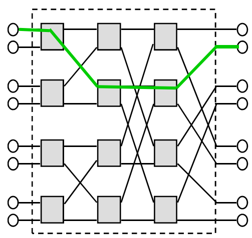
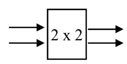
Interconnection networks

1) Large switch structures built from smaller switches

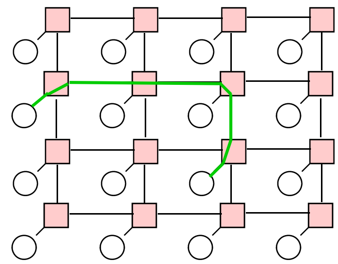


A 2x2 switch or router

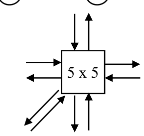


A processor and/or memory

2) Topology-induced switching structures

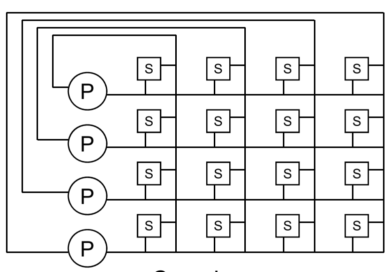


A 5x5 switch or router

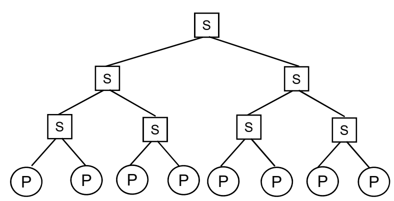


A processor + memory

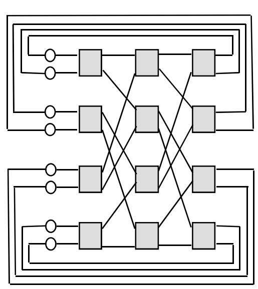
Common interconnections



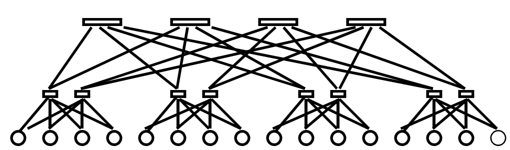
Crossbar



Tree

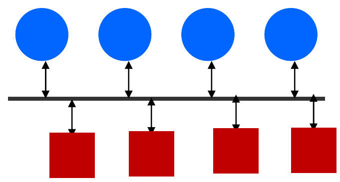


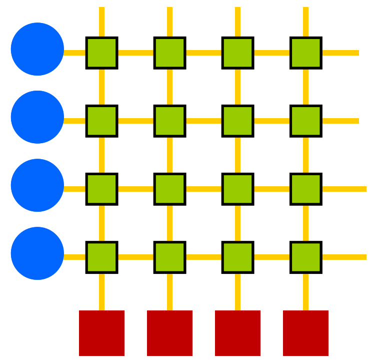
Multistage



Fat tree

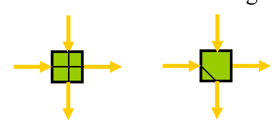
Buses and crossbars



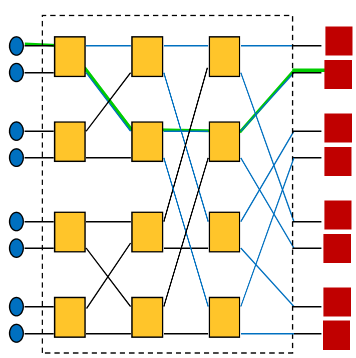


- Cost
- Latency
- Bandwidth
- Scalability

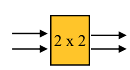
Each switch is a 2x2 switch that can be set to one of 2 settings

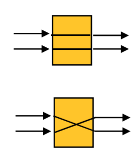


Multistage networks



A 2x2 switch or router





Circuit switching: circuits are established between inputs and outputs – arbitrate entire circuits.

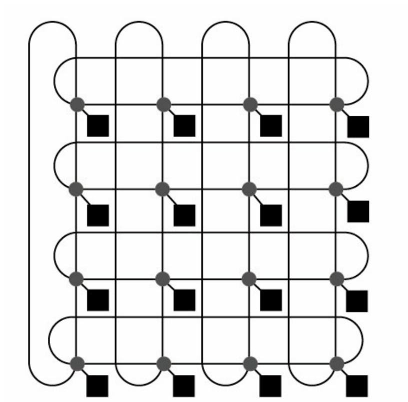
Packet switching: packets are buffered at intermediate switches – arbitrate individual switches.

- NxN Omega network: log N stages, with N/2, 2x2 switches.
- A blocking network: some input-output permutations cannot be realized due to path conflicts.

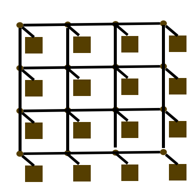
Evaluating Interconnection Network topologies

- *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 wires you must cut to divide the network into two equal parts.
- *Cost:* The number of links or switches (whichever is asymptotically higher) is a meaningful measure of the cost. However, a number of other factors, such as the ability to layout the network, the length of wires, etc., also factor in to the cost.

2-D torus



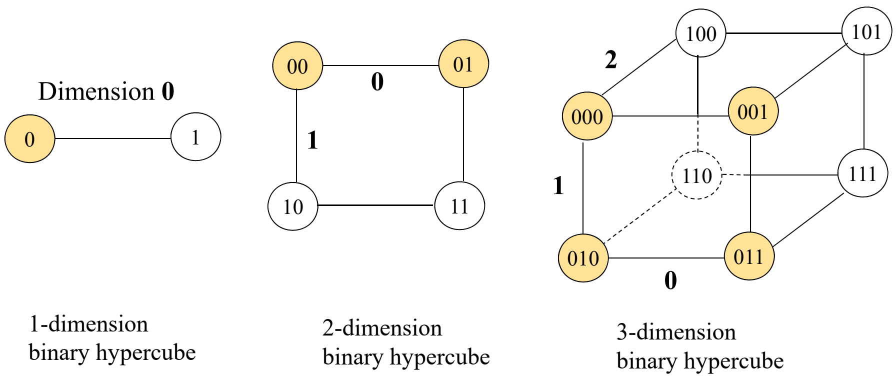
- Diameter??
- Bisection bandwidth??
- Routing algorithms
  - x-y routing
  - Adaptive routing
- 2D mesh (without the wrap-around connections)



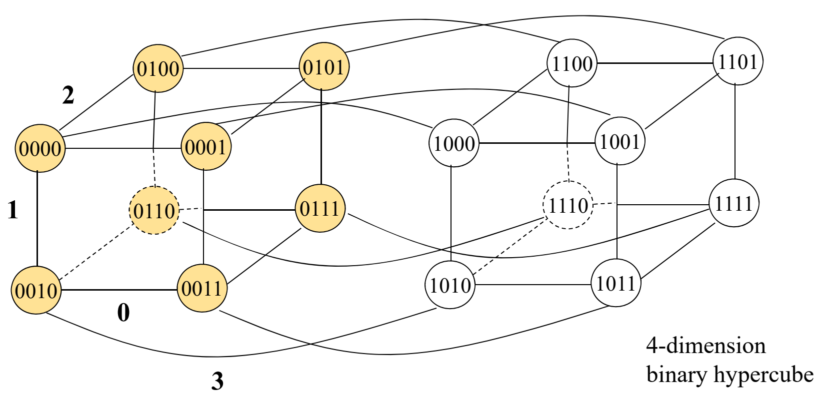
- **Variants**
  - 1-D (ring), 3-D.

Hypercube interconnections

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

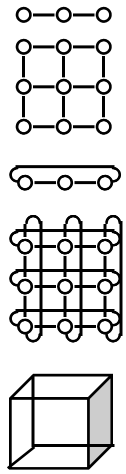


A 4-dimension Hypercube (16 nodes)



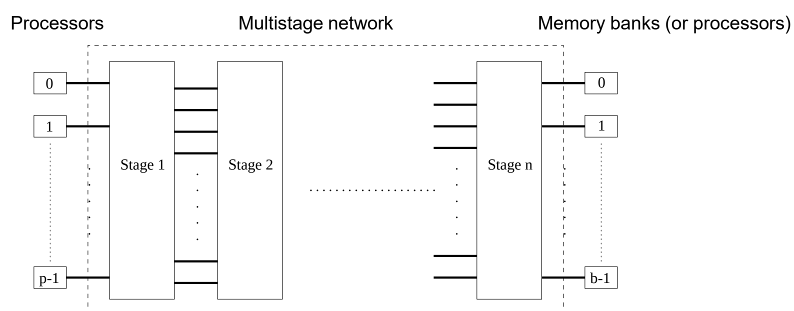
- Can recursively build a q-dimension network – has 2^q nodes

Common interconnection topologies



					N = 1024	
Type	Degree	Diameter	Av Dist	Bisection	Diam	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

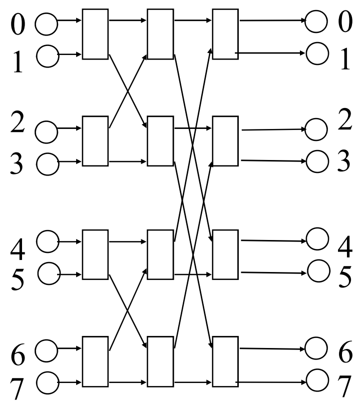
Multistage Networks



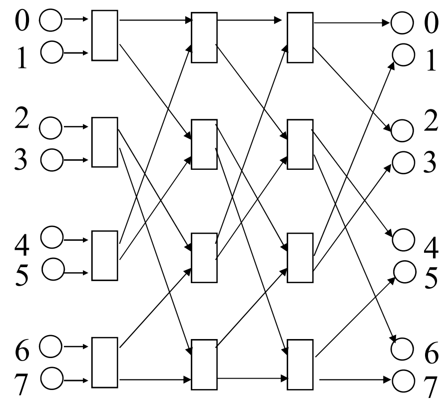
The schematic of a typical multistage interconnection network.

If 2x2 switches are used to build an *pxp* switch (to connect *p* processors to *p* memory banks – *p* being a power of 2), we need at least *log p* stages – why??.

Examples of MINs connecting 2^q inputs to 2^q outputs (using q stages of 2x2 switches)



A butterfly network



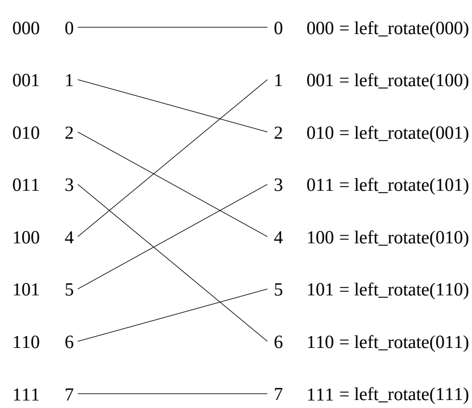
An Omega network
(multiple shuffle/exchange)

Multistage Omega Network

A *pxp* Omega network consists of:

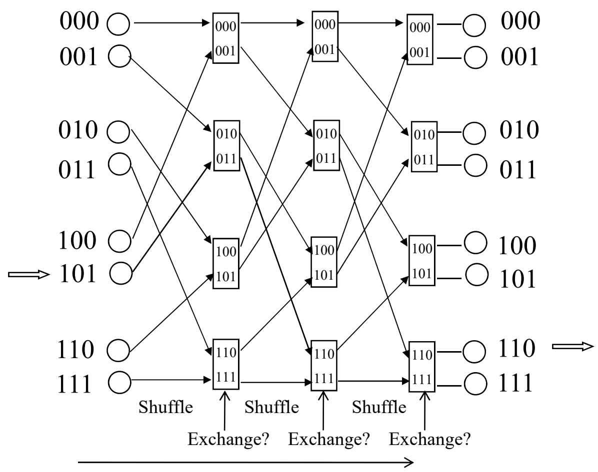
- $q = log\ p$ stages, each having $p/2$, 2x2 switches
- Perfect shuffle connection between stages
  - i connects to $2i$ for $i = 0, \ldots , p/2-1$
  - i connects to $2i+1-p$ for $i = p/2, \ldots , p-1$

- Formally, Shuffle($x_{q-1} , x_{q-2} , \ldots , x_0$) = $x_{q-2} , \ldots , x_0 , x_{q-1}$



Multistage Omega Network

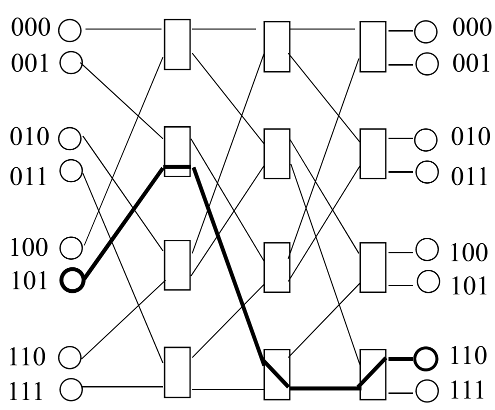
An 8x8 omega network



Shuffle(x_{q-1} , x_{q-2} , … , x_0) = x_{q-2} , … , x_0 , x_{q-1}
Exchange(x_{q-1} , x_{q-2} , … , x_0) = x_{q-1} , x_{q-2} , … , $\overline{x}_0$

Routing in an OMEGA network

(Unique route between a source and a destination)

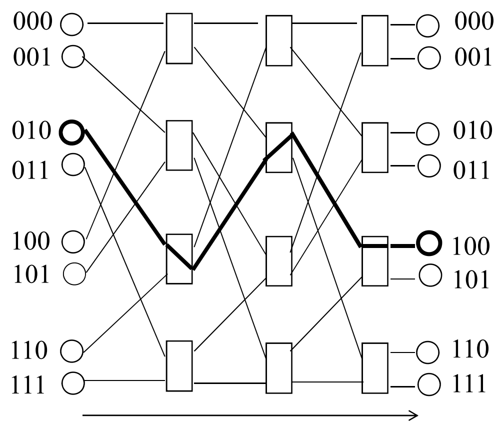


The destination $d_2\, d_1\, d_0$ is coded in message header

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	

Routing in an OMEGA network



Example: to route from source 010 to destination 100
010 xor 100 = 110 = (cross, cross, straight)
Route: cross at first level, cross at second level, straight at last level

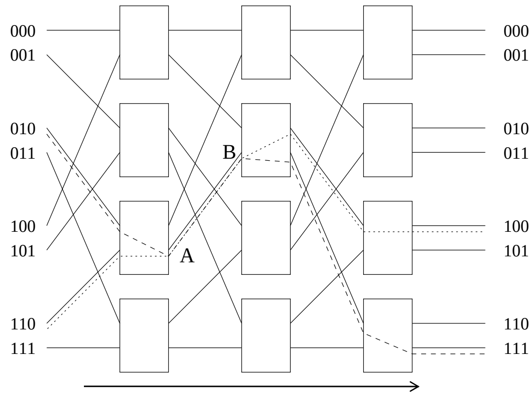
Routing in an OMEGA network

To get from $s_{q-1}, s_{q-2}, \ldots, s_0$ to $d_{q-1}, d_{q-2}, \ldots, d_0$

- Do q shuffles
- After each shuffle, do an exchange to match the corresponding destination bit

```
Source                         01011011
Destination                    11010110
Positions that differ          ^   ^^ ^
Route    01011011  Shuffle to  10110110  Exchange to  10110111
         10110111  Shuffle to  01101111
         01101111  Shuffle to  11011110
         11011110  Shuffle to  10111101
         10111101  Shuffle to  01111011  Exchange to  01111010
         01111010  Shuffle to  11110100  Exchange to  11110101
         11110101  Shuffle to  11101011
         11101011  Shuffle to  11010111  Exchange to  11010110
```

The Omega Network is blocking



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

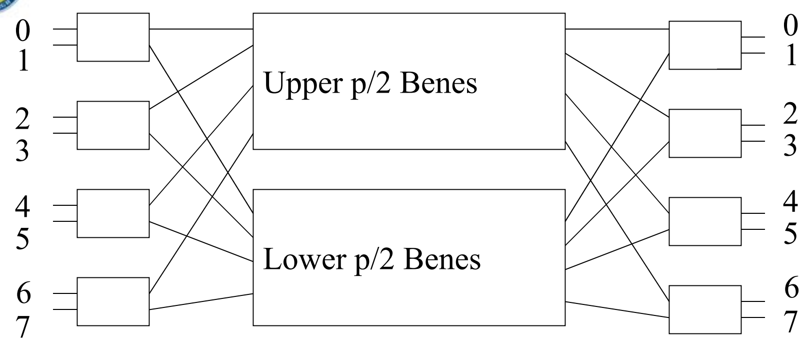
Only a fraction of the *p*! permutations can be realized in an omega network (can you formally prove?).

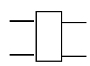
Capabilities for realizing arbitrary permutations

- *Blocking networks*: cannot realize an arbitrary permutation without conflict -- for example, Omega can realize only $p^{p/2}$ permutations.
- *Non-blocking networks*: can realize any permutation on-line -- for example, cross-bar switches.
- *Re-arrangeable networks*: can realize any permutation off-line -- for example, a Benes network can establish any connection in a permutation if it is allowed to re-arrange already established connections.

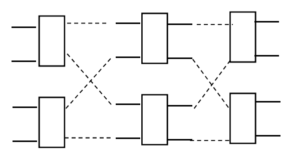
The Benes network

Can be built recursively -- an *p*x*p* Benes is built from two (*p/2* x *p/2)* Benes networks plus two columns of switches.

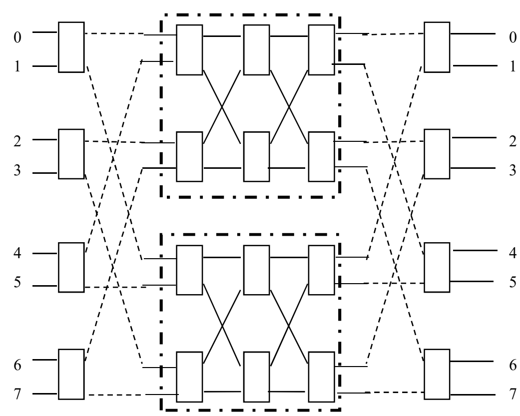




A 2x2 Benes network

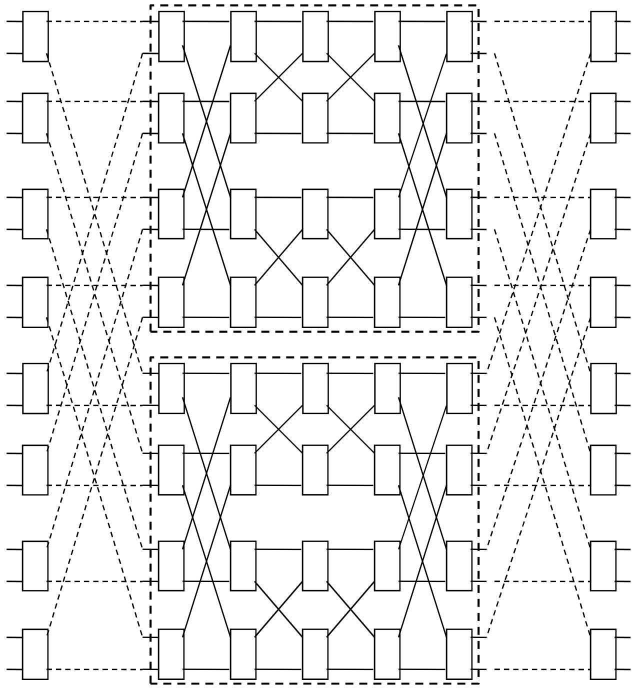


A 4x4 Benes network



An 8x8 Benes network

A 16x16 Benes network



21

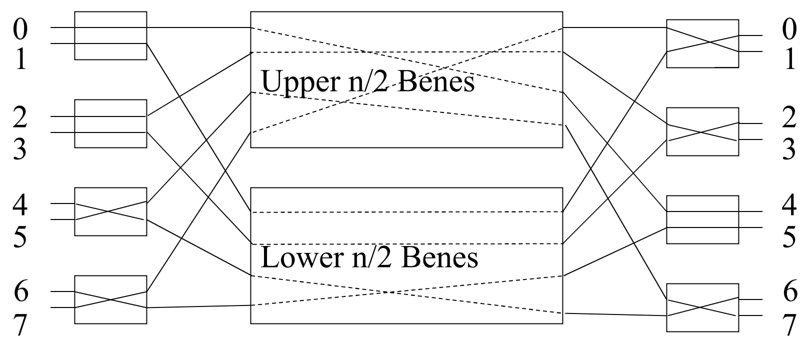
To realize a permutation (i, o_i, $i = 0, \ldots, p-1$) in an pxp Benes network:

- For each connection (i, o_i), determine whether it goes through the upper or lower p/2 Benes.
- Repeat this process recursively for each of the $p/2$ networks.

- Start with $k=0$ and (k, o_k) going through the upper Benes,
- If o_k shares a switch at the last column with o_m, then route (m, o_m) through the lower Benes.
- If j shares a switch at the first column with m, then route (j, o_j) through the upper Benes.
- Continue until you get an input that shares a switch with input 0.
- If there are still unsatisfied connections, repeat the looping.

22

Example for establishing a permutation:
(0,4), (4,2), (3,6), (1,0), (2,3), (6,5), (5,7), (7,1)

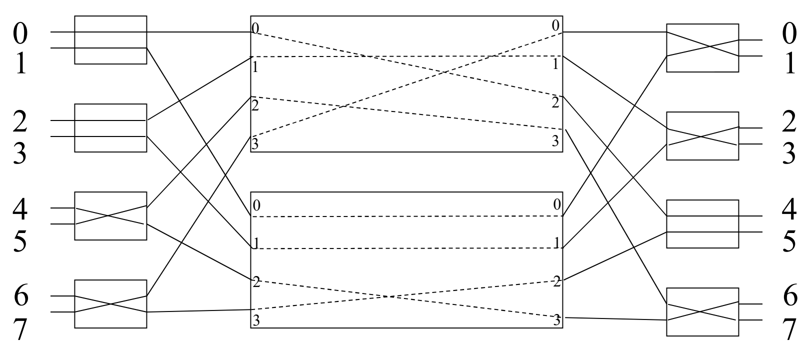


(0,4) upper,
(6,5) lower,
(7,1) upper,
(1,0) lower,

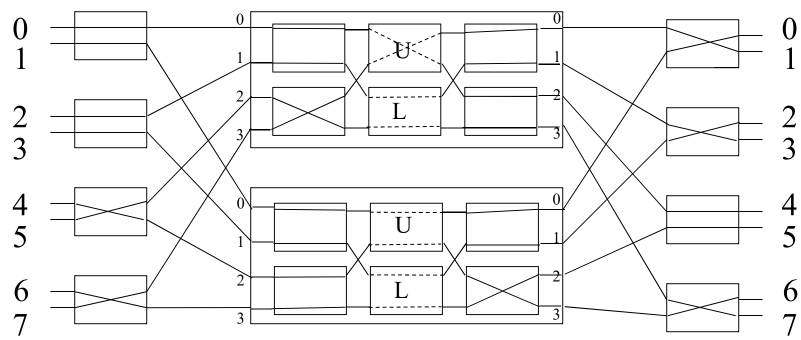
+

(2,3) upper,
(4,2) lower,
(5,7) upper,
(3,6) lower,

Set the upper n/2 network to satisfy (0,2), (1,1), (2,3), (3,0)
Set the lower n/2 network to satisfy (0,0), (1,1), (2,3), (3,2)



Set the upper n/2 network to satisfy (0,2), (1,1), (2,3), (3,0)
Set the lower n/2 network to satisfy (0,0), (1,1), (2,3), (3,2)



Setting the upper n/2 network
(0,2) U,
(2,3) L,
(3,0) U,
(1,1) L

Setting the lower n/2 network
(0,0) U,
(1,2) L,
(2,3) U,
(3,2) L

Fat tree networks

Eliminates the bisection bottleneck of a binary tree

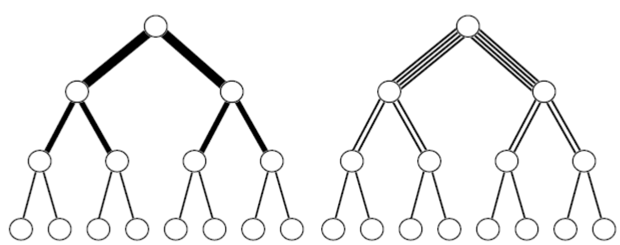
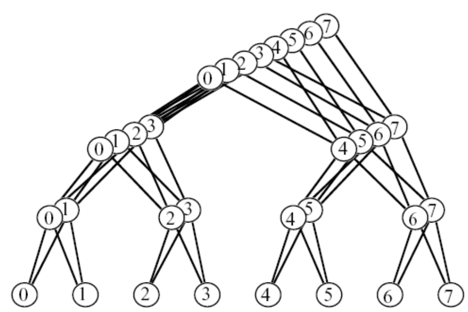
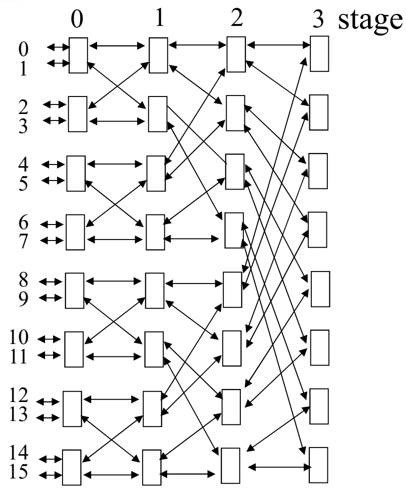
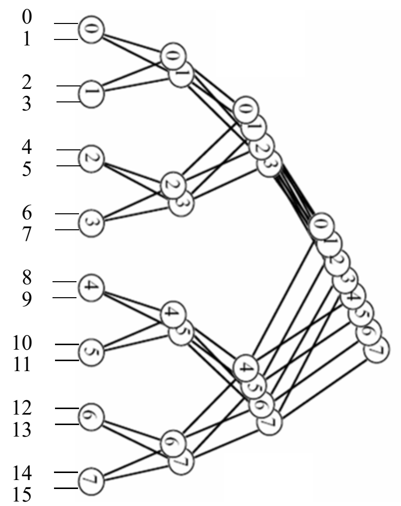


Fig. 15.6. Two representations of a fat tree.



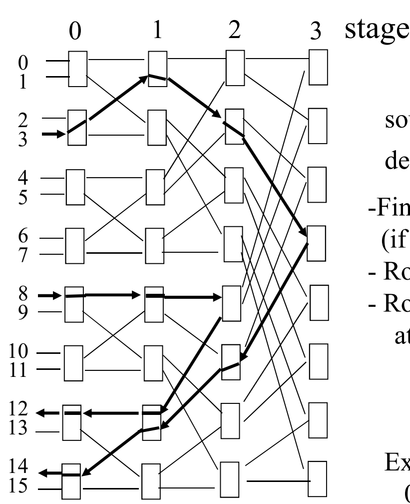
A 16-node fat tree network





A fat tree networks using 2x2 bidirectional switches

Routing in a fat tree



source $s_{q-1}, s_{q-2}, \ldots, s_0$

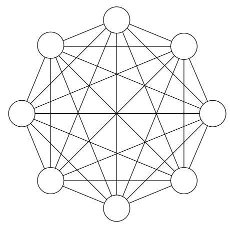
destination $x_{q-1}, x_{q-2}, \ldots, x_0$

-Find smallest k such that $s_i = x_i$, $i=k+1,\ldots,q-1$
(if no such k exists, then $k = q-1$)

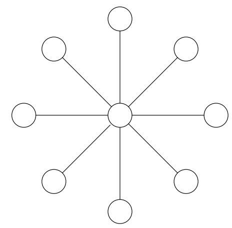
- Route arbitrarily up the tree to a switch in stage k
- Route down the tree as follows:
  at stage i, $i= k, \ldots, 0$
    if $x_i = 0$, route to upper port
    else route to lower port

Examples (q = 4):
  0011 → 1110 (k = 3)
  1000 → 1100 (k = 2)

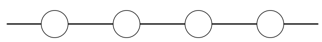
Other networks



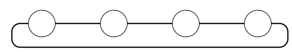
A completely-connected network



A star connected network



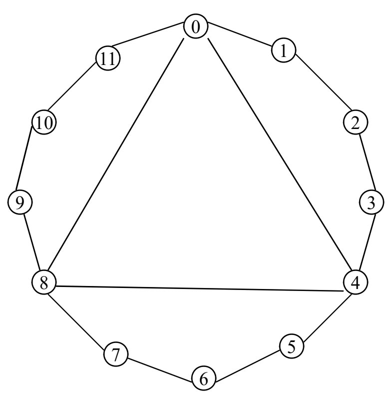
with no wraparound links

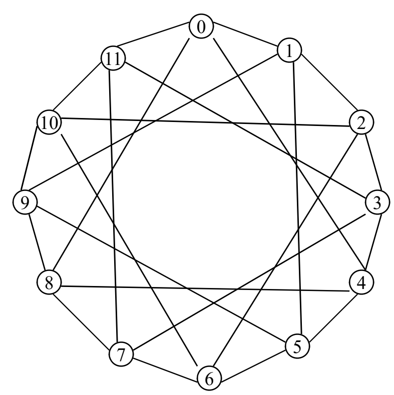


with wraparound link (ring).

Linear arrays

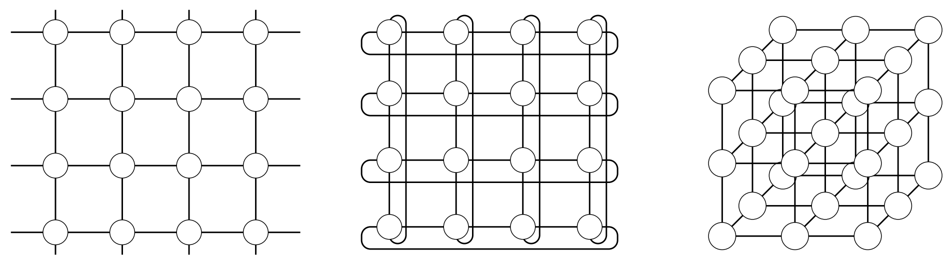
Enhanced ring networks





- Cords (in a chordal ring) may bypass any given number of nodes
- May have more than one set of chords
- What is the diagonal of a chordal ring?
- How do you route?

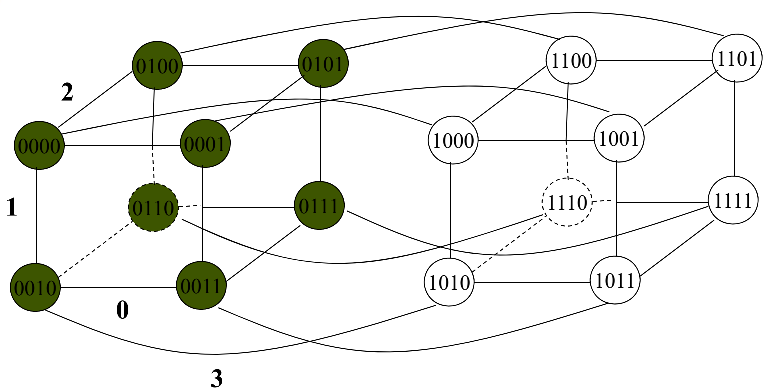
Two- and Three Dimensional Meshes



(a) (b) (c)

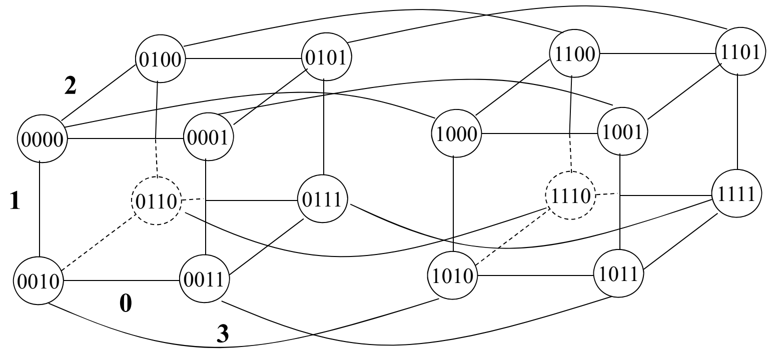
Two and three dimensional meshes: (a) 2-D mesh with no wraparound; (b) 2-D mesh with wraparound link (2-D torus); and (c) a 3-D mesh with no wraparound.

Hypercube interconnections



For a *q* dimension hypercube, calculate

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

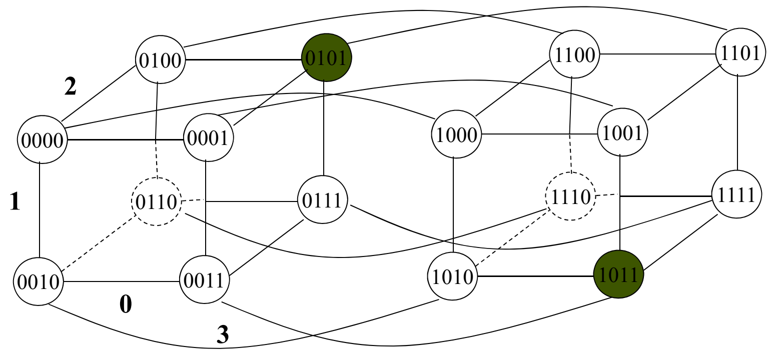


- Each node in a q-dimension hypercube has a q-bits identifier $x_{q-1}, \ldots, x_1, x_0$
- Identifiers of nodes across a dimension j differ in the j^{th} bit (have a unit **Hamming distance**)
- How do you find the route between a source, *s*, and a destination, *d*?

Routing Algorithm

- ***Routing algorithms*** **define the path taken by a packet between source and destination.**
- **One goal is to prevent deadlock, livelock, and starvation**
  - o Deadlock: packets waiting for each other in a cycle.
  - o *Livelock:* packets circulating the network without making any progress towards their destination.
  - o *Starvation:* packets being blocked in buffer as the output channel is always allocated to other packets.
- **Adaptive routing provides alternative paths for packets that encounter unfair channel allocation, faulty hardware, or hot spots in traffic patterns.**
- **Routing algorithms can be classified into**
  - o **Non-adaptive: one unique, predetermined path.**
  - o **Partially adaptive: can route by multiply paths, but not every paths.**
  - o **Fully adaptive: can route along any shortest path in the network**

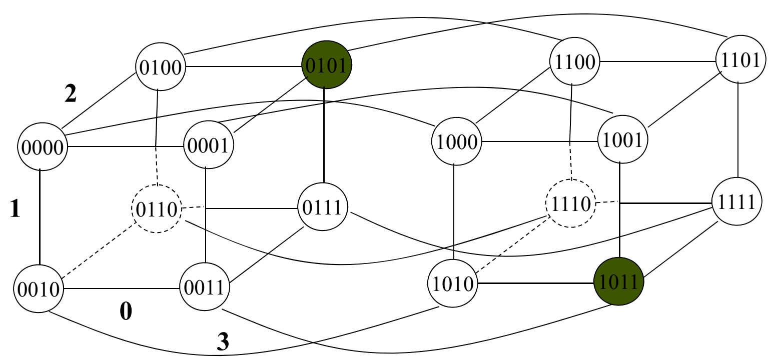
Routing on a hypercube



A message from a source, $s_{q-1}, \ldots, s_0$, to a destination $x_{q-1}, \ldots, x_0$ has to cross any dimension, b, for which $x_b \neq s_b$

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

Dimension-order routing



When a node, $n_{q-1}, \ldots, n_0$, receives a message for destination node $x_{q-1}, \ldots, x_0$, it executes the following

- If $x_k = n_k$ for $k = 0, \ldots, q-1$, , keep the message
- Else { Find the largest k such that $x_k \neq n_k$;

  Send the message to the neighbor across dimension k }