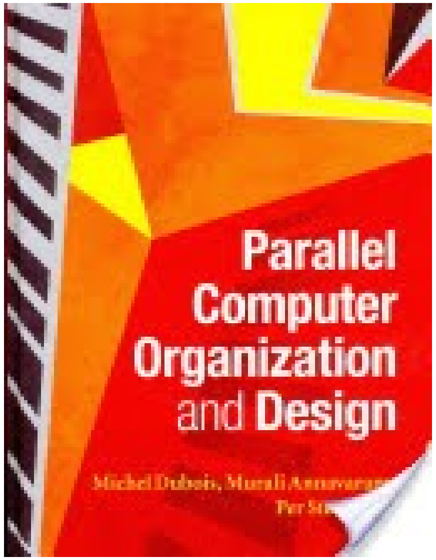
Henk Corporaal
www.ics.ele.tue.nl/~heco/courses/ECA

h.corporaal@tue.nl
TUEindhoven
2016-2017



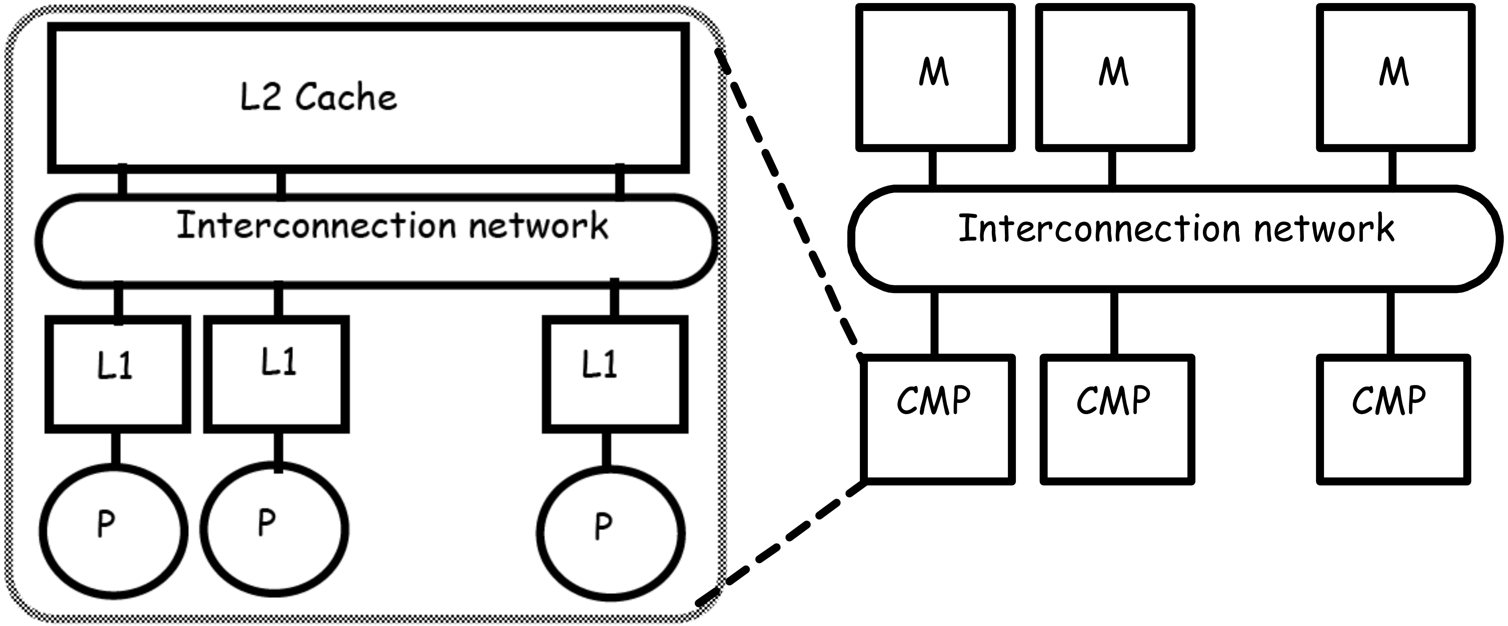
Overview

- Connecting multiple processors or processing elements
- How to connect
  - Topologies
  - Routing
    - Deadlock
  - Switching
  - Performance: Bandwidth and Latency



- Material from book:
  - Chapter 6

Parallel computer systems



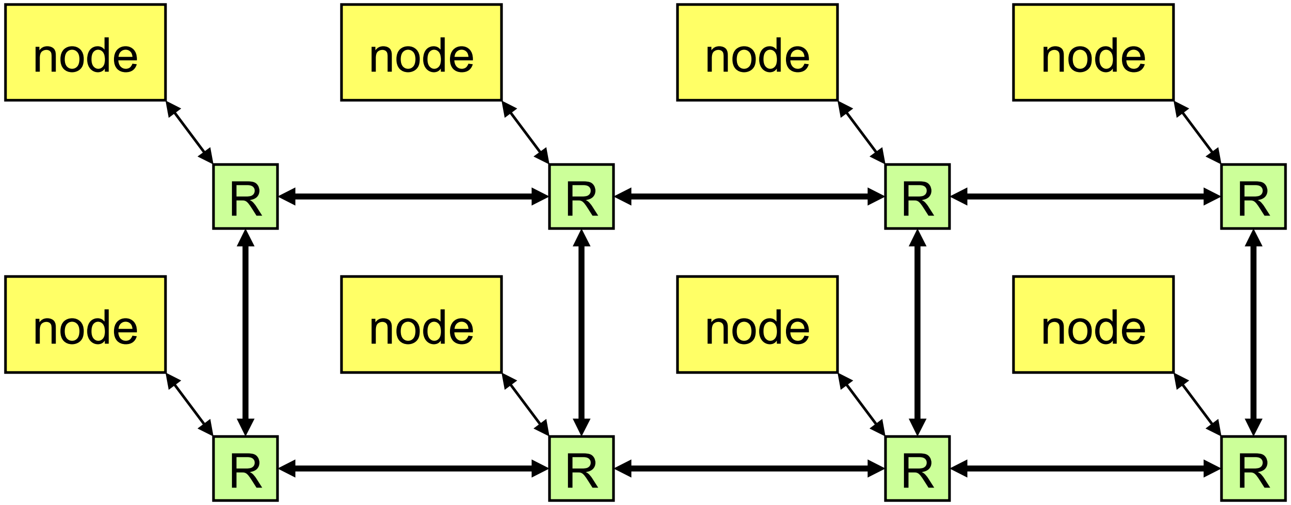
- Interconnect between processor chips (system area network--san)
- Interconnect between cores on each chip (Network On Chip -- NOC)
-
- Others (not covered):
  - WAN (wide-area network)
  - LAN (local area network)

Bus (shared) or Network (switched)

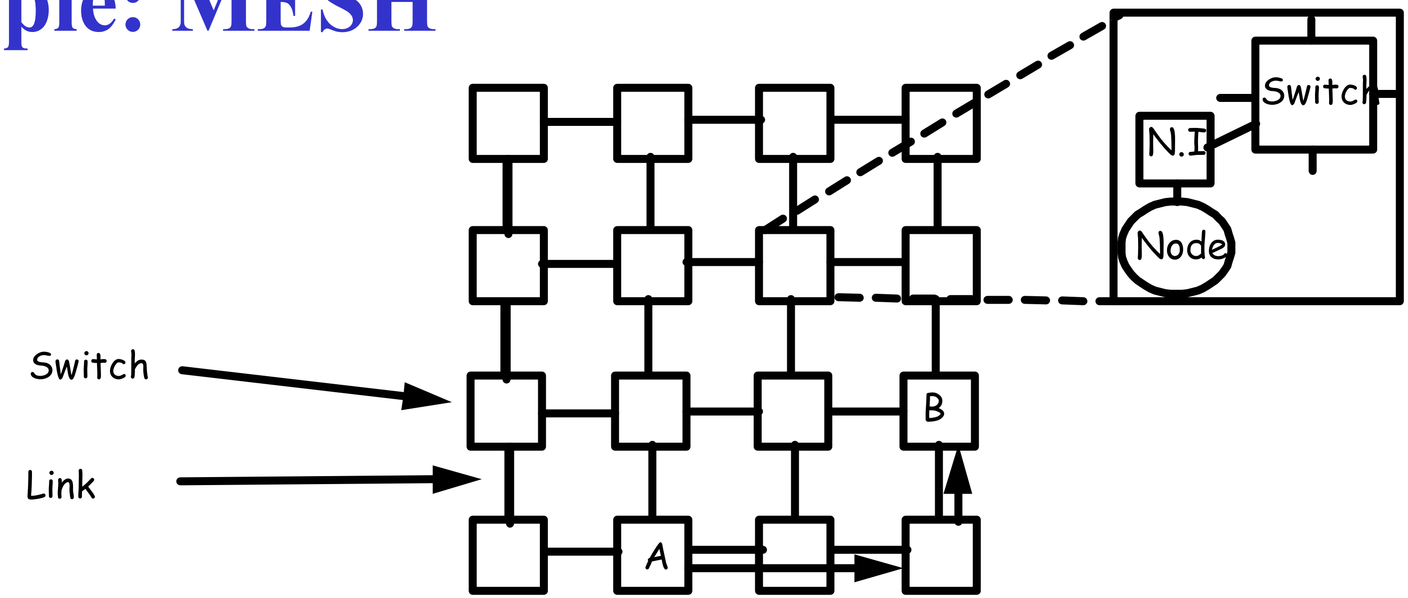
- **Network**:
  - claimed to be more scalable
  - no bus arbitration
  - point-to-point connections

  - but router overhead

Example:
NoC with 2x4 mesh
routing network

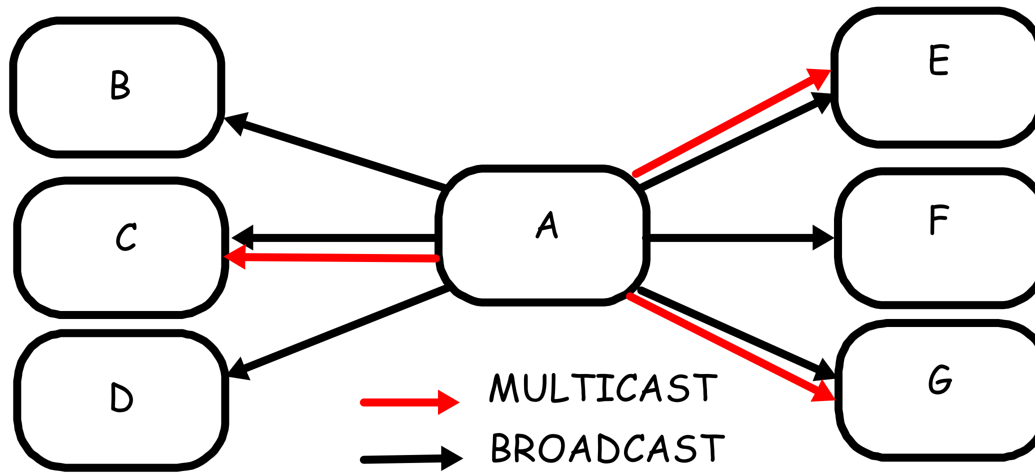


Example: MESH



- Connects nodes: cache and modules, processing elements, processors, …
  - **Nodes** are connected to switches through a network interface (NI)
  - **Switch**: connects input ports to output ports
  - **Link**: wires transferring signals between switches
- Links
  - Width and Clock rate determine Bandwidth
  - Transfer can be synchronous or asynchronous
- From A to B: **hop** from switch to switch
- Decentralized (direct)

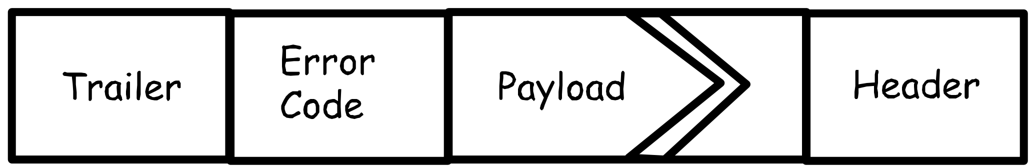
Simple communication model



- Point-to-point message transfer
- Request/reply: request carries ID of sender
- Multicast: one to many
- Broadcast: one to all

Messages and packets

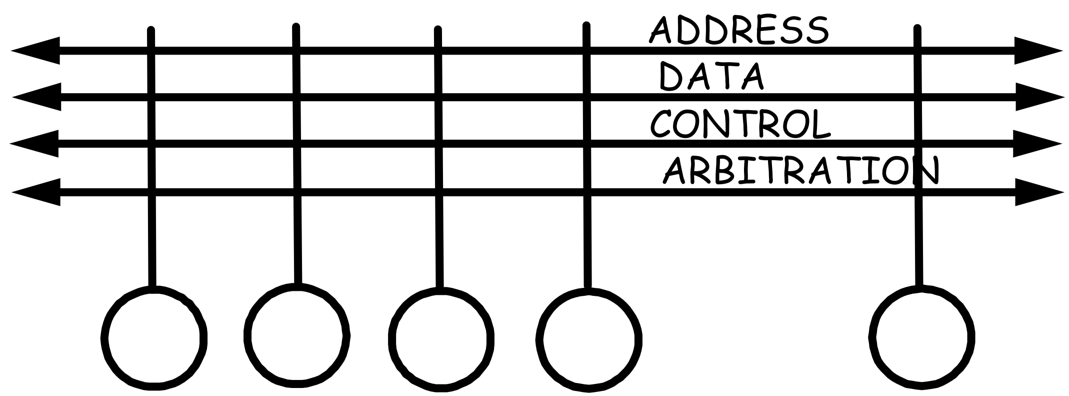
- Messages contain the information transferred
  - Messages are broken down into packets
- Packets are sent one by one



  - Payload: message--not relevant to interconnection
  - Header/trailer: contains information to route packet
  - Error Correction Code: ECC to detect and correct transmission errors

Example: Bus

- Bus = set of parallel wires
  - Broadcast communication.
- Needs arbitration
- Centralized vs distributed arbitration
- Line (wire) multiplexing (e.g. address & data)
- Pipelining
  - For example: arbitration => address => data
- Split-transaction bus vs Circuit-switched bus
- Properties
  - Centralized (indirect)
  - Low cost
  - Shared
  - Low bandwidth



Design Characteristics of a Network

- Topology (how things are connected):
  - Crossbar, ring, 2-D and 3-D meshes or torus, hypercube, tree, butterfly, perfect shuffle, ...
- Routing algorithm (path used):
  - Example in 2D torus: first east-west, then north-south (avoids deadlock)
- Switching strategy:
  - Circuit switching: full path reserved for entire message, like the telephone.
  - Packet switching: message broken into separately-routed packets, like the post office.
- Flow control and buffering (what if there is congestion):
  - Stall, store data temporarily in buffers
  - re-route data to other nodes
  - tell source node to temporarily halt, discard, etc.
- QoS guarantees
- Error handling
- etc, etc.

Switch / Network Topology

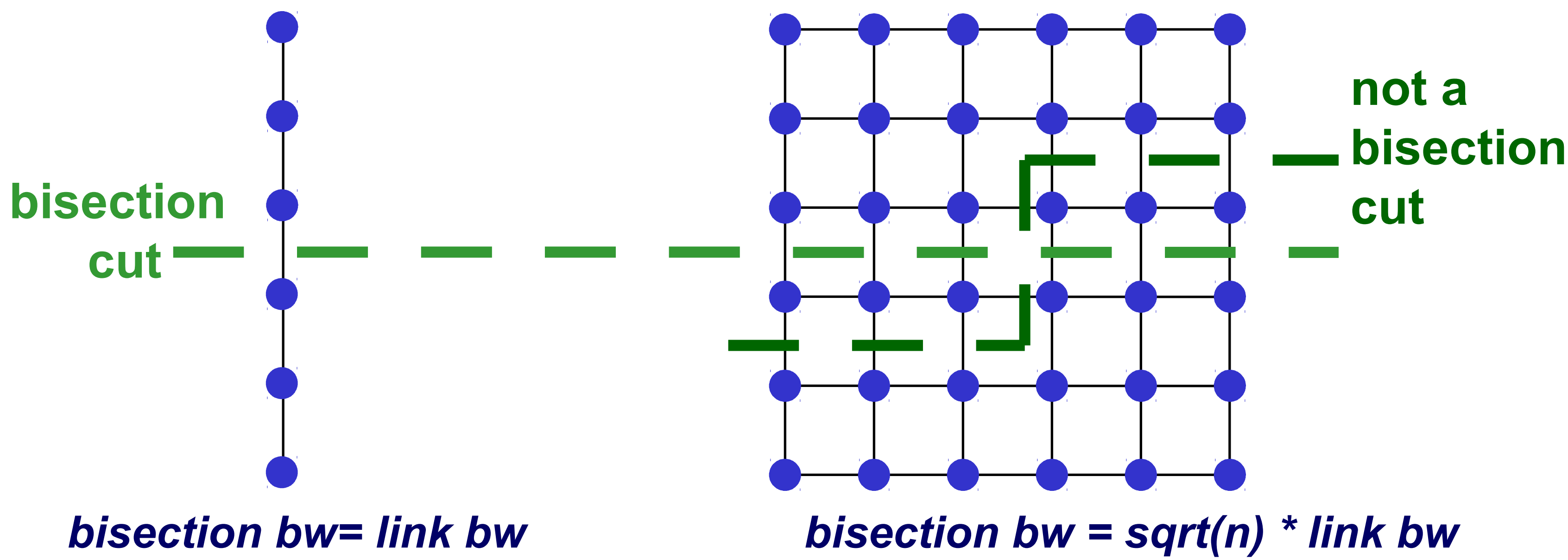
Topology determines:

- **Degree**: number of links from a node
- **Diameter**: max number of links crossed between nodes
- **Average distance**: number of links to random destination
- **Bisection**: minimum number of links that separate the network into two halves

- **Bisection bandwidth** = link bandwidth * bisection

Bisection Bandwidth

- Bisection bandwidth: bandwidth across **smallest cut** that divides network into two equal halves
- Bandwidth across "narrowest" part of the network



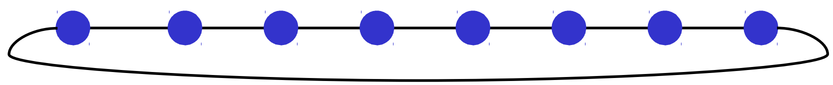
- Bisection bandwidth is important for algorithms in which all processors need to communicate with all others

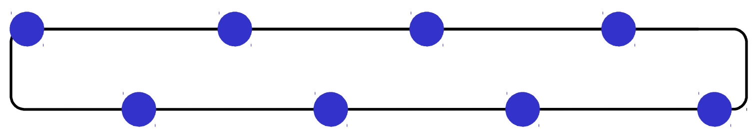
Linear and Ring Topologies

- Linear array

  - Diameter = n-1; average distance ~n/3
  - Bisection bandwidth = 1 (in units of link bandwidth)

- Torus or Ring



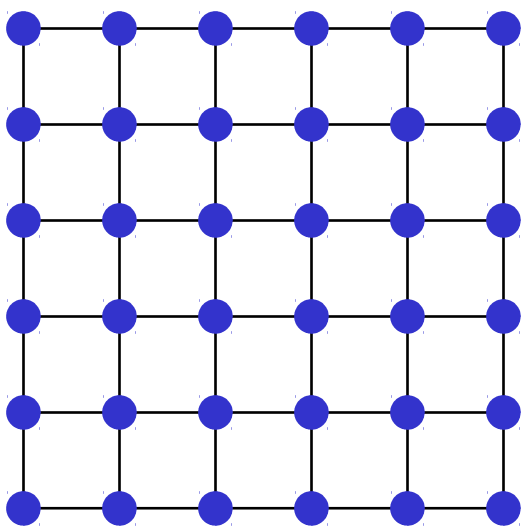


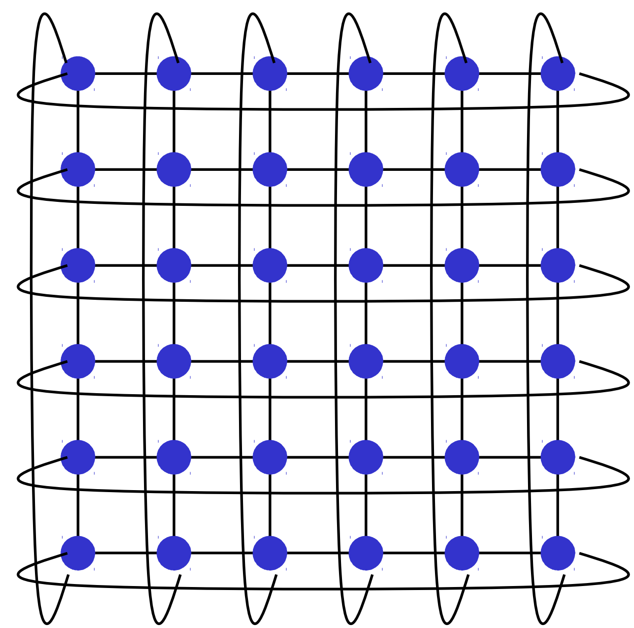
  - Diameter = n/2; average distance ~ n/4
  - Bisection bandwidth = 2
  - Natural for algorithms that work with 1D arrays

Meshes and Tori

- Two dimensional mesh
- Diameter = 2 * (sqrt(n) – 1)
- Bisection bandwidth = sqrt(n)

- Two dimensional torus
- Diameter = sqrt(n)
- Bisection bandwidth = 2* sqrt(n)

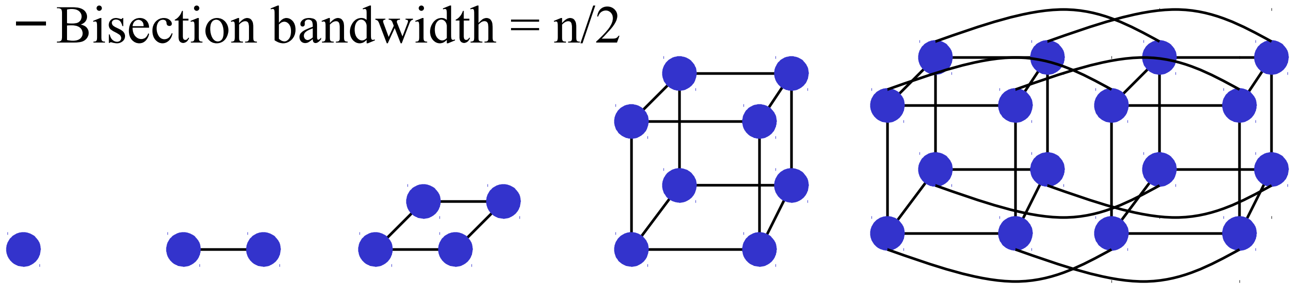




- Generalizes to higher dimensions
- Natural for algorithms that work with 2D and/or 3D arrays

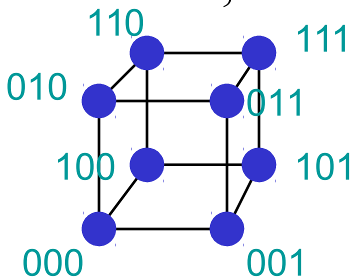
Hypercubes

- Number of nodes n = 2d   for dimension d
  - Diameter = d
  - Bisection bandwidth = n/2



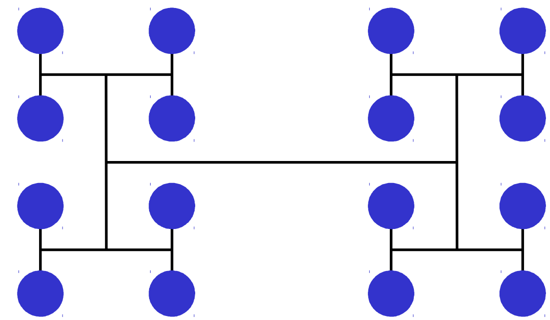
0d 1d 2d 3d 4d

- Popular in early machines (Intel iPSC, NCUBE, CM)
  - Lots of clever algorithms
  - Extension: k-ary n-cubes (k nodes per dimension, so k^n nodes)
- Greycode addressing:
  - Each node connected to others with 1 bit different

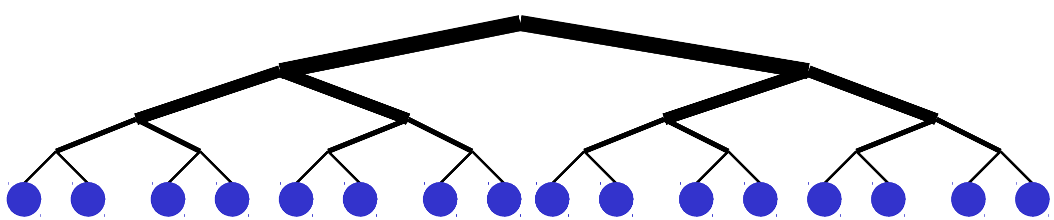


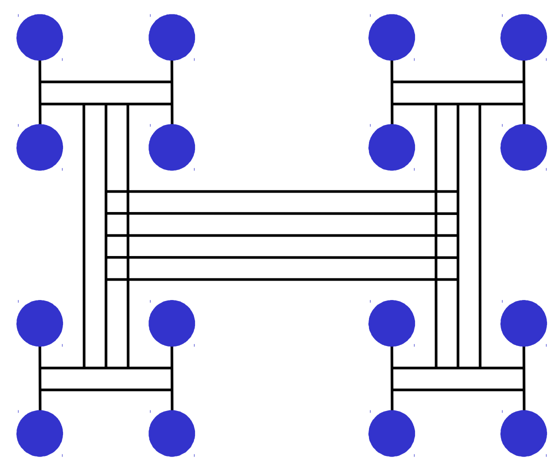
Trees

- Diameter = log n.
- Bisection bandwidth = 1
- Easy layout as planar graph
- Many tree algorithms (e.g., summation)

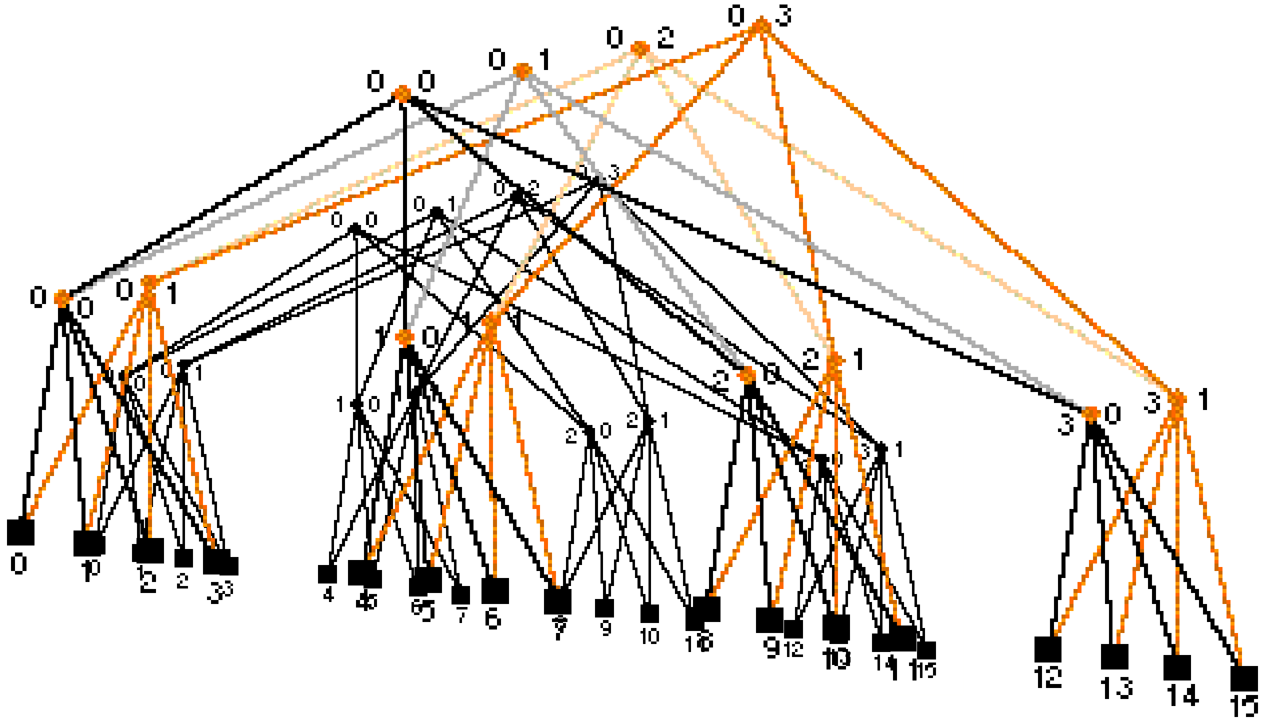


- **Fat trees** avoid bisection bandwidth problem:
  - More (or wider) links near top
  - Example: Thinking Machines CM-5



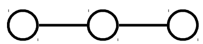


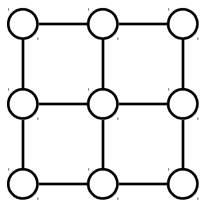
Fat Tree example

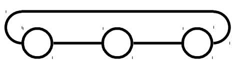


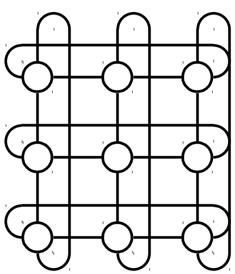
- A multistage fat tree (CM-5) avoids congestion at the root node
- Randomly assign packets to different paths on way up to spread the load
- Increase degree near root, decrease congestion

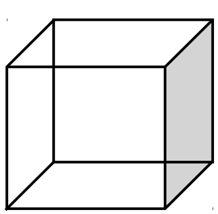
Common Topologies





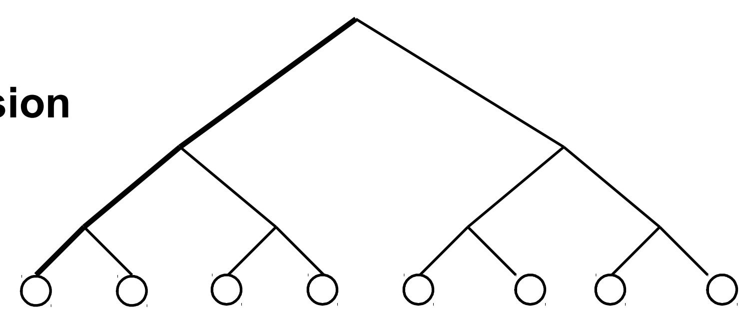




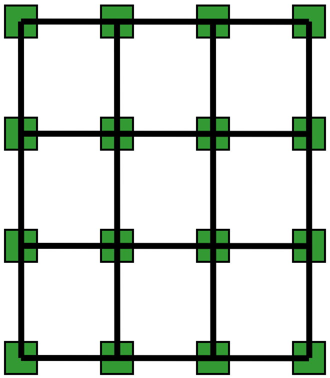


Type	Degree	Diameter	Ave Dist	Bisection
1D mesh	2	N-1	N/3	1
2D mesh	4	$2(N^{1/2} - 1)$	$2N^{1/2} / 3$	$N^{1/2}$
3D mesh	6	$3(N^{1/3} - 1)$	$3N^{1/3} / 3$	$N^{2/3}$
nD mesh	2n	$n(N^{1/n} - 1)$	$nN^{1/n} / 3$	$N^{(n-1) / n}$
Ring	2	N/2	N/4	2
2D torus	4	$N^{1/2}$	$N^{1/2} / 2$	$2N^{1/2}$
Hypercube	$\mathrm{Log}_2 N$	$n = \mathrm{Log}_2 N$	n/2	N/2
2D Tree	3	$2\mathrm{Log}_2 N$	$\sim 2\mathrm{Log}_2 N$	1
Crossbar	N-1	1	1	$N^2/2$

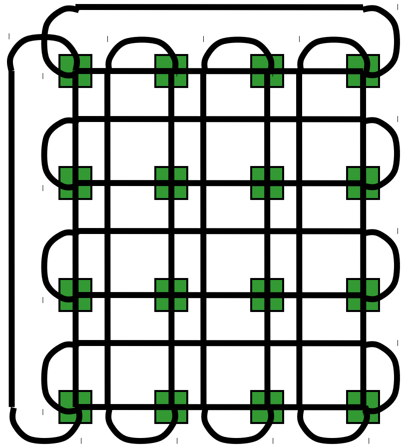
N = number of nodes, n = dimension



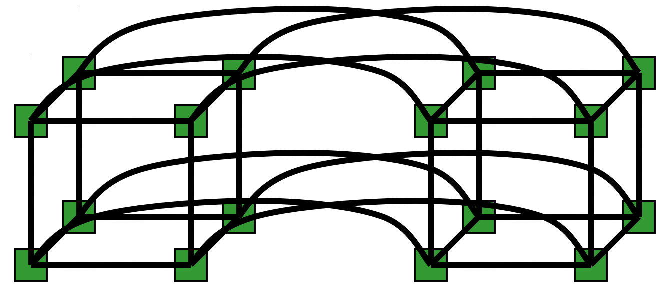
More examples



2D-Grid/Mesh



2D-Torus



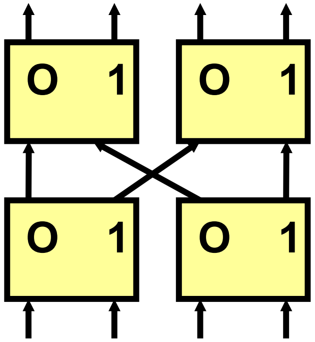
Hypercube

Assume 64 nodes:

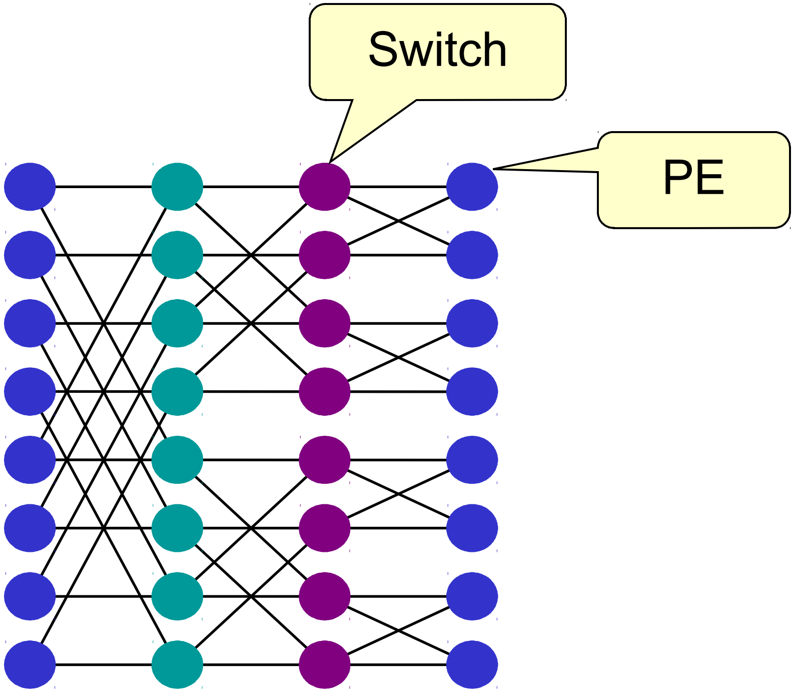
Criteria	Bus	Ring	2D-Mesh	2D-torus	6-cube	Fully connected
Performance Bisection bandwidth	1	2	8	16	32	1024
Cost Ports/switch Total #links	 1	 3 128	 5 176	 5 192	 7 256	 64 2080

Butterflies with $n = (k-1)2^k$ switches

- Connecting $\mathbf{2^k}$ processors, with Bisection bandwidth = $2*2^k$
- Cost: lots of wires
- $^2\log(k)$ hop-distance for **all** connections, however blocking possible
- Used in BBN Butterfly
- Natural for FFT



Butterfly switch



Multistage butterfly network: k=3

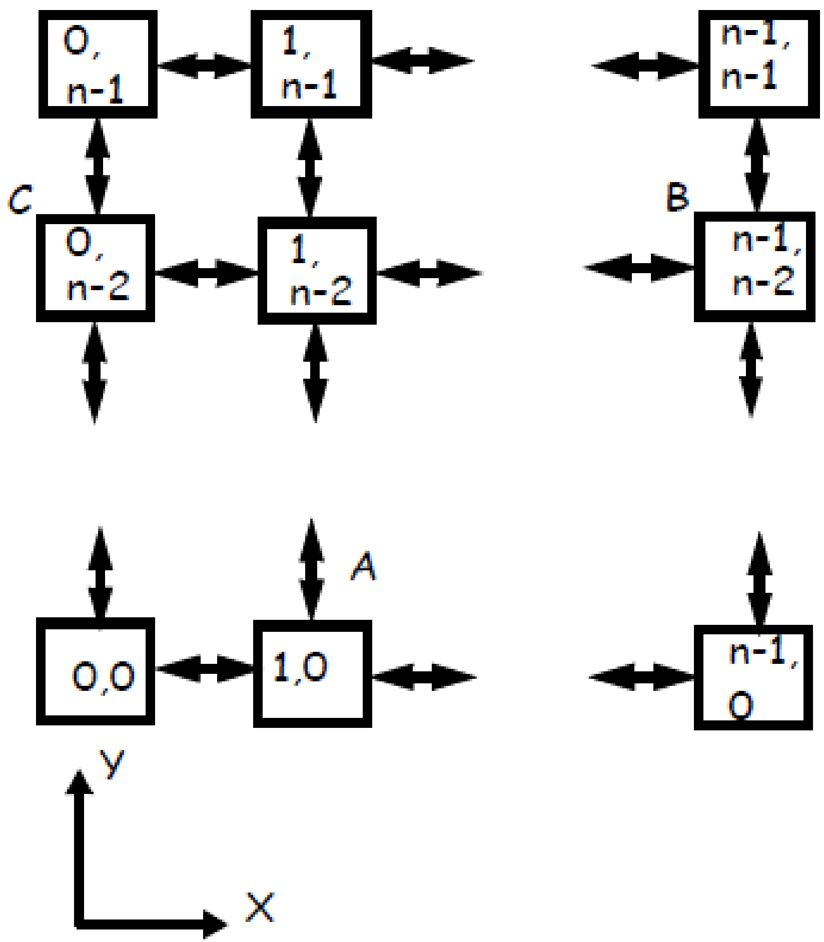
Real machines use all kinds of topologies

older newer

Red Storm (Opteron + Cray network, future)	3D Mesh
Blue Gene/L	3D Torus
SGI Altix	Fat tree
Cray X1	4D Hypercube*
Myricom (Millennium)	Arbitrary
Quadrics (in HP Alpha server clusters)	Fat tree
IBM SP	Fat tree (approx)
SGI Origin	Hypercube
Intel Paragon (old)	2D Mesh
BBN Butterfly (really old)	Butterfly

Routing algorithms

- Dimension-order routing (deterministic) = route 1-dimension at a time



NUMBER NODES SO THAT LOWER LEFT CORNER IS (0,0) AND UPPER RIGHT CORNER IS (n-1,n-1)

USE RELATIVE ADDRESS:

- $(X,Y)=(X_B-X_A,Y_B-Y_A)$
- ROUTE FIRST HORIZONTALLY
  - DECREMENT X
- WHEN X=0, ROUTE VERTICALLY
  - DECREMENT Y
  - UNTIL Y=0

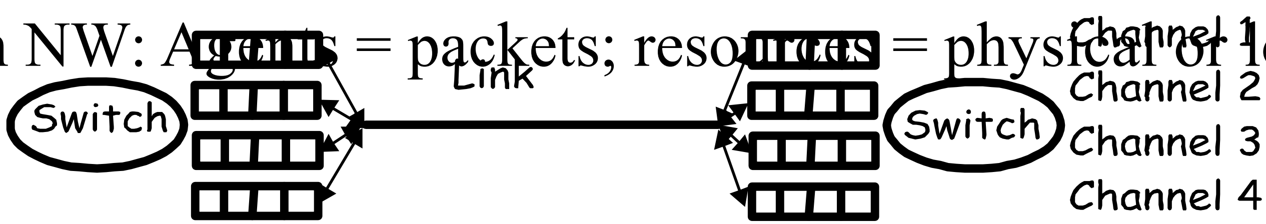
EXAMPLE: MOVE PACKET FROM A TO B

- RELATIVE ADDRESS IS (X,Y)=(n-2,n-2)
- FIRST MOVE PACKET HORIZONTALLY
  - DECREMENT X BY 1 (RIGHT MOVE)
- WHEN X=0, MOVE PACKET VERTICALLY
  - DECREMENT Y BY 1 (UP MOVE)

TRY B TO C

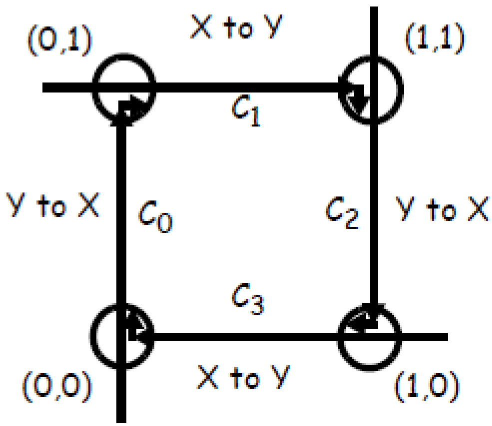
Deadlock

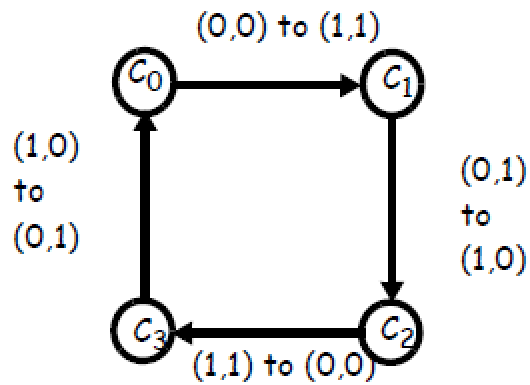
- 4 necessary conditions for deadlock, given a set of agents accessing a set of shared resources:
  - **Mutual exclusion**
    - Only one agent can access the resource at a time
  - **No preemption**
    - No mechanism can force agent to relinquish acquired resource
  - **Hold and wait**
    - Agent holds on its acquired resources while waiting for others
  - **Circular wait**
    - A set of agents wait on each other to acquire each others’ resources
      => no one can make any progress
- shared resources can be SW or HW: critical sections, disk, printer, ..
- In NW: Agents = packets; resources = physical or logical channels



Deadlock avoidance

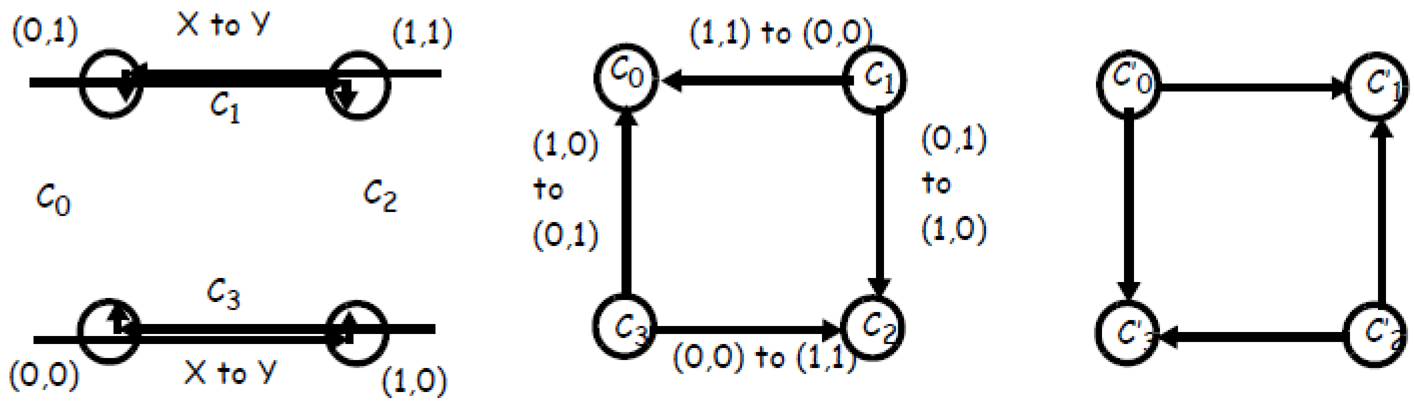
- Assume Mesh or Tori
- Assume that packets are free to follow any route





- In this example each node is trying to send a packet to the diagonally opposite node at the same time, e.g. (0,0) to (1,1)
- To avoid link conflicts, (1,0) uses c3 then c0, and (0,0) uses c0 then c1, etc...
- The resource acquisition graph (or channel-dependency graph) on the right shows circular wait => DEADLOCK Possible

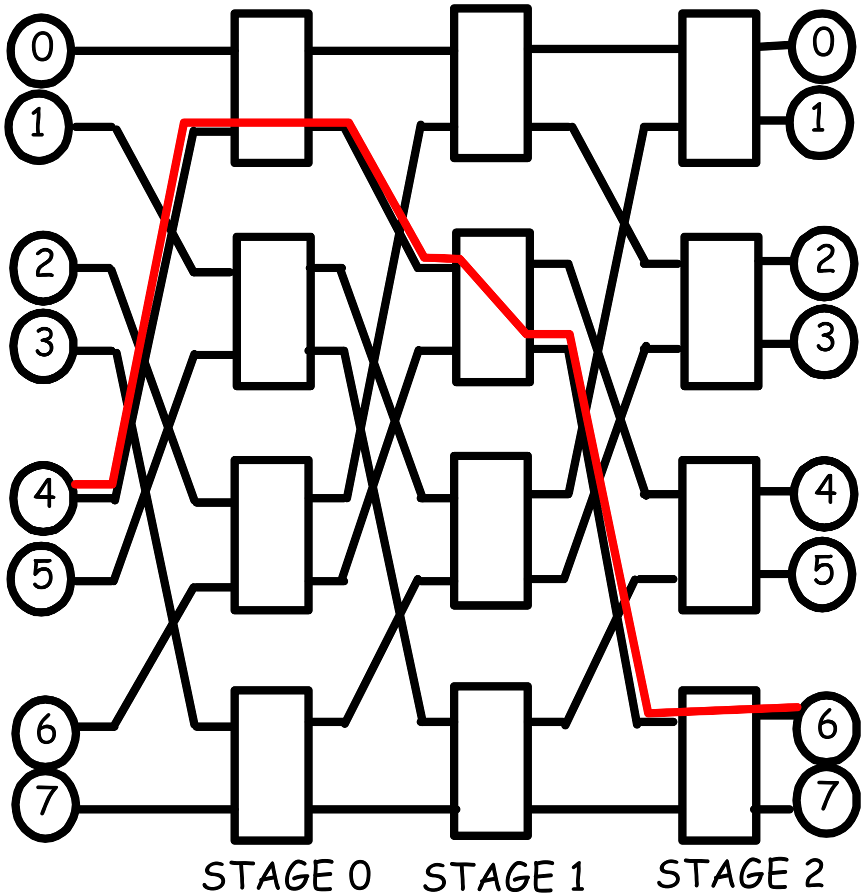
Deadlock avoidance



- Enforce dimension-order routing (xy-routing)
  - Packet moves first horizontally, then vertically
  - No cycle!!!
- Problem: contention for channels
  - If (0,0) wants to send a packet to (1,1), it must first use c3
  - If c3 is occupied, could take alternate route c0 => c1
- To avoid deadlocks, use virtual channels
  - Alternate set of channels in which yx routing is enforced e.g., c'1
  - If c3 is occupied, the packet can safely route through c'0 and c'1

Routing in butterflies: Omega NW

- Use source and/or destination addresses



USE THE DESTINATION ADDRESS IN THIS CASE, 3 BITS <d2,d1,d0>

USE ith BIT OF THE DESTINATION (di) TO SELECT UPPER OR LOWER OUTPUT PORT FOR STAGE n-1-i

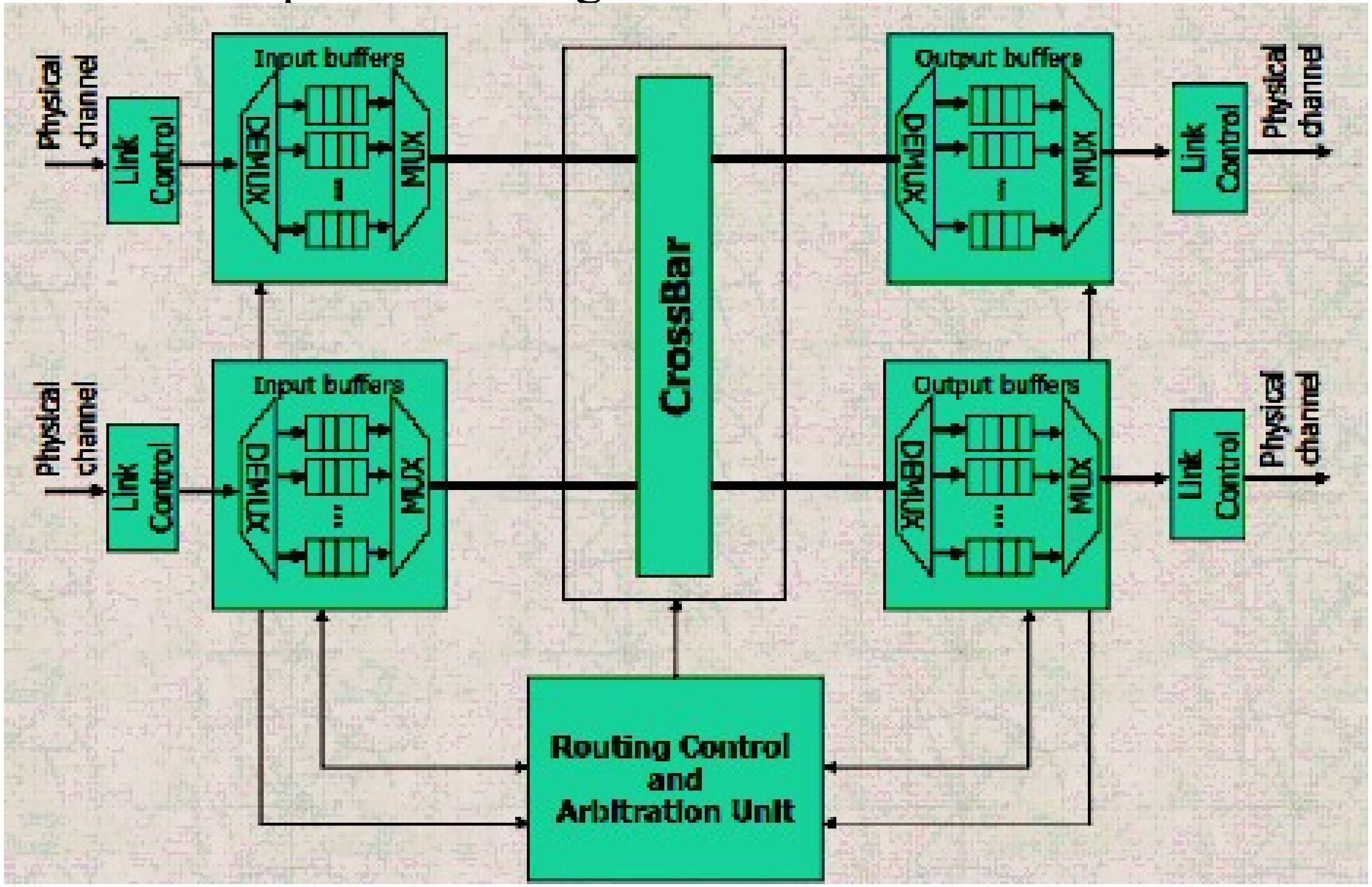
- 0 => UP
- 1 => DOWN

EXAMPLE: ROUTE FROM 4 TO 6

- DESTINATION ADDRESS IS 110
  - DOWN IN STAGE 0
  - DOWN IN STAGE 1
  - UP IN STAGE 2

Switch micro-architecture

- Physical channel = link
- Virtual channel = buffers + link
  - link is time-multiplexed among flits



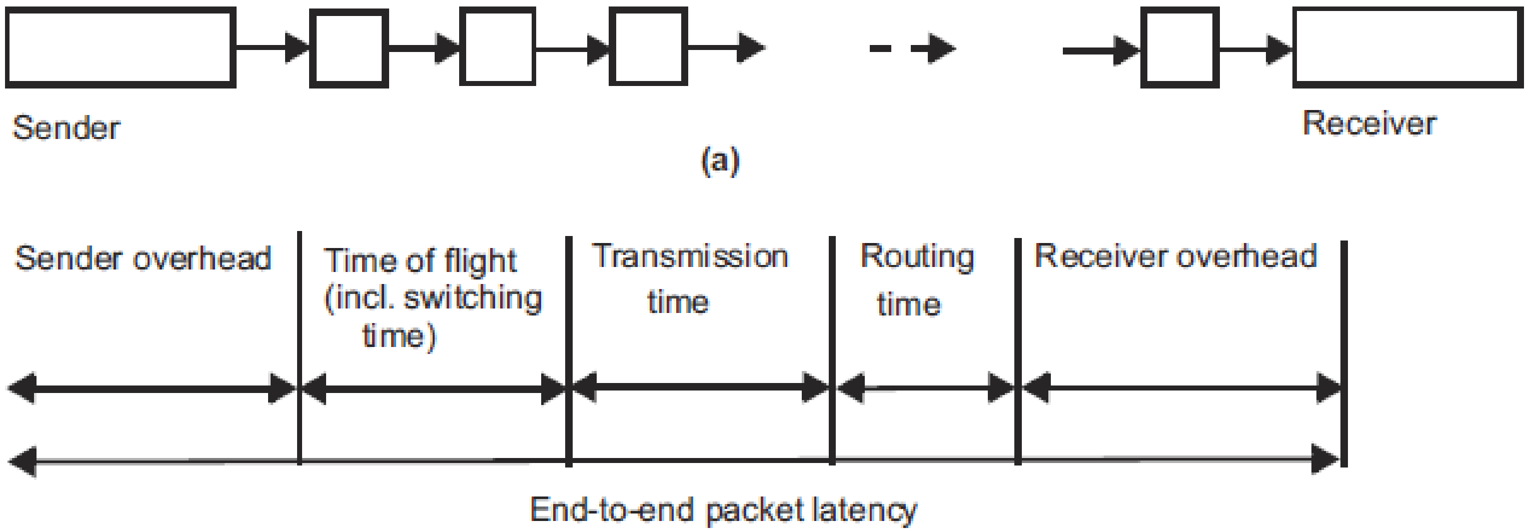
Switching strategy

- Defines how connections are established in the network
- **Circuit switching** = Establish a connection for the duration of the network service
  - Example: circuit switching in mesh
    - Establish path in network
    - Transmit packet
    - Release path
    - Low latency; high bandwidth
    - Good when packets are transmitted continuously between two nodes
- **Packet switching** = Multiplex several services by sending packets with addresses
  - Example: remote memory access on a bus
    - Send a request packet to remote node
    - Release bus while memory access takes place
    - Remote node sends reply packet to requester
    - In between send and reply, other transfers are supported
  - Example: remote memory access on a mesh

2 Packet switching strategies

- **store-and-forward =** packet switching, packets move from node to node and are stored in buffers in each node
- **cut-through** = packets move through nodes in pipeline fashion, so that the entire packet moves through several nodes at one time
  - Two implementations of cut-through:
    - **Virtual cut-through switching**:
      - The entire packet is buffered in the node when there is a transmission conflict
      - When traffic is congested and conflicts are high, virtual cut through behaves like store-and-forward
    - **Wormhole switching**:
      - Each node has enough buffering for a flit (flow control unit)
      - A flit is made of consecutive phits (physical transfer unit), which basically is the width of a link (= number of bits transferred per clock)
      - A flit is a fraction of the packet, so the packet can be stored in

Latency models



- **Sender overhead**: creating the envelope and moving packet to NI
- **Time of flight**: time to send a bit from source to destination when the route is established and without conflicts. (Includes switching time.)
- **Transmission time**: time to transfer a packet from source to destination, *once the first bit has arrived at destination*
  - phit: number of bits transferred on a link per cycle
  - Basically: packet size/phit size
  - flit: flow control unit

Latency models

- **Routing time**: time to set up switches
- **Switching time**: depends on switching strategy (store-and-forward vs cut-through vs circuit-switched). Affects time of flight and included in that.
- **Receiver overhead**: time to strip out envelope and move packet in

Measures of latency

- Routing distance: Number of links traversed by a packet
- Average routing distance: average over all pairs of nodes
- Network diameter: longest routing distance over all pairs of nodes

- Packets of a message can be pipelined
  - Transfer pipeline has 3 stages
    - Sender overhead-->transmission -->receiver overhead
  - Total message time = time for the first packet + (n-1)/pipeline throughput

End-to-end message latency =

sender overhead + time of flight + transmission time +
routing time +
*(n-1) * MAX(sender ov, transmission time, receiver overhead)*

- summarize what you learned

EXTRA SLIDES

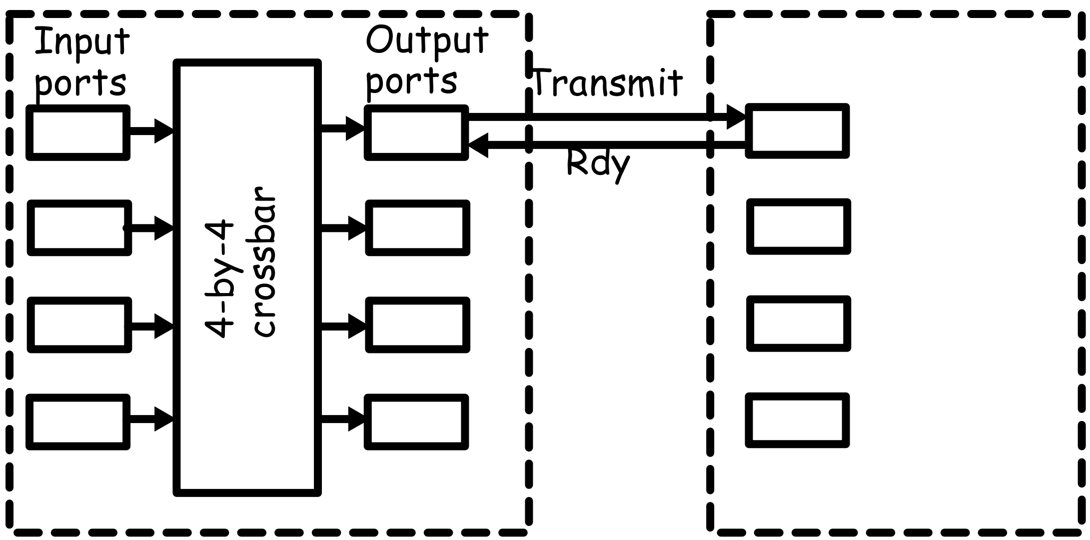
Comparison between topologies

Interconnection network	Switch degree	Network diameter	Bisection width	Network size
Crossbar switch	N	1	N	N
Butterfly (built from k-by-k switches)	k	$\log_k N$	N/2	N
k-ary tree	k+1	$2\log_k N$	1	N
Linear array	2	N-1	1	N
Ring	2	N/2	2	N
n-by-n mesh	4	2(n-1)	n	$N=n^2$
n-by-n torus	4	n	2n	$N=n^2$
k-dimensional hypercube	k	k	2^{k-1}	$N=2^k$
k-ary n-cube	2k	nk/2	$2k^{n-1}$	$N=n^k$

- Switch degree: # of ports for each switch (switch complexity)
- Network diameter: worst-case routing distance between any two nodes
- Bisection width: # of links in bisection (worst-case bandwidth)
- Network size: # of nodes

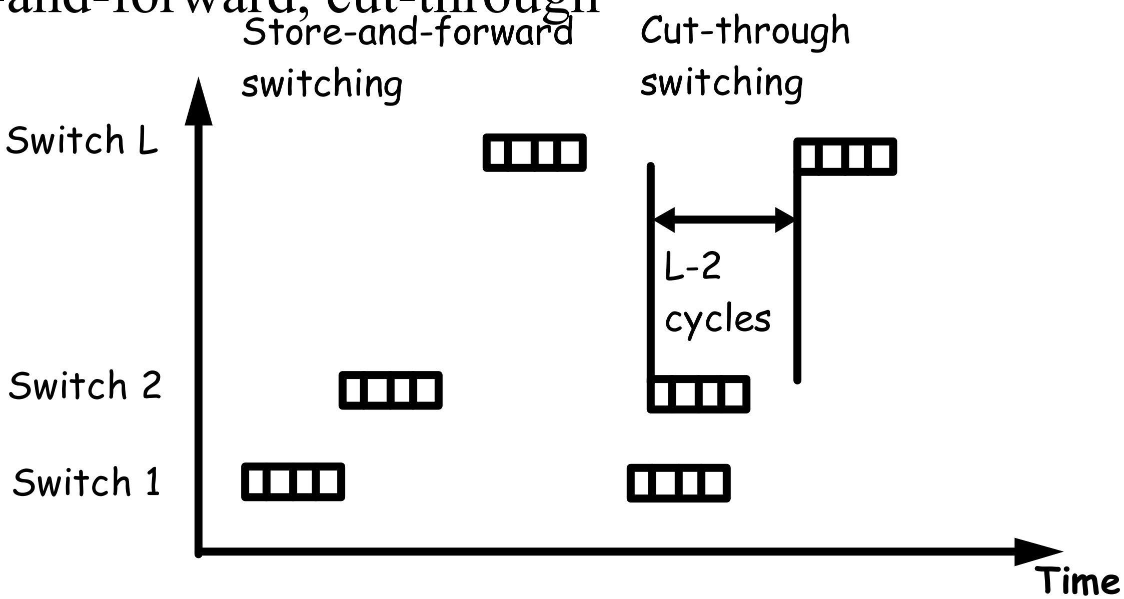
Flow control

- Refers to mechanisms to handle conflicts in switch-based networks
- Buffers at input and output ports
    - In virtual cut-through: buffer for entire packet
    - In wormhole: buffer for integral number of flits
- Link-level flow control
    - Handshake signal
        - Rdy indicates whether flits can be transmitted to the destination
        - Buffering for cut-through (whole packet) vs wormhole (a few flits



Switching strategies

- Circuit switching:
  - Route is set up first
  - Routing time = l x r + time of flight
    - R to set each switch, l number of switches, and tof to inform the node back
- Packet switching
  - Route is set up as the packet moves from switch to switch
  - Store-and-forward, cut-through



Switching strategies

Packet latency = sender ov + tof (incl.Switching time) + transmission time + routing time + receiver ov

R: routing time per switch; n: # of phits; l: # of switches; tof: time of flight

- Circuit switching
  - Packet latency = sender ov + 2xtof + n + lxr + receiver ov
  - Tof = l because there are l switches and nb of phits to switch is one
- Store-and-forward
  - Packet latency = sender ov + tof + n + lxr + receiver ov
  - Tof = lxn because switching involves a whole packet (n phits)
- Cut-through switching
  - Packet latency = sender ov + tof + n + lxr + receiver ov
  - Tof = l as in circuit switching
- Virtual cut-through switching
  - Similar to circuit switching but better bw
  - Note that when traffic is congested, cut-through = store-and-forward
- Wormhole switching
  - Handles conflicts differently
  - Switch port has at least enough buffering for a flit
  - Blocked packets simply stay in the flit buffers provided in their path
  - Packet flits hold circuits in multiple switches

Bandwidth models

- Bottlenecks increase latency
  - Transfers are pipelined
  - Effective bandwidth = packet_size/max(sender ov, receiver ov, transmission time)

- Network contention affects latency and effective bandwidth (not counted in above formula)

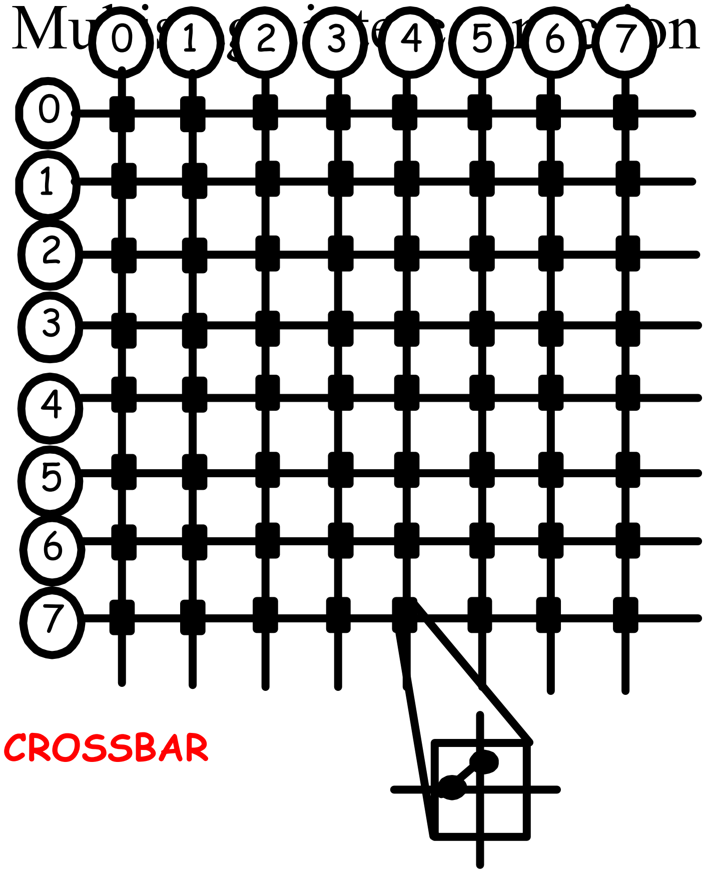
- Bisection width
  - Network is seen as a graph
  - Vertices are switches and edges are links
  - Bisection is a cut through a minimum set of edges such that the cut divides the network graph into two isomorphic --i.E., Same-- subgraphs
    - Example: mesh
  - Measures bandwidth when all nodes in one subgraph communicate only with nodes in the other subgraph

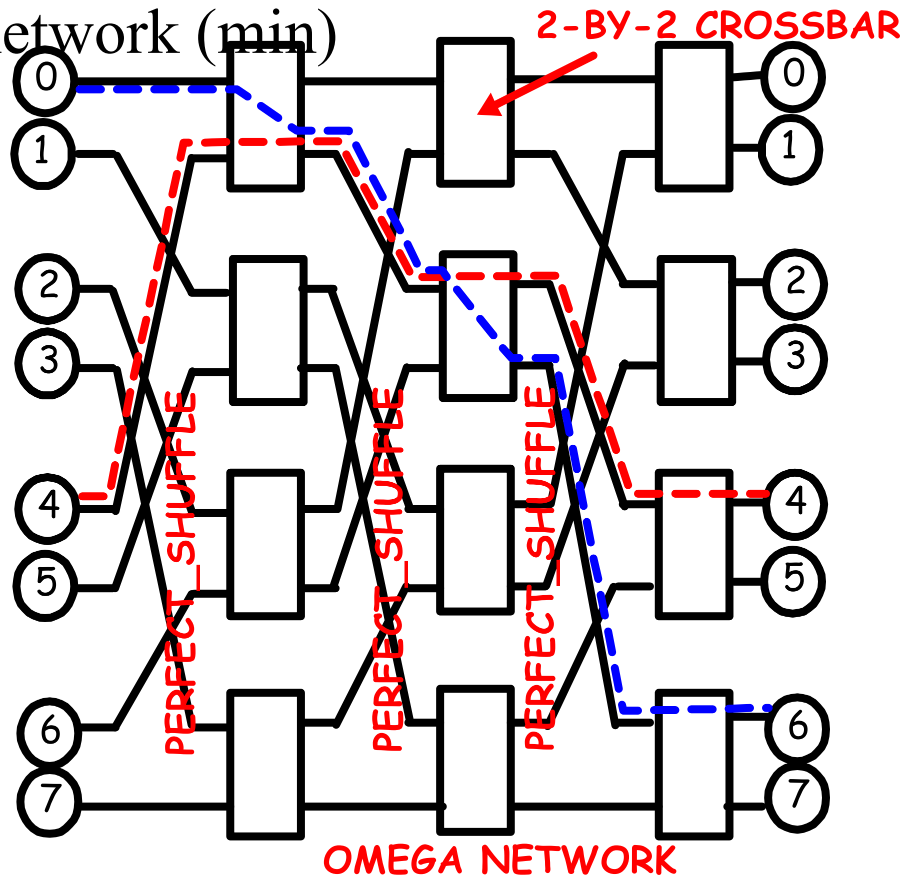
- Aggregate bandwidth
  - Bandwidth across all links divided by the number of nodes

Topologies

Indirect networks: in is centralized

- Bus
- Crossbar switch
- Multistage interconnection network (min)

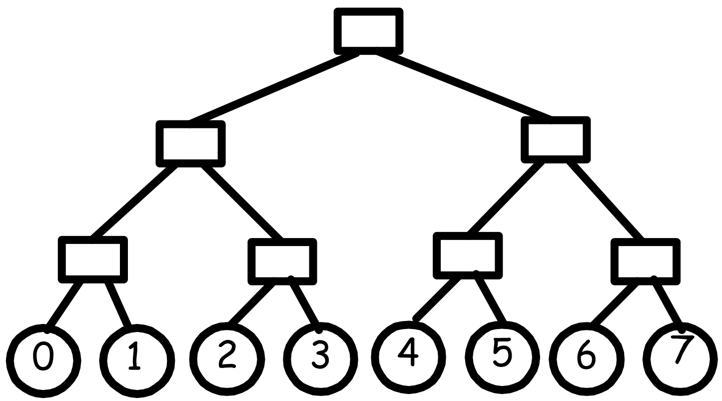




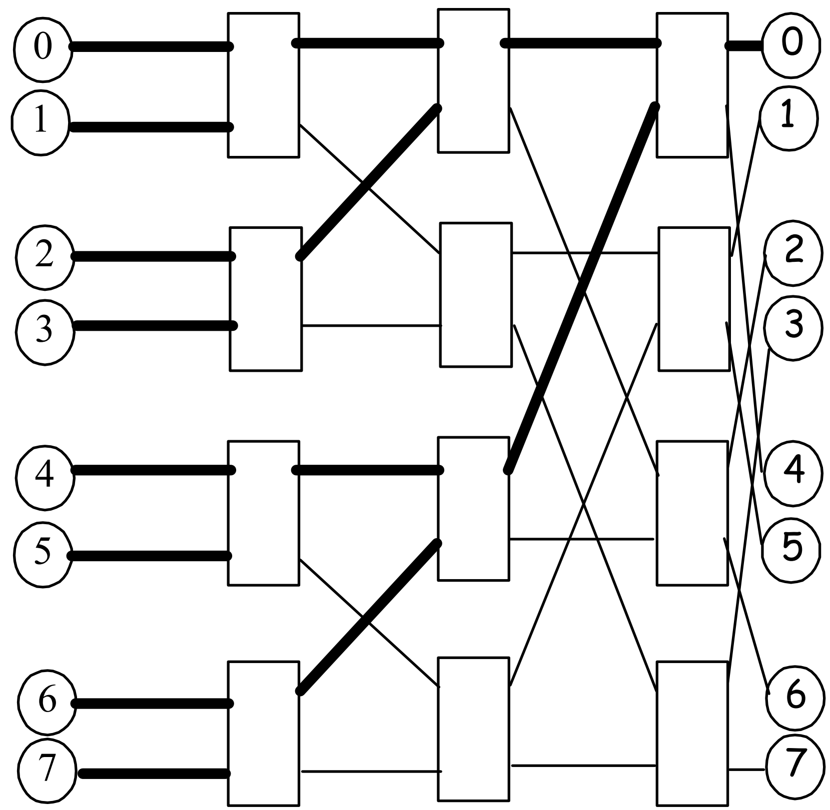
Topologies

Indirect networks: in is centralized

- Tree
- Butterfly



BINARY TREE



BUTTERFLY: EMBEDDED TREES

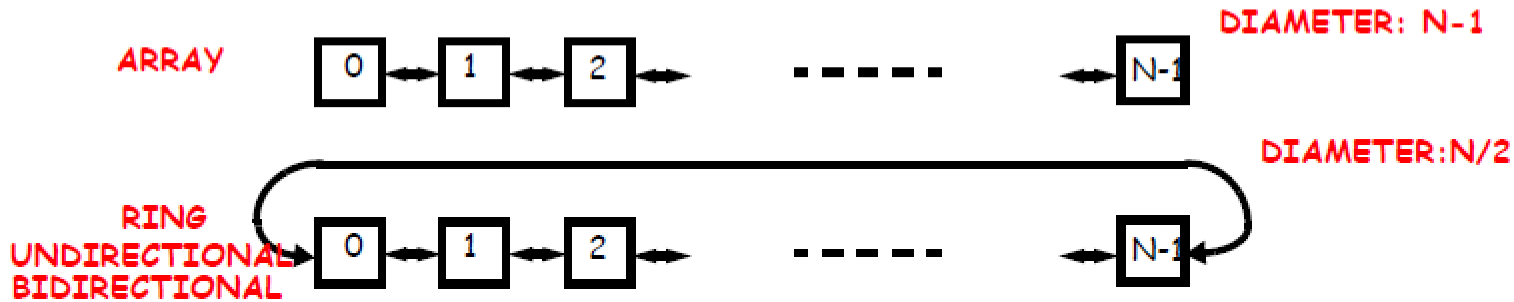
Best to connect different types of nodes

Topologies

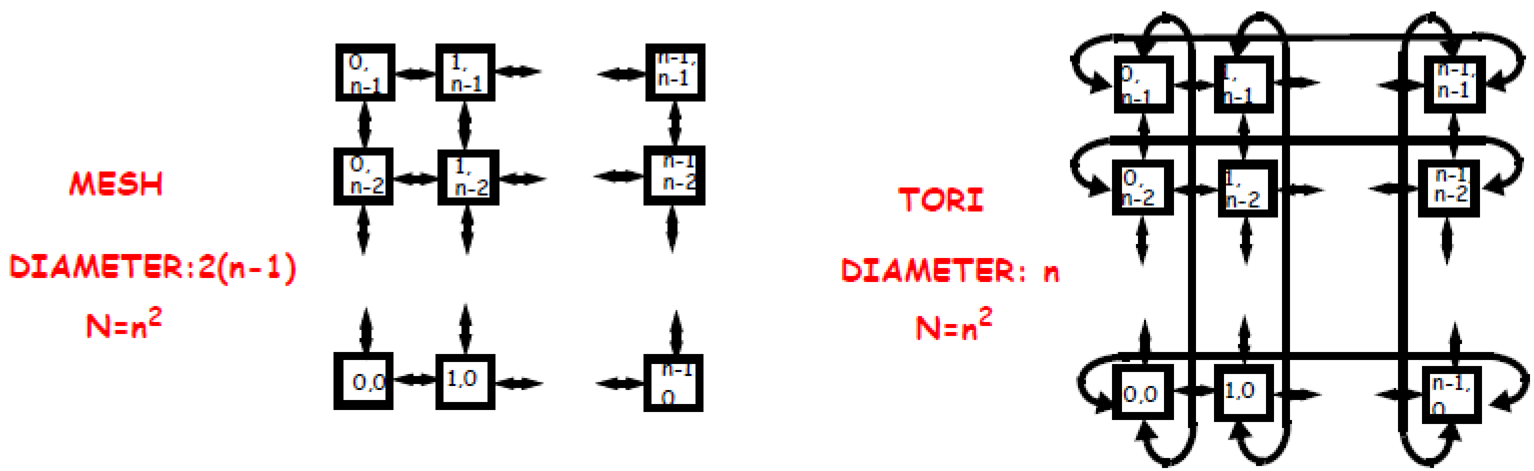
Direct networks: nodes are directly connected to one another

Decentralized

- Linear array and ring



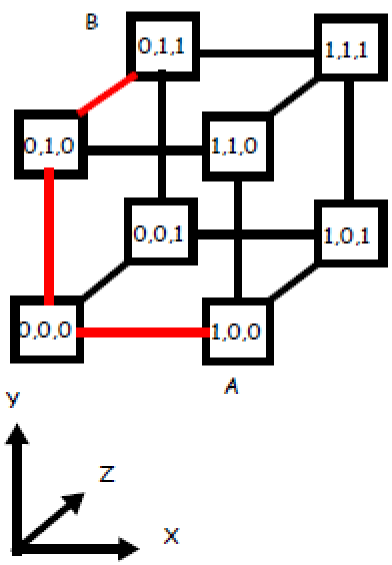
-]



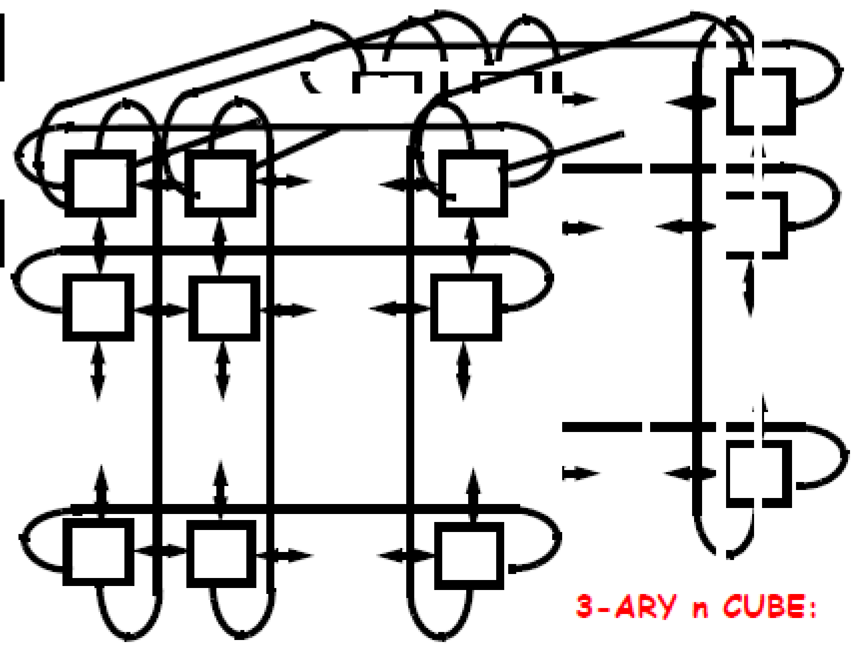
Topologies

Direct networks: nodes are directly connected to one another

Hypercube and k-ary n-cube



3-D HYPERCUBE
3-ARY 2 CUBE

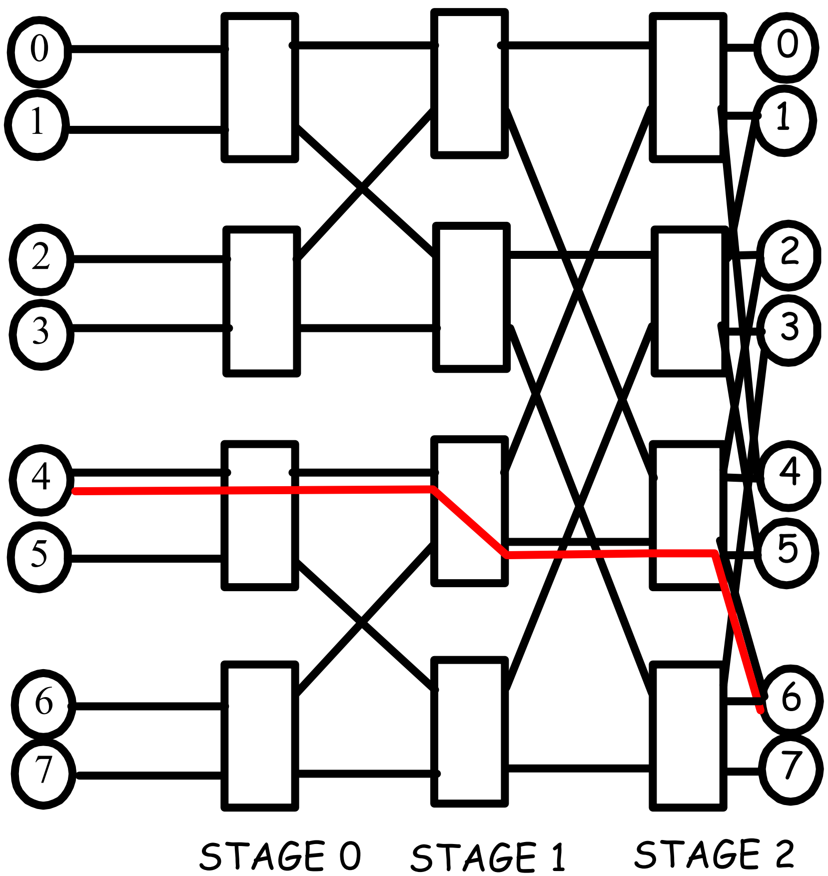


3-ARY n CUBE:

k (-ary): DIMENSION
n (-cube): NB OF NODES IN EACH DIMENSION

Routing algorithms

Butterfly network



USE RELATIVE ADDRESS

- BITWISE EXCLUSIVE OR OF SOURCE AND DESTINATION ADDRESSES TO FORM THE ROUTING ADDRESS
- IF BIT i IS ZERO, ROUTE STRAIGHT
- IF BIT i IS ONE, ROUTE ACROSS

EXAMPLE: ROUTE FROM 4 TO 6

- SOURCE: 100
- DESTINATION: 110
- EX-OR: 010