IN (Interconnection Network)

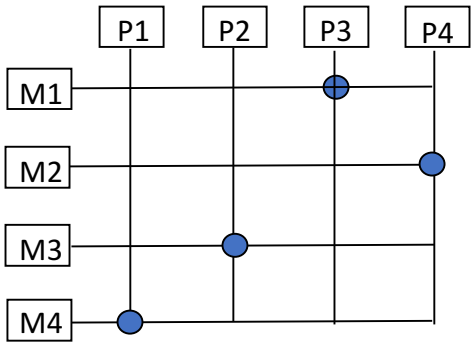
IN in MIMD shared-memory systems

Bus – shared among connected components;

flexible, low cost but, contention problem

→ not efficient for connecting many components (not scalable)

Cross-bar (x-bar) – switched network



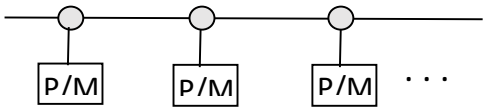
ex) 4 simultaneous comm.
P1 – M4
P2 – M3
P3 – M1
P4 – M2

Both bus and X-bar are dynamic topologies.

IN in MIMD distributed-memory systems

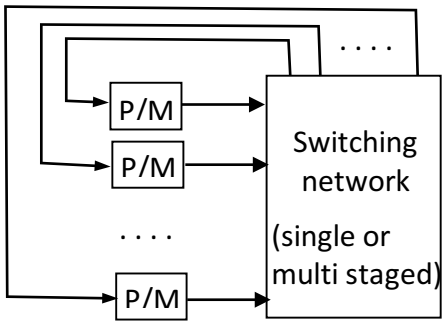
- Direct (static) IN – each switch is directly connected to a P-M pair;

  ex) ring, mesh, torus, fully connected,



- Indirect (dynamic) IN – switches may not be directly connected to a node;

  ex) X-bar (single stage),
  Omega network (multi stages),
  Benes network (multi stages)



Static (direct) topologies

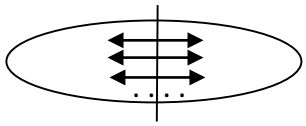
Assumptions/symbols used below:

network size (N) – # of nodes in IN

node degree (d) – # of links per internal node

network diameter (D) – any node can go to any other node within D hops
(i.e., at worst case (longest distance), best solution)

bisection width (connectivity) – # of simultaneous communications between 2 halves of the network



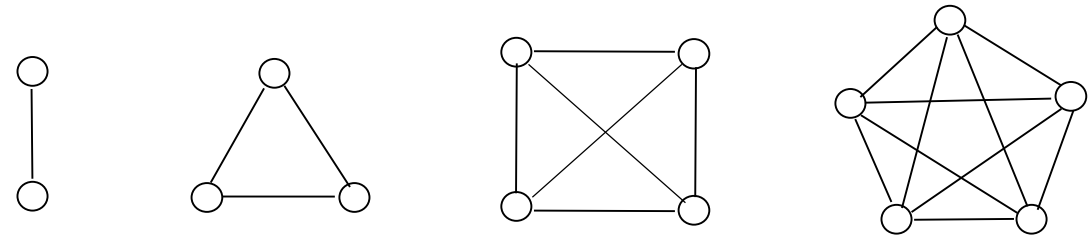
(a) Completely connected

node degree d = N-1 (N-1 links/node)

network diameter D = 1

fault tolerant

routing: directly connect to dest (1 hop) – high HW cost

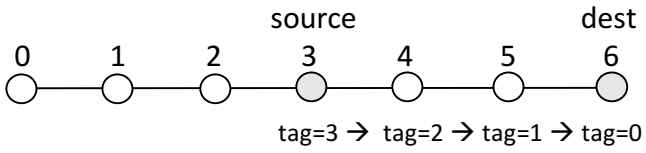


N=2 N=3 N=4 N=5

(b) Linear array

d = 2 links/node

D = N-1



not fault tolerant: one fault → disconnects the entire list

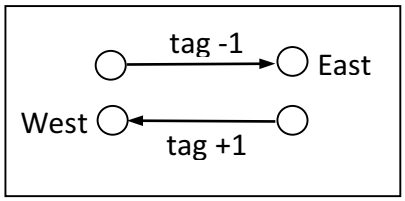
routing: compute 1-D tag value and perform <u>E-W routing</u>;

source tag = dest_id – source_id;

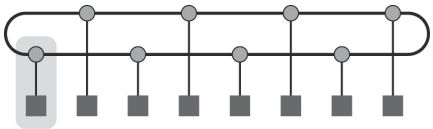
ex) source_id = 3, dest_id = 6

→ source tag = 6-3 = 3

trace until tag = 0;



(c) Ring (bidirectional)



d = 2 links/node

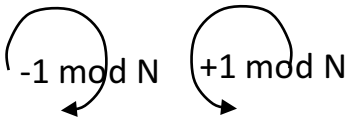
D = $\lfloor N/2 \rfloor$

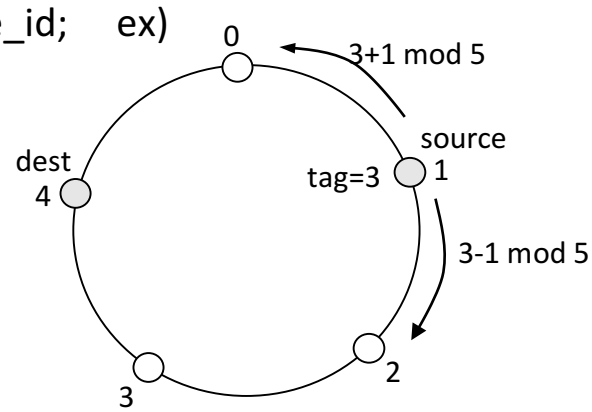
fault tolerant (partly) – both directions available

Routing: source_tag = dest_id – source_id; ex)

new_tag = tag (+/- 1) MOD N;

trace until tag = 0;





(d) Mesh

<u>2-D mesh</u>

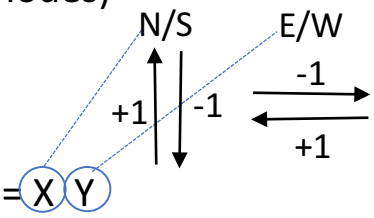
ex) 3^2=9, n=3, k=2

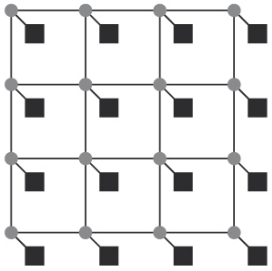
$N = n^k$, where k, n are dimension, size of 1-D

d= 2k = 4 links/node (inner nodes)

D = k(n-1)

fault tolerant



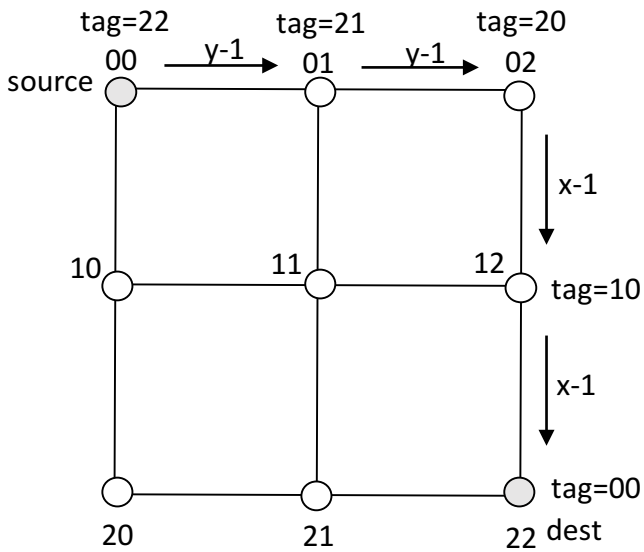


Routing: X-Y routing, tag = X Y

source tag (2-D) = dest_id – source_id;

ex) souece_id = 00, dest_id = 22; → source tag = 22

trace until tag = 00; if a node is busy, take any available path;



3-D mesh

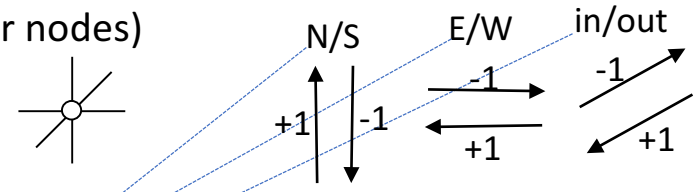
ex) 3^3=27, n=3, k=3

$N = n^k$, where k, n are dimension, size of 1-D

d = 2k = 6 links/node (inner nodes)

D = k(n-1)

fault tolerant, expensive

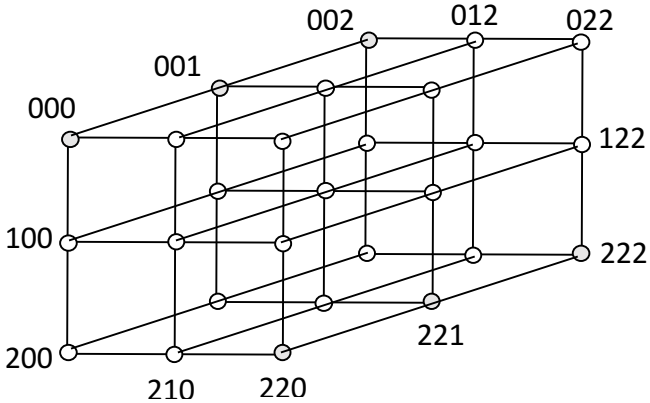


Routing: X-Y-Z routing, tag = X Y Z

source tag value (3-D) = dest_id – source_id;

ex) souece_id = 000, dest_id = 222; → source tag = 222

trace until tag = 000; if a node is busy, take any available path;

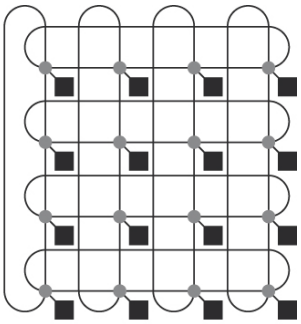


ex) N = 27

d = 2k = 2*3 = 6

D = k(n-1) = 3(3-1) = 6

(e) Torus (2-D and 3-D)



mesh + ring → symmetric

d = 2k, where k is dimension, for all nodes

symmetric node degree, i.e.,

2-D torus: d = 2k = 2*2 = 4 links/node

3-D torus: d = 2k = 2*3 = 6 links/node

D = k $\lfloor \frac{\sqrt[k]{N}}{2} \rfloor$ //ex) 2-D 3x3 torus, D = 2 $\lfloor \frac{\sqrt[2]{9}}{2} \rfloor$ = 2* $\lfloor 3/2 \rfloor$ = 2

reduced network diameter, e.g., N=9, 2-D mesh (D=4) vs. 2-D torus (D=2)

3-D torus used in Cray T3D/TSE

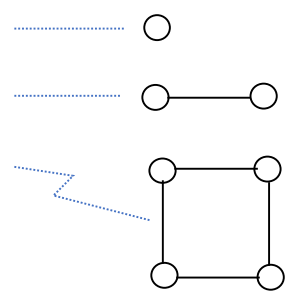
(f) Hypercube

network size N = 2^k, where k is dimension

node degree d = k

diameter D = k

dimension(k)	# of nodes(N)	node degree(d)	diameter(D)
0 (0 cube)	1	0 link/node	0
1 (1 cube)	2	1 link/node	1
2 (2 cube)	4	2 links/node	2
3 (3 cube)	8	3 links/node	3
4 (4 cube)	16	4 links/node	4



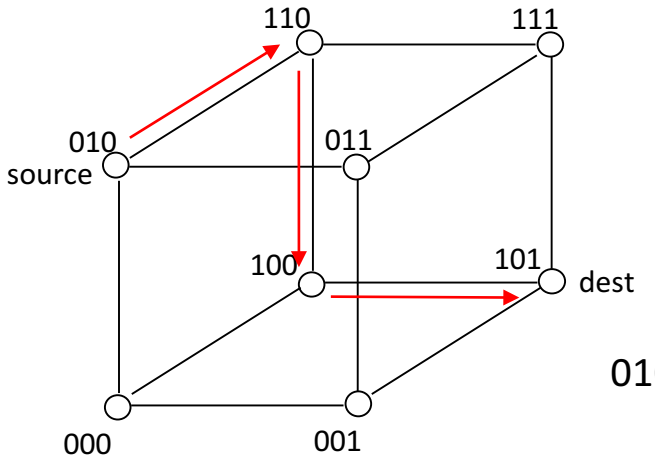
Routing: <u>gray coding</u> for node id – adjacent node_id differs in only 1 bit;

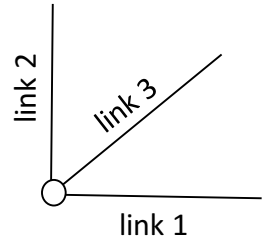
available links = source_id XOR dest_id;

repeat this with new source_id until no links are available;

possible policies: take left-most (or, right-most) available link first;

ex) <u>3-D cube</u> with soure_id = 010, dest_id = 101





010 XOR 101 → 1 1 1

link3 *link2* *link1*

available links to take

if we use the policy of taking the left-most avail. one first,

take link3: new soure_id = 110, and perform

110 XOR 101 → 0 1 1 (link2 and link1 are available);

take link2: new source_id = 100, and perform

100 XOR 101 → 0 0 1 (link1 is available);

take link1: new source_id = 101, and perform

101 XOR 101 → 000 (no links are available, stop)

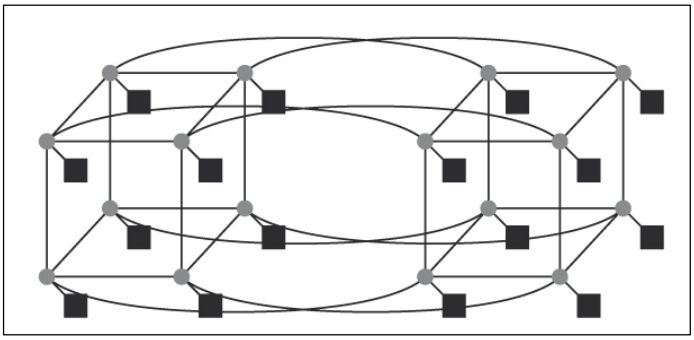
4-D cube routing:

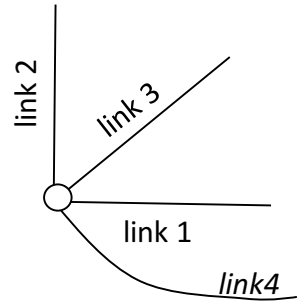
gray coding and do analogous operation;

ex) source_id = 0011, dest_id = 1100

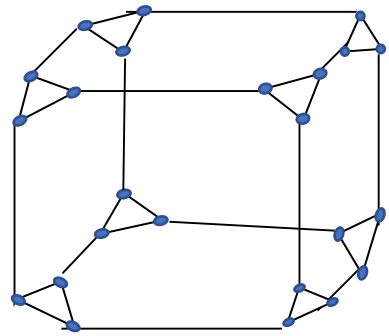
0011 XOR 1100 → 1111 (order of avail. link4,3,2,1)

4-D hypercube





(g) Cube-connected cycles (CCC)



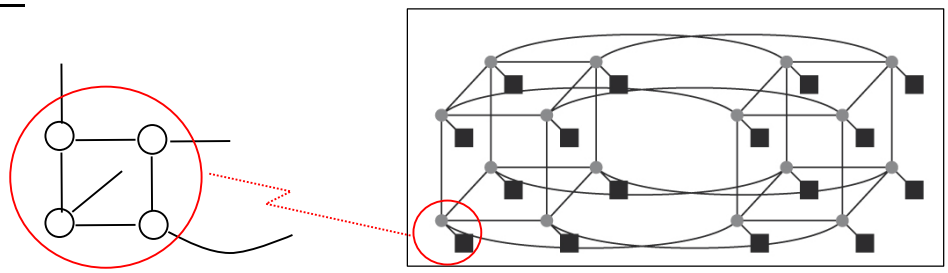
<u>3-D CCC (3-CCC)</u>

k-CCC : $k*2^k$ nodes

ex) 3-CCC,
$3*2^3 = 24$ nodes

fixed node degree (independent from k), d=3
network diameter $D = 2k - 1 + \lfloor k/2 \rfloor \approx 2k$

<u>4-CCC</u> : refined 4 nodes in each node of 4-cube



d = 3 (fixed)

$D = 2k - 1 + \lfloor k/2 \rfloor = 2*4 - 1 + \lfloor 4/2 \rfloor = 9$

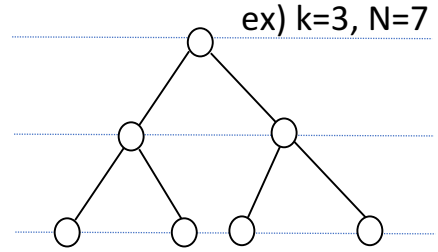
(h) binary tree

node degree = 3 (3 links/node), diameter $D = 2(k - 1)$

k-level binary tree: $N = \frac{2^k - 1}{2 - 1}$ nodes

k-level m-way tree: $N = \frac{m^k - 1}{m - 1}$ nodes

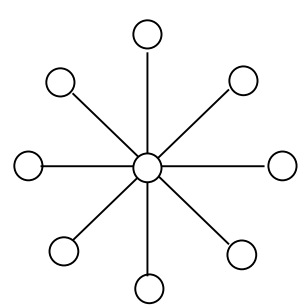
ex) k=3 → N=7 nodes (Fig(h))



(k) star

node degree (internal node) $d = N - 1$

diameter D = 2



Dynamic IN

- Bus – not used in SIMD, used in shared-mem MIMD
- X-bar (crossbar) – single stage – shared-mem (P-M, P-P), dist-mem(P/M-P/M)
- MIN (multi-stage IN)
  - Omega network
  - Benes network

  based on multi-stage shuffling

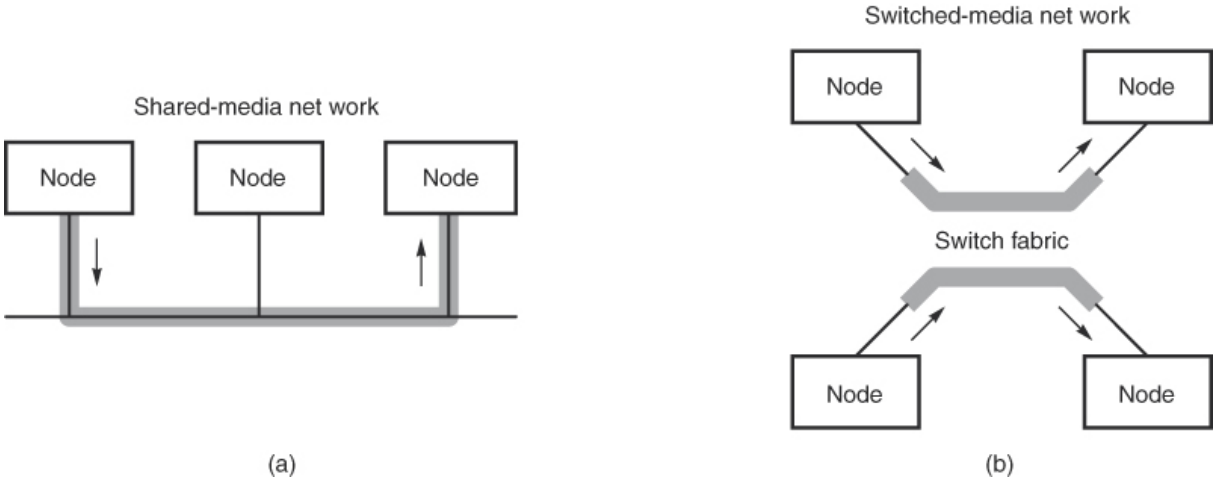
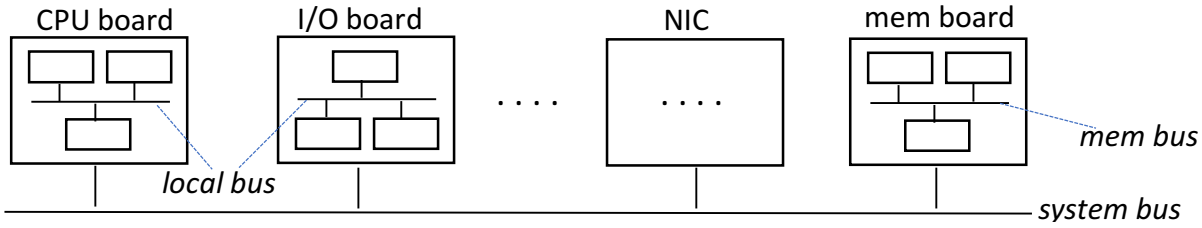


Figure F.8 (a) A shared-media network versus (b) a switched-media network. Ethernet was originally a shared media network, but switched Ethernet is now available. All nodes on the shared-media must dynamically share the raw bandwidth of one link, but switched-media networks can support multiple links, providing higher raw aggregate bandwidth.

Bus: a collection of wires/connectors for data transactions among processors, memory modules and peripheral devices.

- system bus (P ↔ M) – datapath, address/control lines;
- local bus – in each board (CPU board, mem board, I/O board);
- memory bus – local bus in the memory board;



disadvantages: time shared use: bus access – one at a time → contention;
limited scalability;
processor bandwidth is a portion of the total bus bandwidth;

Crossbar switches

much higher bandwidth than bus;

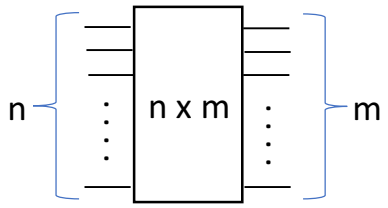
single stage switched network via cross point on/off;

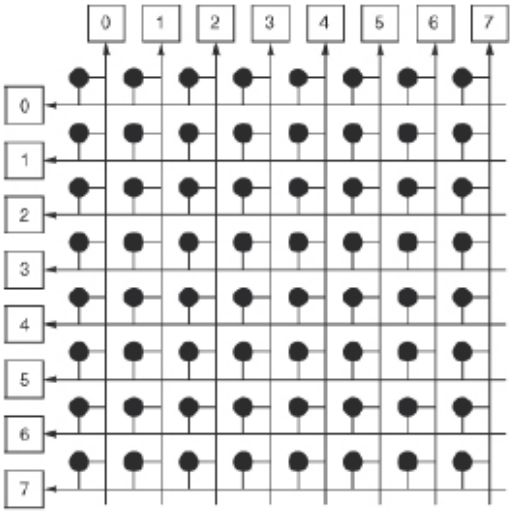
number of cross points → HW cost/complexity;

usage: P-P communication in SMP, MPP, COW;
P-M communication in SMP, PVP;

ex) 8x8 X-bar switch

in general, n*m, where
$n,m = 2^x$





The crossbar network requires N^2 crosspoint switches, shown as black dots

Concurrent communication is available, e.g., PE1 – M2 and PE3 – M1;

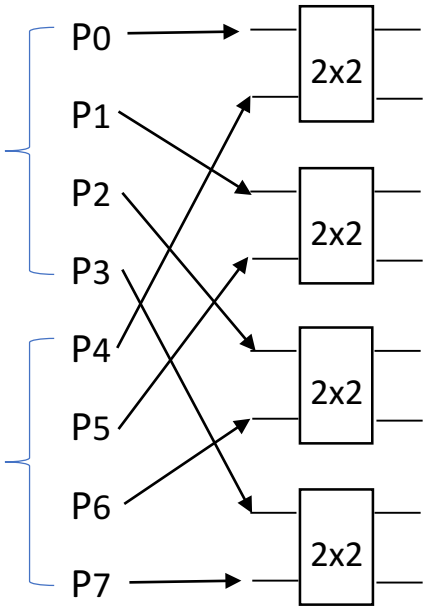
uniform latency – only 1 set of switches in any path;

contention case → one waits;

operation of MIMD requires P-M IN be changed in a dynamic fashion;

Shuffling

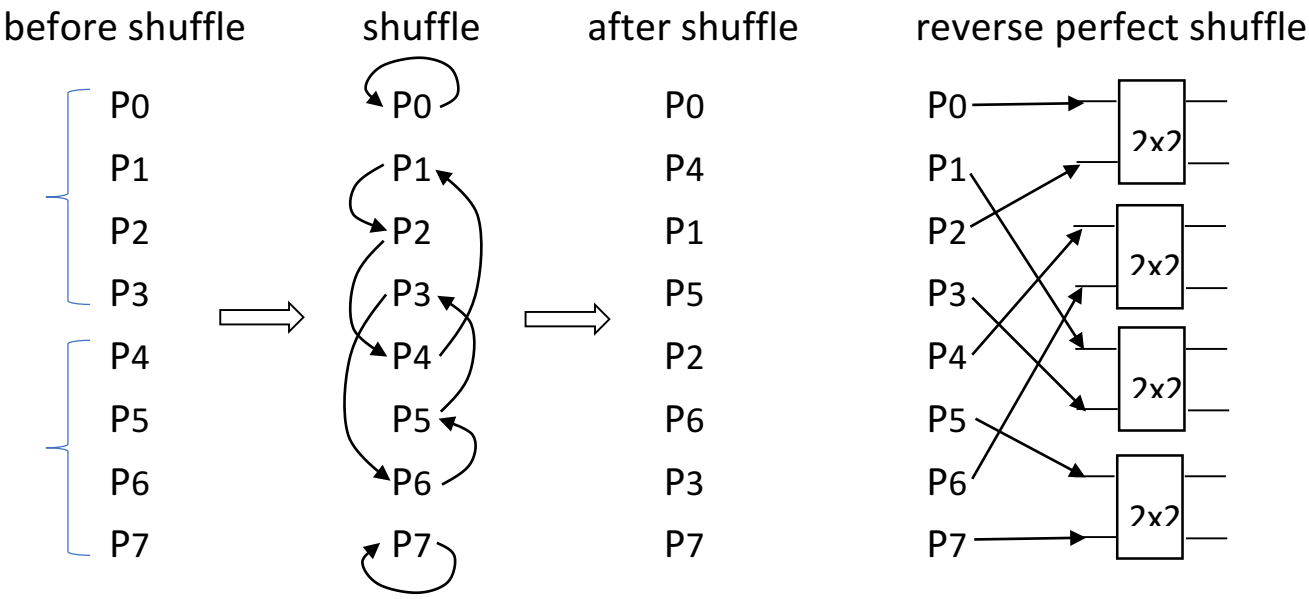
perfect shuffle



Concept:

divide PEs into 2 groups, e.g., (0,1,2,3) and (4,5,6,7);

2 groups are then merged, s.t.,
- 0 is adjacent to 4;
- 1 is adjacent to 5;
- 2 is adjacent to 6;
- 3 is adjacent to 7;

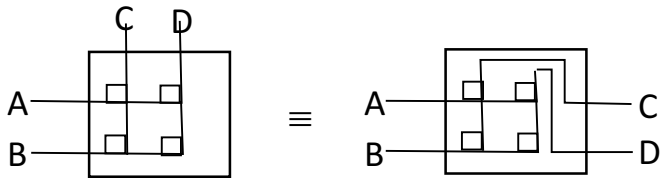


if $i < N/2$, shuffle $(i) = 2*i$

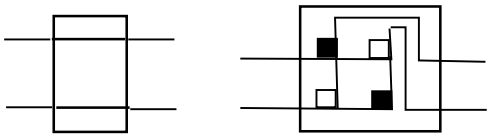
if $i \geq N/2$, shuffle $(i) = [(2*i) \text{ MOD } N] + 1$

node_id: x y z ex) P0 (000) → 000; P1 (001) → 010; P2 (010) → 100

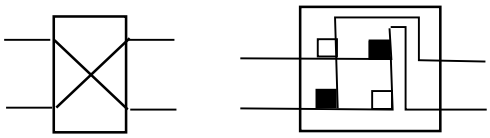
2x2 X-bar switch operations



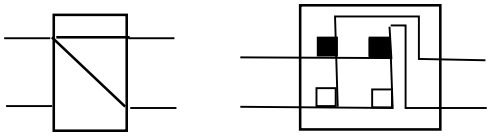
pass:



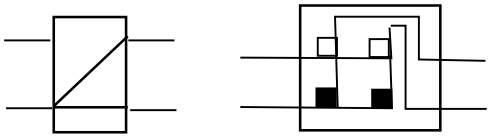
cross:



upper-broadcast:



lower-broadcast:



MIN (multi-stage IN)

dynamic switch connection (on/off);

fixed inter-stage connections between adjacent stages;

ex) Omega network, Benes network

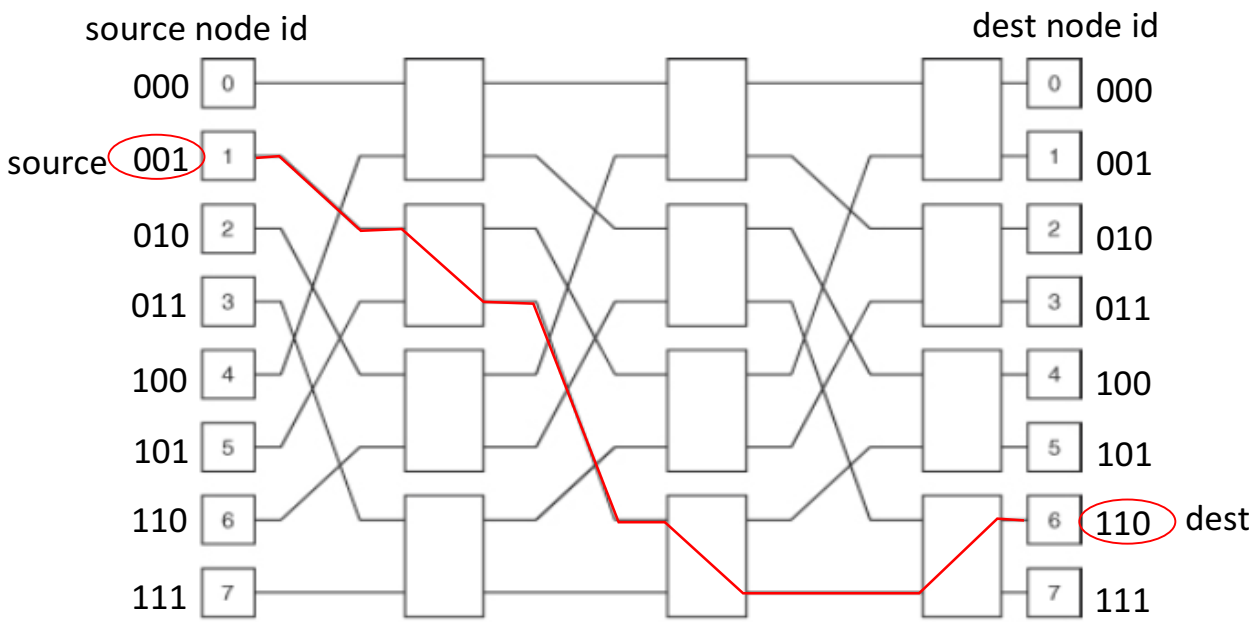
Omega network

N x N Omega network with 2x2 X-bar switches:

- logN stages, each stage with $\frac{N}{2}$ 2x2 X-bar switches
  → total (logN) * $\frac{N}{2}$ 2x2 X-bar switches;
- perfect shuffle between stages;
- packet switching (data, dest-address);

read in each switch

ex) N=8 (i.e., 8x8)



routing: dest_id routing

ex) dest_id = 1 1 0

1st switch low 2nd switch low 3rd switch high

- Any node to any node comm. is possible, but ∃ only 1 unique path from an input (source) to an output (dest).

  problem: blocked case; → Sol: buffered in switch and wait

Benes network

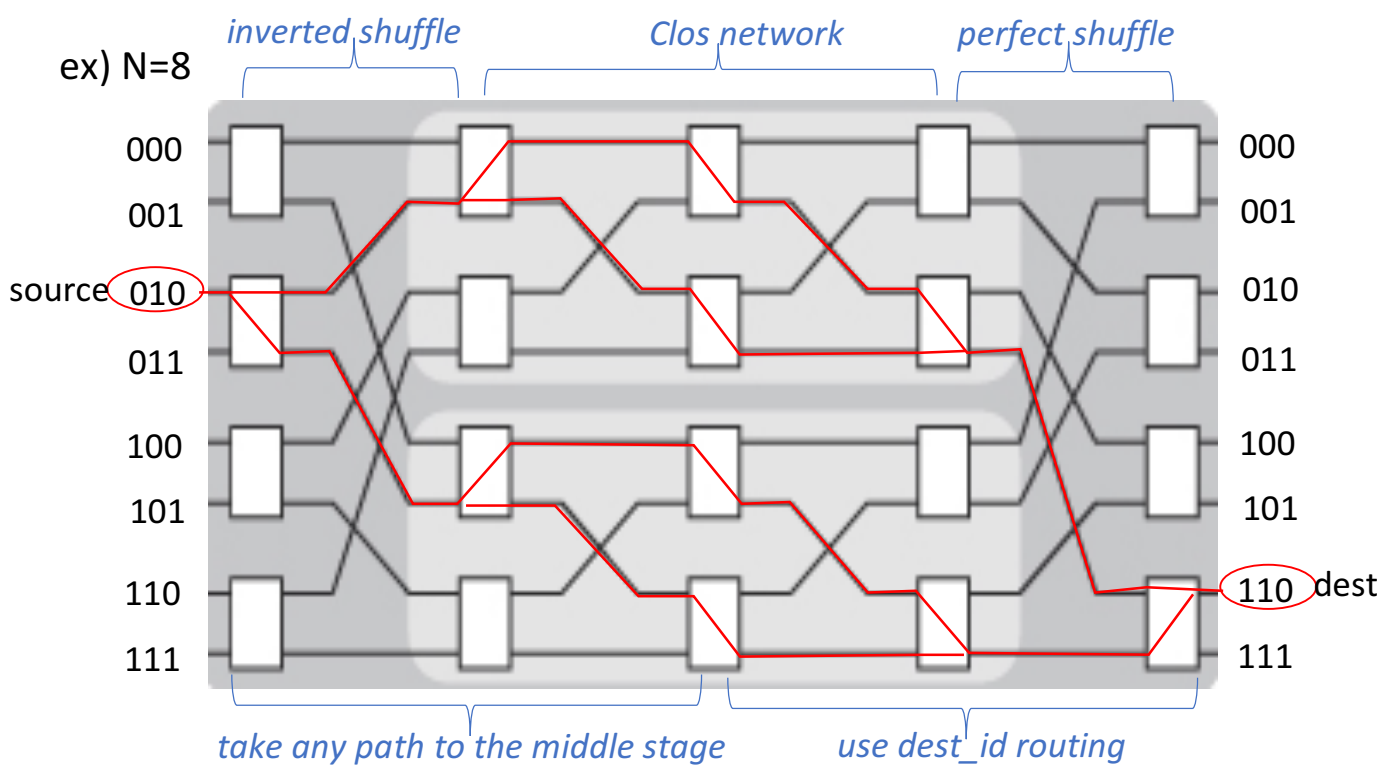
- ∃ multiple paths between a source and a dest.;
  → reduced blocking, increased bandwidth;
- More complex HW;
- increased network latency;
  increased # of stages → complexity in routing path computation;
  so, suitable for circuit switching since the routing path computation takes a considerable amount of time.

# of stages = 2(log N) – 1

ex) N=8 → 5 stages, N=16 → 7 stages

total # of 2x2 switches = $\frac{N}{2}[\,2(logN) - 1]$

ex) N=8



total # of paths between a source and dest = $2^{\lfloor \text{total \# of stages}/2 \rfloor}$

ex) N=8 → 2^2 = 4 paths

N=16 (16x16) Beneŝ network

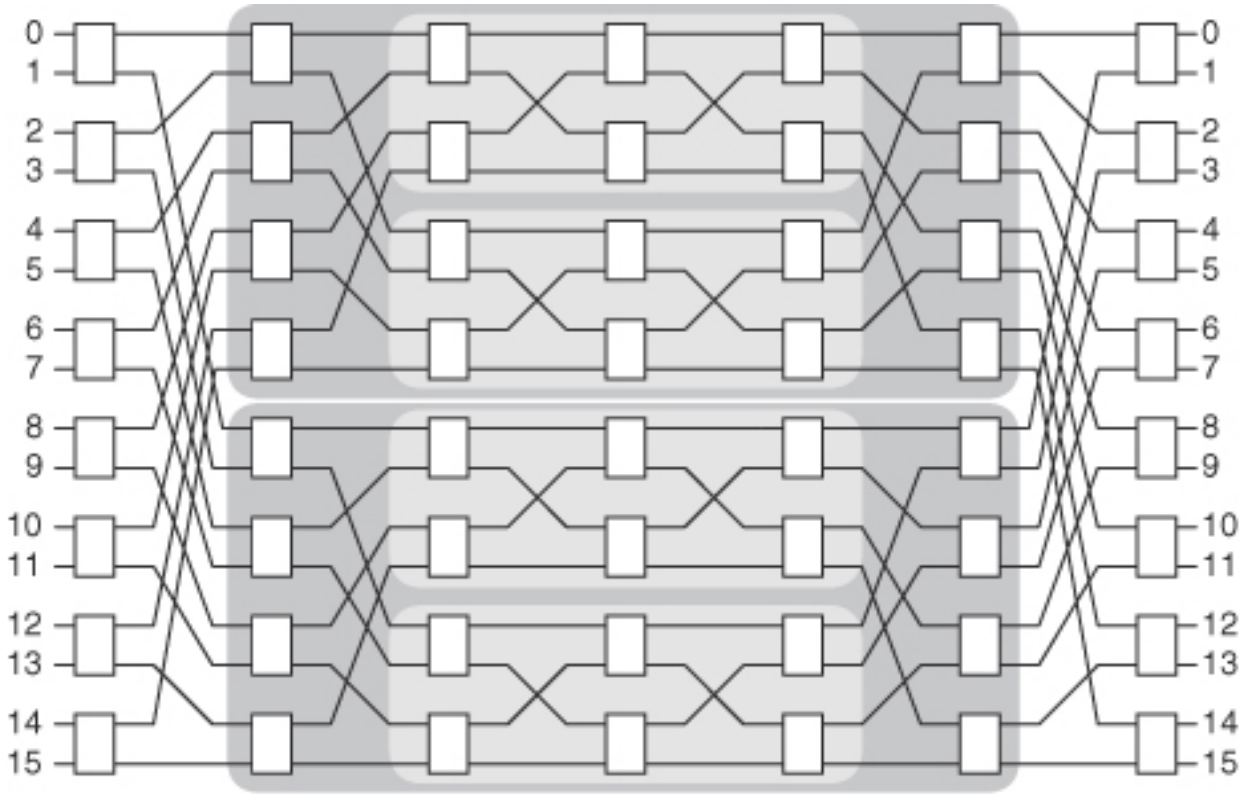


Figure F.12 Beneŝ network. A 16-port Clos topology, where the middle-stage switches shown in the darker shading are implemented with another Clos network whose middle-stage switches shown in the lighter shading are implemented with yet another Clos network, and so on, until a Beneŝ network is produced that uses only 2 x 2 switches everywhere.

2(log N) -1 = 7 stages

total # of paths between a source and a dest = $2^{\lfloor \text{total \# of stages}/2 \rfloor} = 2^3 = 8$

total # of 2x2 switches = $\frac{N}{2}$ * [(2 logN) – 1] = 8*7 = 56

MK book distr-mem — indirect IN (dynamic)

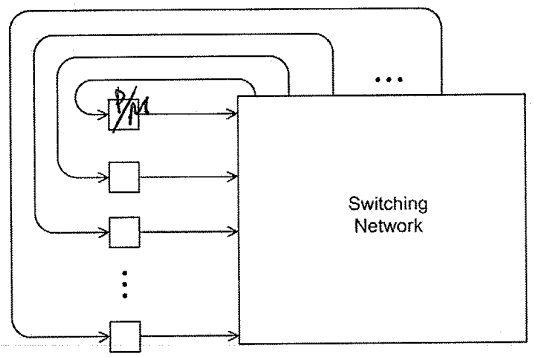


Fig 2.13 A generic indirect network

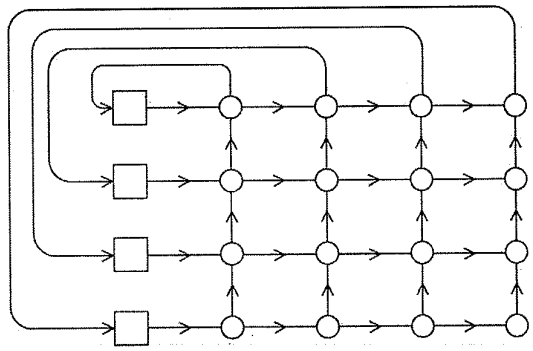


Fig 2.14 A crossbar interconnect for distributed-memory
(4×4) dynamic — single stage

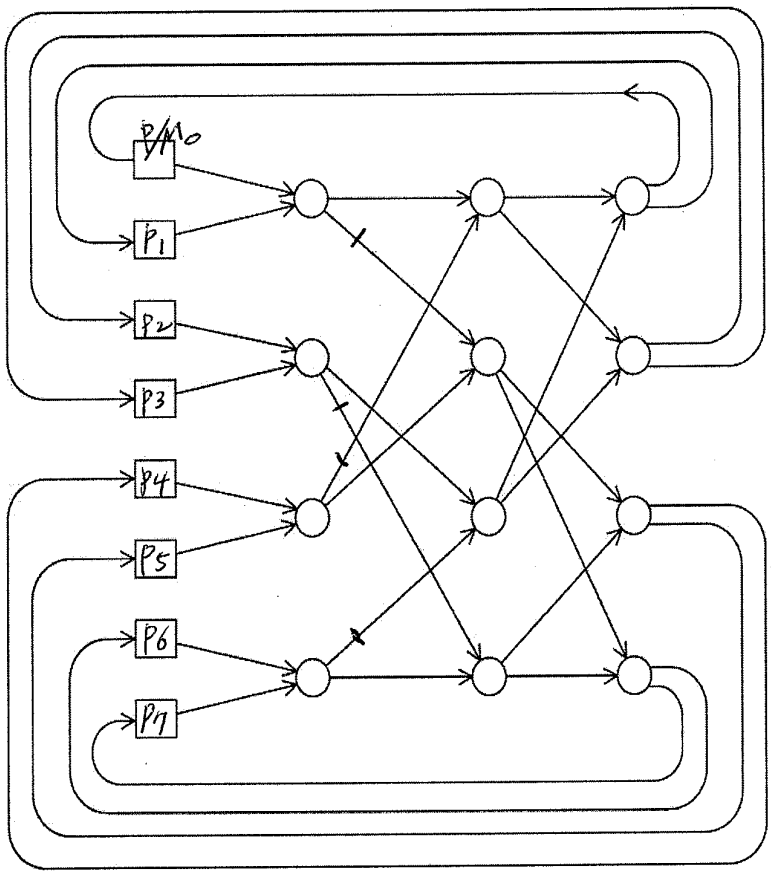


Fig 2.15 An omega network
dynamic — multistage

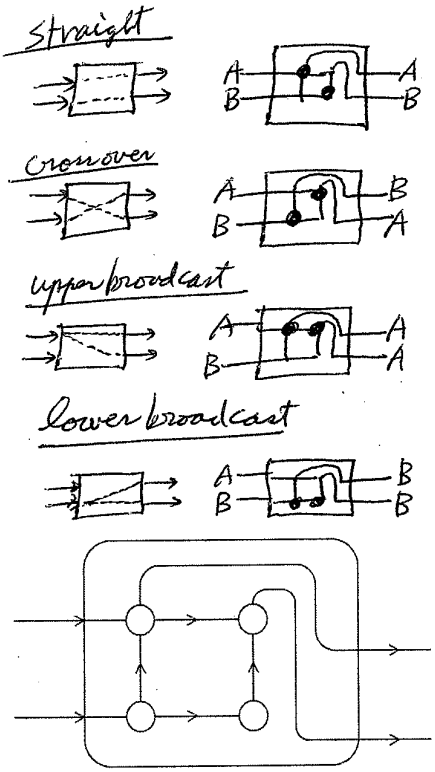


Fig 2.16 A switch in an omega network
(2×2)