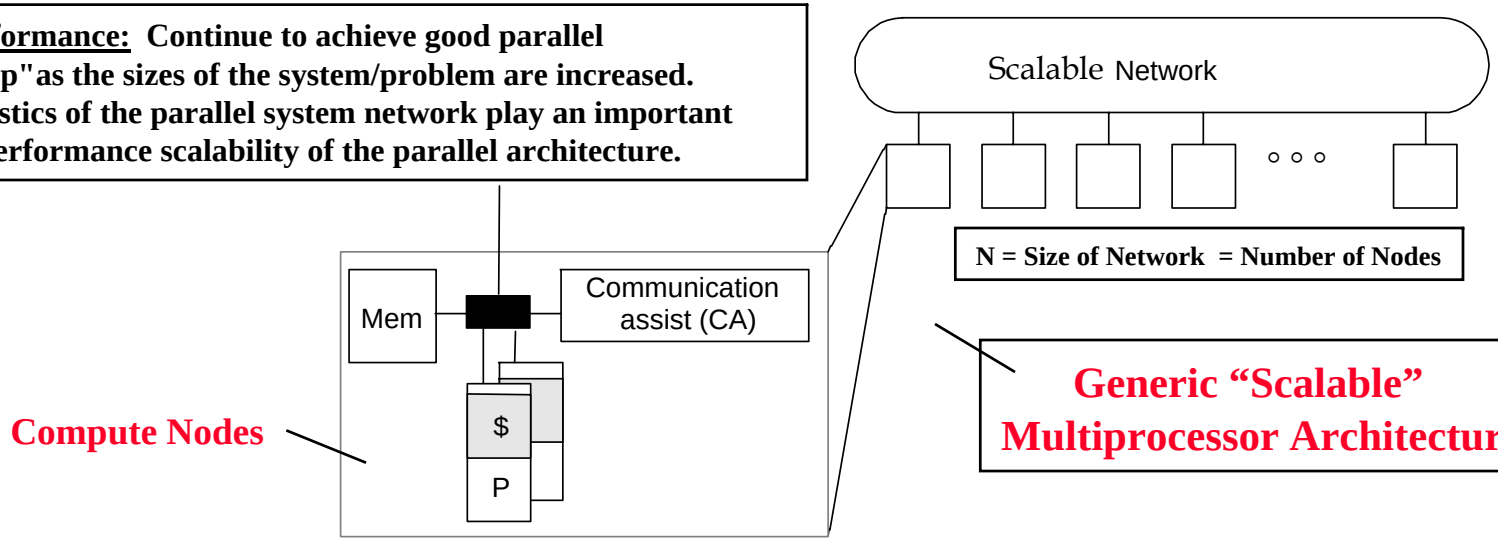
Network Properties, Scalability and Requirements For Parallel Processing

Scalable Parallel Performance: **Continue to achieve good parallel performance "speedup"as the sizes of the system/problem are increased. Scalability/characteristics of the parallel system network play an important role in determining performance scalability of the parallel architecture.**



Node: processor(s), memory system, plus *communication assist:*

- **Network interface and communication controller.**

1 2

Two Aspects of Network Scalability: Performance and Cost/Complexity

- **Scalable network.**
- Function of a parallel machine network is to **efficiently** transfer information from source node to destination node in support of network transactions that realize the programming model.

1 **Network performance should scale up as its size is increased.** i.e network performance scalability

- Latency grows slowly with network size N. e.g $O(\log_2 N)$ vs. $O(N^2)$
- Total available bandwidth scales up with network size. e.g $O(N)$ vs. $O(\log_2 N)$

2 **Network cost/complexity should grow slowly in terms of network size N.**

e.g. $O(N\log_2 N)$ as opposed to $O(N^2)$

i.e network cost/complexity scalability

(PP Chapter 1.3, PCA Chapter 10)

N = Size of Network (Number of Nodes)

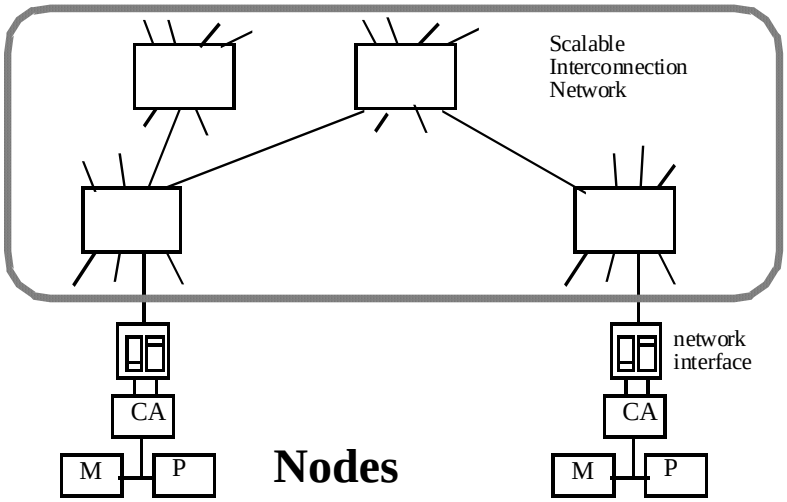
Network Requirements For Parallel Computing

1. Low network latency even when approaching network capacity.
2. High sustained bandwidth that matches or exceeds the communication requirements for given computational rate.
3. High network throughput: Network should support as many concurrent transfers as possible.
4. Low Protocol overhead.
5. Cost/complexity and performance Scalable:
   - Cost/Complexity Scalability: Minimum network cost/complexity increase as network size increases.
     - In terms of number of links/switches, node degree etc.
   - Performance Scalability: Network performance should scale up with network size. - Latency grows slowly with network size.
     - Total available bandwidth scales up with network size.

To Start with: For A given network Size N

To reduce communication overheads, O

As network Size Increases



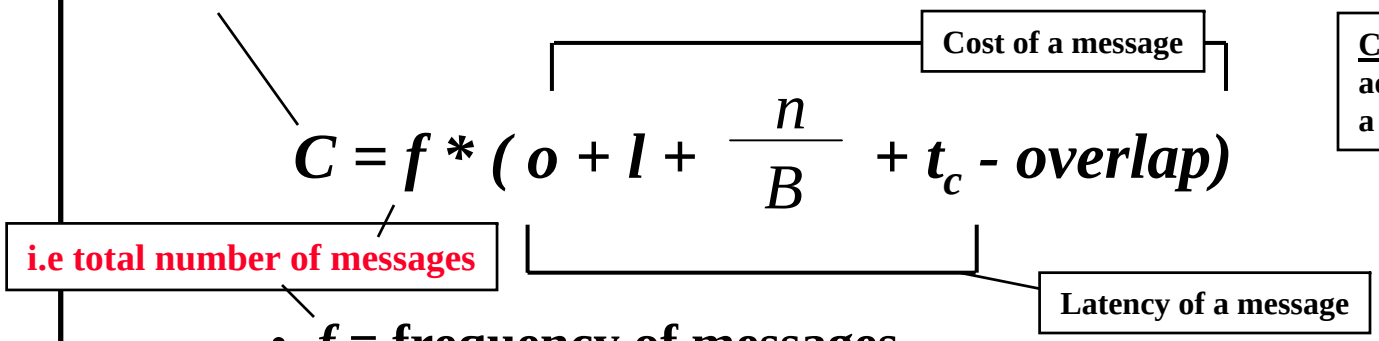
Scalable network

Two Aspects of Network Scalability: Performance and Complexity

Cost of Communication

Given amount of comm (inherent or artifactual), goal is to reduce cost

- **Cost of communication as seen by process:**



Communication Cost: Actual time added to parallel execution time as a result of communication

  - f = frequency of messages
  - o = overhead per message (at both ends)
  - l = network delay per message
  - n = data sent for per message
  - B = bandwidth along path (determined by network, NI, assist)
  - t_c = cost induced by contention per message
  - *overlap* = amount of latency hidden by overlap with comp. or comm.

  – Portion in parentheses is cost of a message (as seen by processor)

  – That portion, ignoring overlap, is *latency* of a message

  ➔ Goal: reduce terms in latency and increase overlap

From lecture 6

Network Representation & Characteristics

- A parallel machine interconnection network is a graph V = {switches or processing nodes} connected by communication channels or links $C \subseteq V \times V$

Routers

- Each channel has width w bits and signaling rate $f = 1/\tau$ (τ is clock cycle time)
  - Channel bandwidth $b = wf$ *bits/sec*

frequency

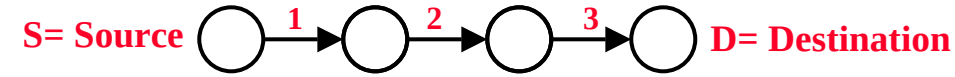
  - Phit (physical unit) data transferred per cycle (usually channel width w).
  - Flit - basic unit of flow-control (minimum data unit transferred across a link).

Flow Unit (frame)

Phit: W W W W W — Flit

i.e Flow Unit or frame or data link layer unit

- Number of channels per node or switch is switch or node *degree*.
- Sequence of switches and links followed by a message in the network is a *route*.
  - Routing Distance: number of links or hops h on route from source to destination.



h = 3 hops in route from S to D

- A network is generally characterized by:
  - Type of interconnection.

Static (point-to-point) or Dynamic

  - Topology.

Network node connectivity/ interconnection structure of the network graph

  - Routing Algorithm.

Deterministic (static) or Adaptive (dynamic)

  - Switching Strategy.

Packet or Circuit Switching

  - Flow Control Mechanism.

Store & Forward (SF) or Cut-Through (CT)

Network Characteristics

- **Type of interconnection:**

1 – **Static, Direct Dedicated (or point-to-point) Interconnects:**

- Compute nodes connected directly using static point-to-point links.
- Such networks include:
  - Fully connected networks , Rings, Meshes, Hypercubes etc.

or channels

2 – **Dynamic or Indirect Interconnects:**

- Switches are usually used to realize dynamic links (paths or virtual circuits) between nodes instead of fixed point-to-point connections.
- Each node is connected to specific subset of switches.
- Dynamic connections are usually established by configuring switches based on communication demands.
- Such networks include: + Wireless Networks ?
  - Shared-, broadcast-, or bus-based connections. (e.g. Ethernet-based).
  - Single-stage Crossbar switch networks. One large switch, Size = NxN
  - Multi-stage Interconnection Networks (MINs) including:
    - Omega Network, Baseline Network, Butterfly Network, etc.

N = Size of Network = Number of Nodes

Network Characteristics

- **Network Topology:** Or Network Graph Connectivity

  Physical interconnection structure of the network graph:

  - **Node connectivity:** Which nodes are directly connected (nodes or switches)
  - **Total number of links needed:** Impacts network cost/total bandwidth
  - **Node Degree:** Number of channels per node. (+ Network Complexity)
  - **Network diameter:** Minimum routing distance in links or hops between the the farthest two nodes .
  - **Average Distance** in hops between all pairs of nodes .
  - **Bisection width:** Minimum number of links whose removal disconnects the network graph and cuts it into approximately two equal halves.
    - Related: **Bisection Bandwidth** = Bisection width x link bandwidth
  - **Symmetry:** The property that the network looks the same from every node.
  - **Homogeneity:** Whether all the nodes and links are identical or not.

Simplify Mapping

Network Topology Properties

Hop = link = channel in route

Network Topology and Requirements for Parallel Processing

1. **For Cost/Complexity Scalability: The total number of links, node degree and size/number of switches used should grow slowly as the size of the network is increased.**
2. **For Low network latency: Small network diameter, average distance are desirable (for a given network size).**
3. **For Latency Scalability: The network diameter, average distance should grow slowly as the size of the network is increased.**
4. **For Bandwidth Scalability: The total number of links should increase in proportion to network size.**
5. **To support as many concurrent transfers as possible (High network throughput): A high bisection width is desirable and should increase proportional to network size.**
   - **Needed to reduce network contention and hot spots.**

More on this later in the lecture

Network Characteristics

- **End-to-End Routing Algorithm and Functions:**
  - The set of paths that messages may follow.

1- **Deterministic (static) Routing:** The route taken by a message determined by source and destination regardless of other traffic in the network.

2- **Adaptive (dynamic) Routing:** One of multiple routes from source to destination selected to account for other traffic to reduce node/link contention.

- **Switching Strategy:**
  - Circuit switching vs. packet switching.
- **Flow Control Mechanism:** Done at/by Data Link Layer?
  - When a message or portions of it moves along its route:
    1. Store & Forward (SF)Routing,
    2. Cut-Through (CT) or Worm-Hole Routing. (usually uses circuit switching)

    AKA pipelined routing
  - What happens when traffic is encountered at a node:
    - Link/Node Contention handling.
    - Deadlock prevention. e.g use buffering
- Broadcast and multicast capabilities.
- Switch routing delay. Δ
- Link bandwidth. b

Network Characteristics

- Hardware/software implementation complexity/cost.
- Network throughput: Total number of messages handled by network per unit time.
- Aggregate Network bandwidth: Similar to network throughput but given in total bytes/sec.
- Network hot spots: Form in a network when a small number of network nodes/links handle a very large percentage of total network traffic and become saturated. → Large Contention Delay t_c
- Network scalability:
  - The feasibility of increasing network size, determined by:
    - Performance scalability: Relationship between network size in terms of number of nodes and the resulting network performance (average latency, aggregate network bandwidth).
    - Cost scalability: Relationship between network size in terms of number of nodes/links and network cost/complexity.

Also number/size of switches for dynamic networks

Communication Network Performance :

Network Latency

Time to transfer n bytes from source to destination:

S = Source
D = Destination

$Time(n)_{s-d}$ = overhead + routing delay — i.e. Network Latency

i.e. no contention delay t_c — O

+ channel occupancy + contention delay

Unloaded Network Latency = routing delay + channel occupancy

channel occupancy = $(n + n_e) / b$

b = channel bandwidth, bytes/sec

n = payload size

n_e = packet envelope: header, trailer. Added to payload

Effective link bandwidth = $bn / (n + n_e)$

~ i.e. transmission time

The term for unloaded network latency is refined next by examining the impact of flow control mechanism used in the network

Next

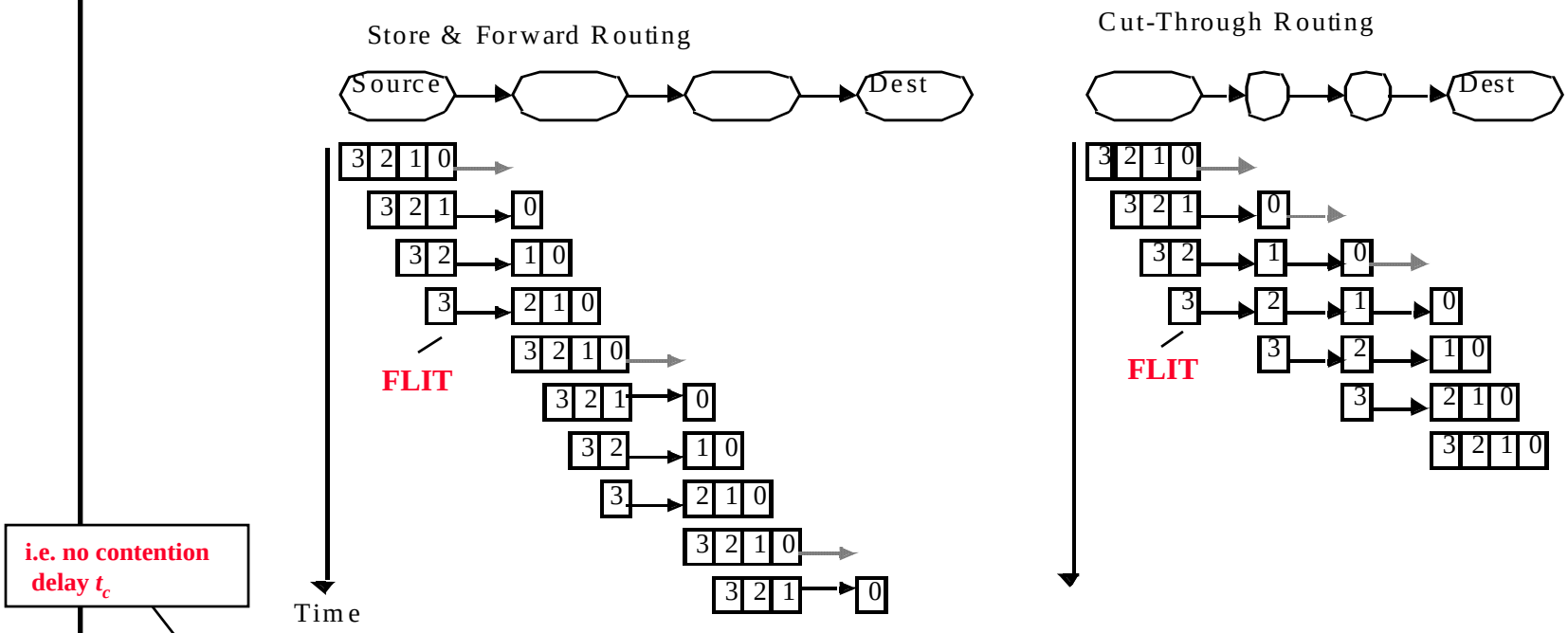
channel occupancy = transmission time

Flow Control Mechanisms:

Usually Done by Data Link Layer

Store&Forward (SF) Vs. Cut-Through (CT) Routing

AKA Worm-Hole or pipelined routing



i.e. no contention delay t_c

Unloaded network latency for n byte packet (message):

$$h(n/b + \Delta) \quad \text{vs} \quad n/b + h\Delta$$

Channel occupancy

Routing delay

h = distance in hops

(number of links in route)

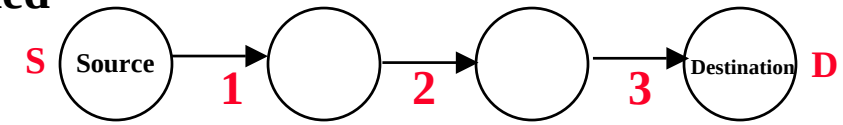
Δ = switch delay

b = link bandwidth **n = size of message in bytes**

Store &Forward (SF) Vs. Cut-Through (CT) Routing Example

Example:

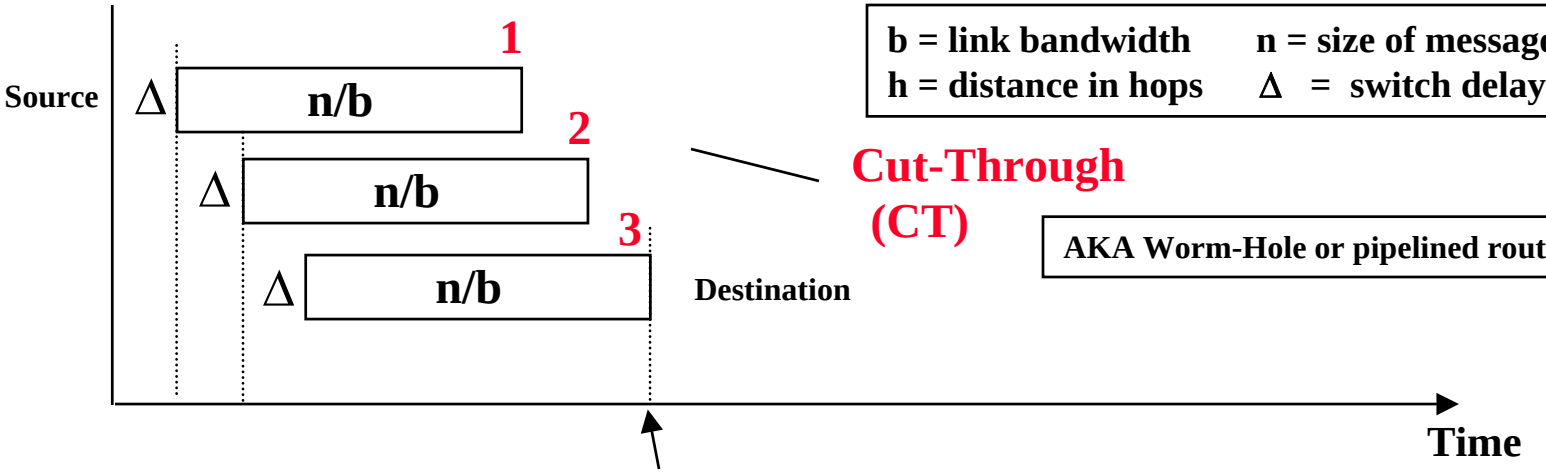
For a route with h = 3 hops or links, unloaded — i.e No contention delay t_c



Route with h = 3 hops from S to D

$T_{sf}(n, h) = h(n/b + \Delta) = 3(n/b + \Delta)$

b = link bandwidth n = size of message in bytes
h = distance in hops Δ = switch delay



AKA Worm-Hole or pipelined routing

$T_{ct}(n, h) = n/b + h\Delta = n/b + 3\Delta$

Channel occupancy

Routing delay

Communication Network Performance :

Refined Unloaded Network Latency Accounting For Flow Control

(i.e no contention, $T_c = 0$) + ignoring overhead term o

- **For an unloaded network (no contention delay) the network latency to transfer an n byte packet (including packet envelope) across the network:**

Unloaded Network Latency = channel occupancy + routing delay

- **For store-and-forward (sf) routing:**

$$\text{Unloaded Network Latency} = T_{sf}(n, h) = h(n/b + \Delta)$$

- **For cut-through (ct) routing:**

$$\text{Unloaded Network Latency} = T_{ct}(n, h) = n/b + h\Delta$$

b = channel bandwidth n = bytes transmitted

h = distance in hops Δ = switch delay

(number of links in route)

channel occupancy = transmission time

Reducing Unloaded* Network Latency

(i.e no contention delay, $T_c = 0$) + ignoring overhead term o

1 • **Use cut-through routing:**

Channel occupancy | Routing delay

– Unloaded Network Latency = $T_{ct}(n, h) = n/b + h\,\Delta$

2 • **Reduce number of links or hops *h in route*:**

how? – **Map communication patterns to network topology** →

e.g. nearest-neighbor on mesh and ring; all-to-all

- Applicable to networks with static or direct point-to-point interconnects: Ideally network topology matches problem communication patterns.

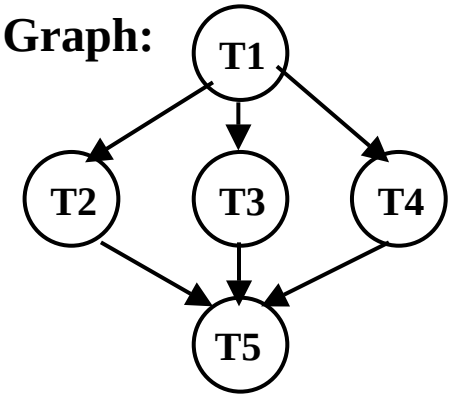
3 • **Increase link bandwidth b.**

4 • **Reduce switch routing delay Δ.**

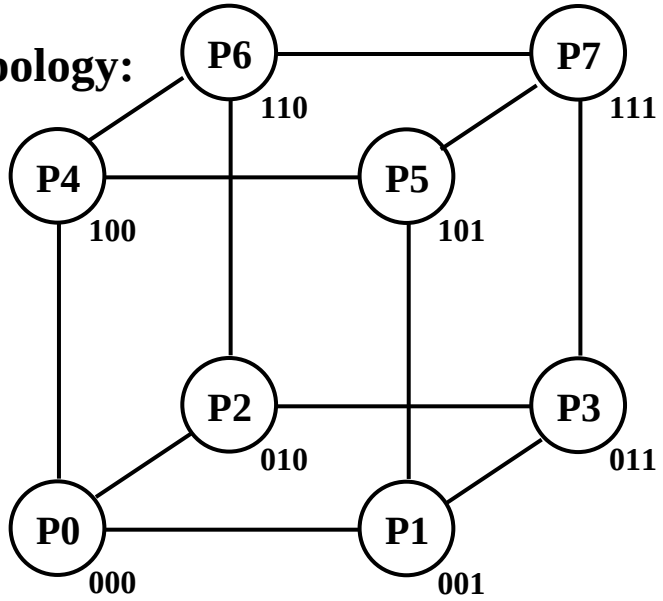
* Unloaded implies no contention delay t_c

Mapping of Task Communication Patterns to Topology Example

Task Graph:



Parallel System Topology:
3D Binary Hypercube



Poor Mapping:

T1 runs on P0
T2 runs on P5
T3 runs on P6
T4 runs on P7
T5 runs on P0

h = 2 or 3

- Communication from T1 to T2 requires 2 hops
  Route: P0-P1-P5
- Communication from T1 to T3 requires 2 hops
  Route: P0-P2-P6
- Communication from T1 to T4 requires 3 hops
  Route: P0-P1-P3-P7
- Communication from T2, T3, T4 to T5
  - similar routes to above reversed (2-3 hops)

Better Mapping:

T1 runs on P0
T2 runs on P1
T3 runs on P2
T4 runs on P4
T5 runs on P0

h = 1

- Communication between any two communicating (dependant) tasks requires just 1 hop

From lecture 6

h = number of hops h in route from source to destination

Available Effective Bandwidth

- **Factors affecting effective local link bandwidth available to a single node:**
  - 1 – Accounting for Packet density: $b \times n/(n + n_e)$
  - 2 – Also Accounting for Routing delay: $b \times n / (n + n_e + w\Delta)$
  - 3 – Contention: t_c
    - At endpoints.
    - Within the network.

n_e = Message Envelope (headers/trailers)

Routing delay

At Communication Assists (CAs)

- **Factors affecting throughput or Aggregate (total) network bandwidth:**
  - 1– Network bisection bandwidth:
    - Sum of bandwidth of smallest set of links when removed partition the network into two unconnected networks of equal size.
  - 2– Total bandwidth of all the C channels: Cb bytes/sec, Cw bits per cycle or C phits per cycle.

Example – Suppose N hosts each issue a message every M cycles with average routing distance h and average distribution: i.e uniform distribution over all channels

of size n bytes

- Each message occupies h channels for $\ell = n/w$ cycles
- Total network load = $Nh\ell / M$ phits per cycle.
- Average Link utilization = Total network load / Total bandwidth

C phits

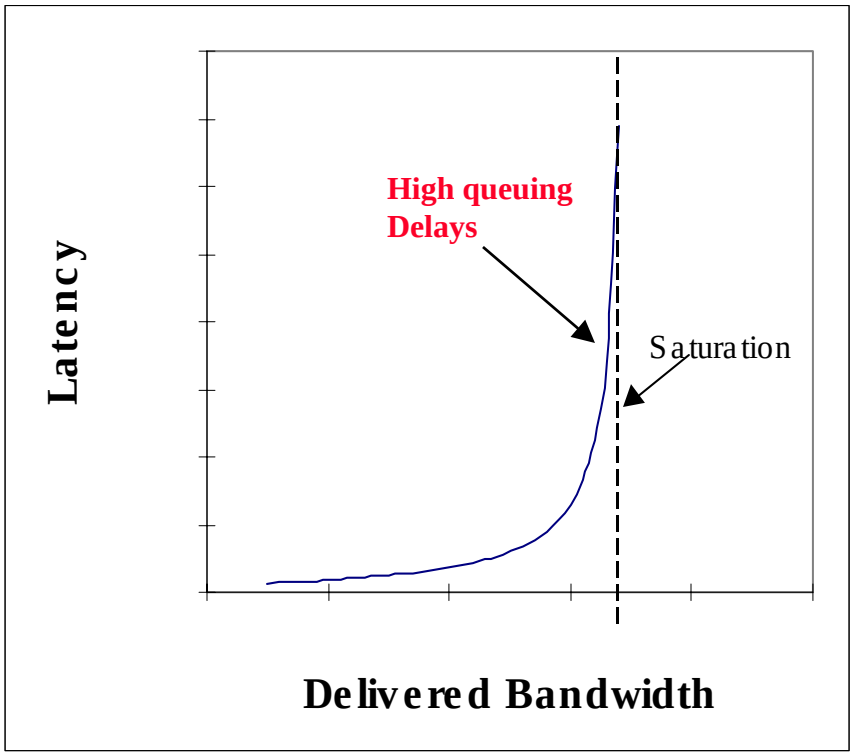
Should be less than 1

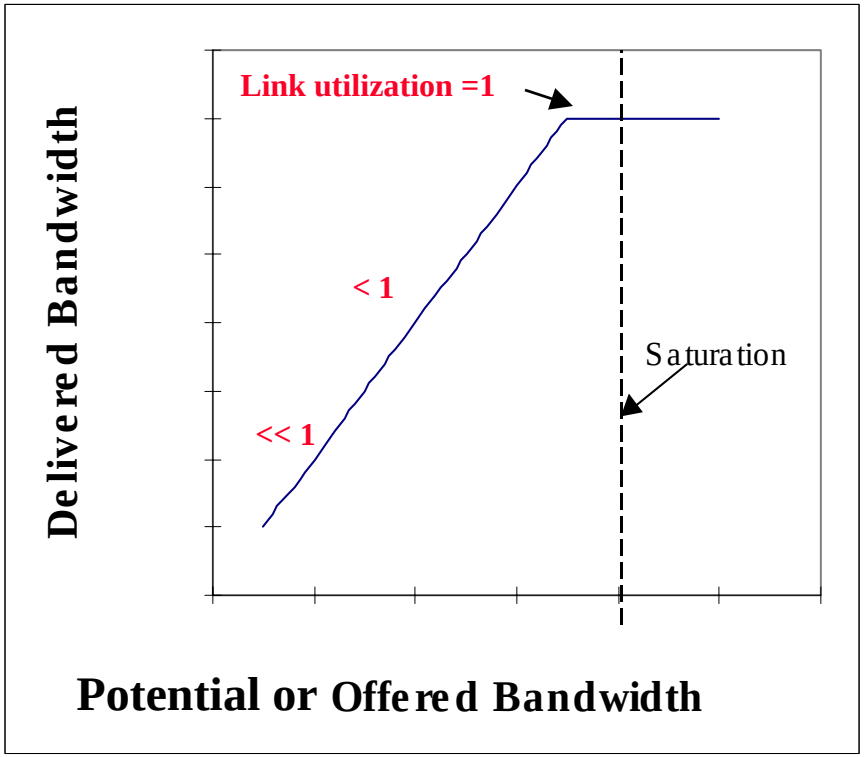
- Average Link utilization: $\rho = Nh\ell /MC < 1$

Phit = w = channel width in bits
b = channel bandwidth
n = message size

Note: equation 10.6 page 762 in the textbook is incorrect

Network Saturation

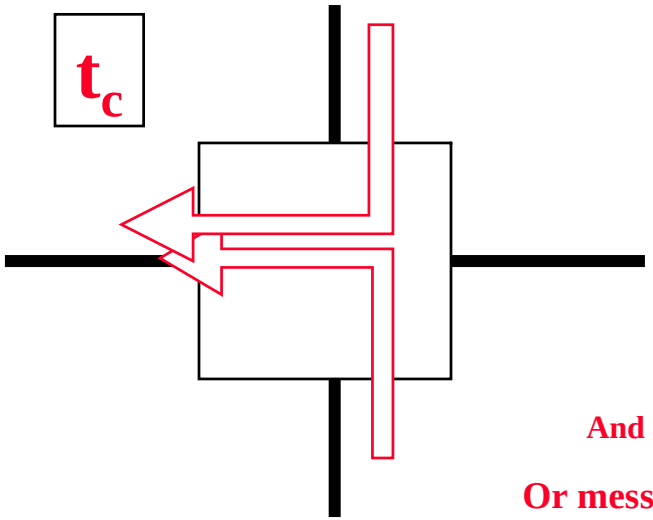




Indications of Network Saturation

→ **Large Contention Delay t_c**

Network Performance Factors: Contention



Network Hot Spots

Network hot spots:
Form in a network when a small number of network nodes/links handle a very large percentage of total network traffic and become saturated.

Caused by communication load imbalance creating a high level of contention at these few nodes/links.

And Low Network Bisection Width?

Or messages

- **Contention: Several packets trying to use the same link/node at same time.**
  - **May be caused by limited available buffering.**
  - **Possible resolutions/prevention:**
    - **Drop one or more packets (once contention occurs).** — i.e to resolve contention
    - **Increased buffer space.**
    - **Use an alternative route (requires an adaptive routing algorithm or a better static routing to distribute load more evenly).** — i.e. Dynamic
    - **Use a network with better bisection width (more routes).**

To Prevent:

Example Next

→ *Reduces hot spots and contention*

- **Most networks used in parallel machines block in place:**
  - **Link-level flow control.**
  - **Back pressure to the source to slow down flow of data.**

Causes contention delay t_c

Slow Down!

Reducing node/link contention:

AKA Dynamic

Deterministic Routing vs. Adaptive Routing

AKA Static

Example: Routing in 2D Mesh

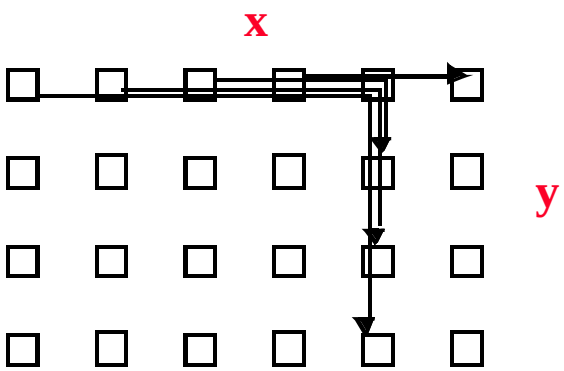
1 **Deterministic (static) Dimension Order Routing in 2D mesh:** Each packet carries signed distance to travel in each dimension [Δx, Δy]. First move message along x then along y.

e.g. x then y (always)

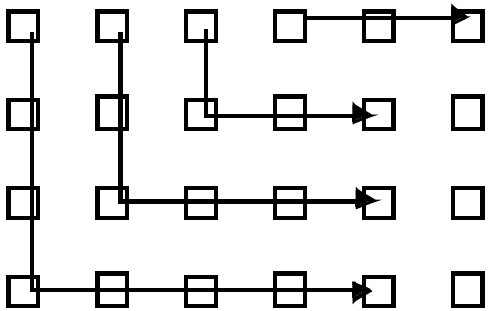
2 **Adaptive (dynamic) Routing in 2D mesh:** Choose route along x, y dimensions according to link/node traffic to reduce node/link contention.

- More complex to implement.

X then Y (always)



Y then X ?



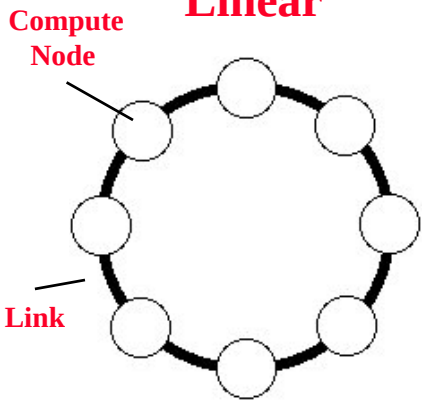
1 **Deterministic Dimension Routing along x then along y (node/link contention)**

2 **Adaptive (dynamic) Routing (reduced node/link contention)**

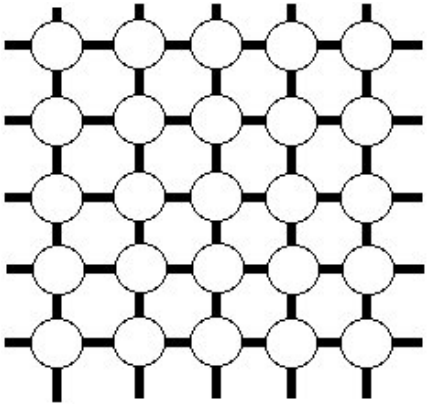
Sample Static Network Topologies

(Static or point-to-point)

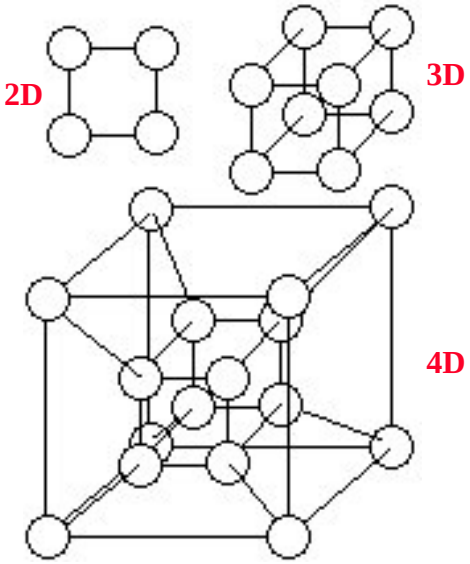
Linear



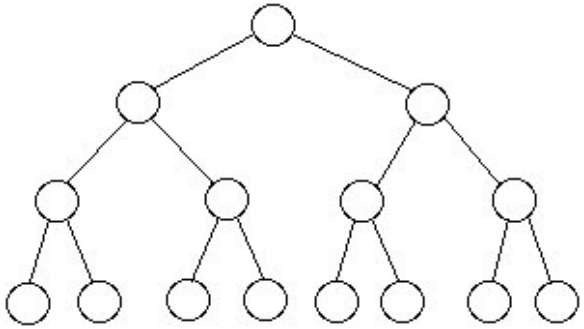
Ring



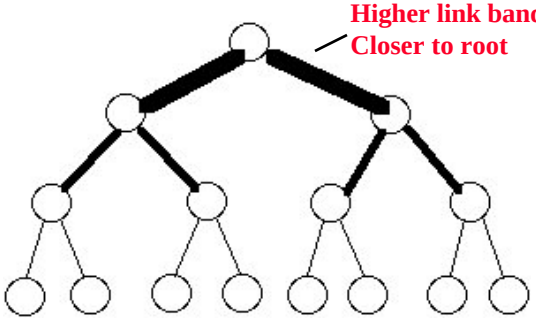
2D Mesh



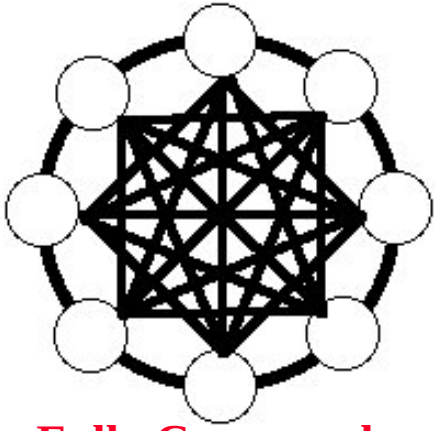
Hybercube



Binary Tree



Fat Binary Tree



Fully Connected

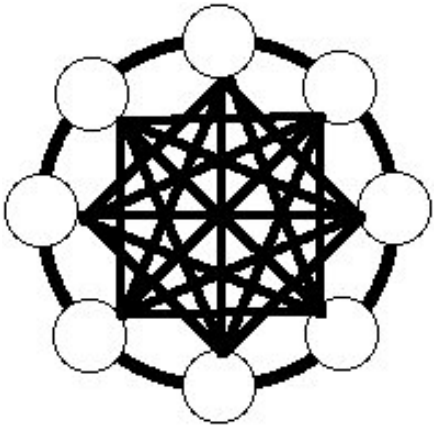
Static Point-to-point Connection Network Topologies

- Direct point-to-point links are used.
- Suitable for predictable communication patterns matching topology.

Match network graph (topology) to task graph

Fully Connected Network: Every node is connected to all other nodes using N- 1 direct links

N(N-1)/2 Links -> $O(N^2)$ complexity
Node Degree: N -1
Diameter = 1
Average Distance = 1
Bisection Width = $(N/2)^2$



Linear Array:

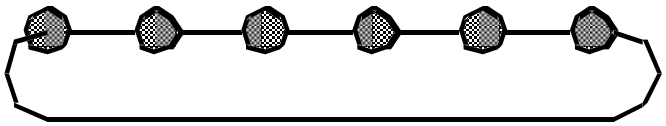
AKA 1D Mesh

N-1 Links -> O(N) complexity
Node Degree: 1-2
Diameter = N -1
Average Distance = 2/3N
Bisection Width = 1

Route A -> B given by
relative address R = B-A

Ring:

AKA 1D Torus
Or Cube



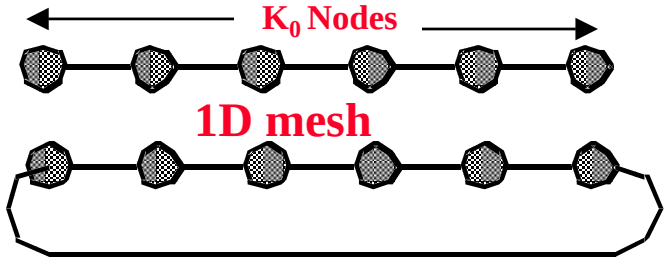
N Links -> O(N) complexity
Node Degree: 2
Diameter = N/2
Average Distance = 1/3N
Bisection Width = 2

Examples: Token-Ring, FDDI, SCI (Dolphin interconnects SAN), FiberChannel Arbitrated Loop, KSR1

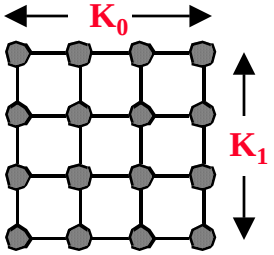
N = Number of nodes

Static Network Topologies Examples: Multidimensional Meshes and Tori

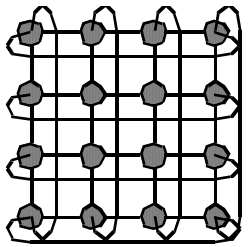
Toruses?



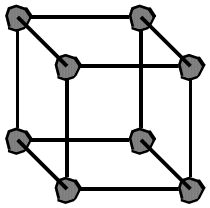
1D torus



4x4 2D mesh



4x4 2D torus



3D binary cube
(AKA 2-ary cube or Torus)

Mesh ***d*-dimensional array or mesh:**

k_j nodes in each of d dimensions

k_j may not be equal in each dimension

- $N = k_{d-1} X ... X k_0$ nodes

A node is connected to nodes that differ by one in every dimension

- Described by *d*-vector of coordinates $(i_{d-1}, ..., i_0)$
- Where $0 \leq i_j \leq k_j - 1$ for $0 \leq j \leq d-1$

N = Number of nodes

Mesh ***d*-dimensional *k*-ary mesh:** $N = k^d$

A mesh with k nodes in each of d dimensions

- $k = \sqrt[d]{N}$ or $N = k^d$
- Described by *d*-vector of radix *k* coordinate.
- Diameter = d(k-1)

Torus ***d*-dimensional *k*-ary torus (or *k*-ary *d*-cube):**

A Torus with k nodes in each of d dimensions

Mesh + Edges wrap around, every node has degree 2d and connected to nodes that differ by one (mod k) in every dimension.

N = Total number of nodes

Properties of d-dimensional k-ary Meshes and Tori (*k*-ary *d*-cubes)

k nodes in each of d dimensions

Routing: *Deterministic or static Routing*

- *Dimension-order routing (both).*
  - Relative distance: $R = (b_{d-1} - a_{d-1}, \ldots, b_0 - a_0)$
  - Traverse $r_i = b_i - a_i$ hops in each dimension.

a = Source Node
b = Destination Node

Diameter:

- d(k-1) for mesh
- $d \lfloor k/2 \rfloor$ for cube or torus

For k = 2 Diameter = d (for both)

Average Distance:

- d x $2k/3$ for mesh.
- $dk/3$ for cube or torus.

Number of Nodes:

- $N = k^d$ for both

Number of Links:

- $dN - dk$ for mesh
- $dN = d\,k^d$ for cube or torus

(More links due to wrap-around links)

Node Degree:

- d to $2d$ for mesh.
- $2d$ for cube or torus.

Bisection width:

- k^{d-1} links for mesh.
- $2k^{d-1}$ links for cube or torus.

N = Number of nodes

Static (point-to-point) Connection Networks Examples:

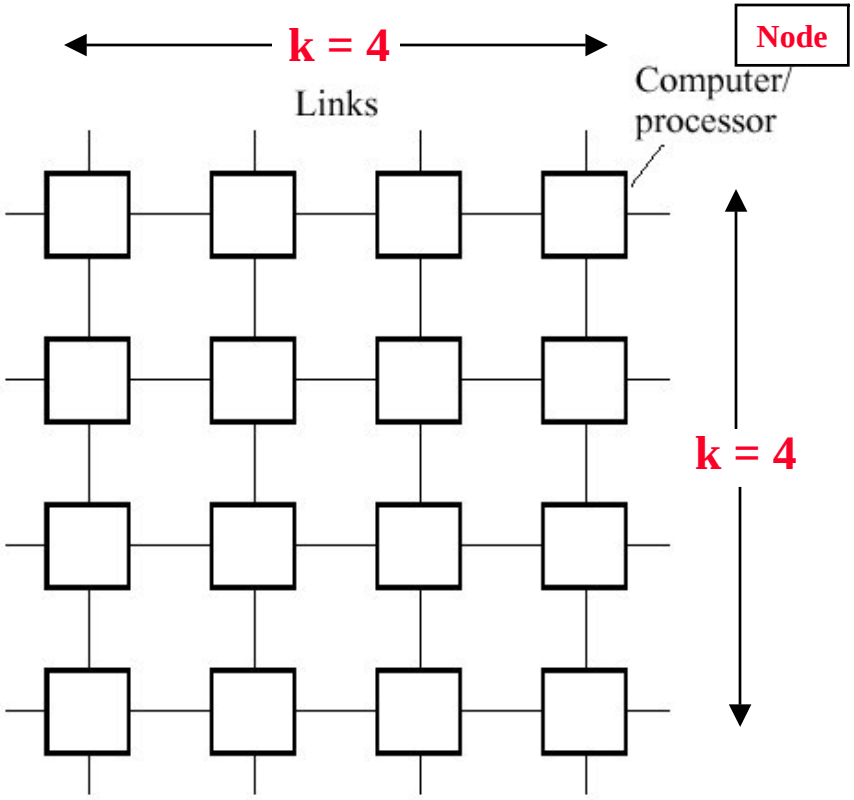
2D Mesh

(*2*-dimensional *k*-ary mesh)

For an *k* x *k* 2D Mesh:

- **Number of nodes N = k^2**
- **Node Degree: 2-4**
- **Network diameter: 2(*k*-1)**
- **No of links: 2N - 2*k***
- **Bisection Width: *k***
- **Where *k* = $\sqrt{N}$**

K = 4 nodes in each dimension



Here k = 4 N = 16
Diameter = 2(4-1) = 6
Number of links = 32 -8 = 24
Bisection width = 4

How to transform 2D mesh into a 2D torus?

Static Connection Networks Examples

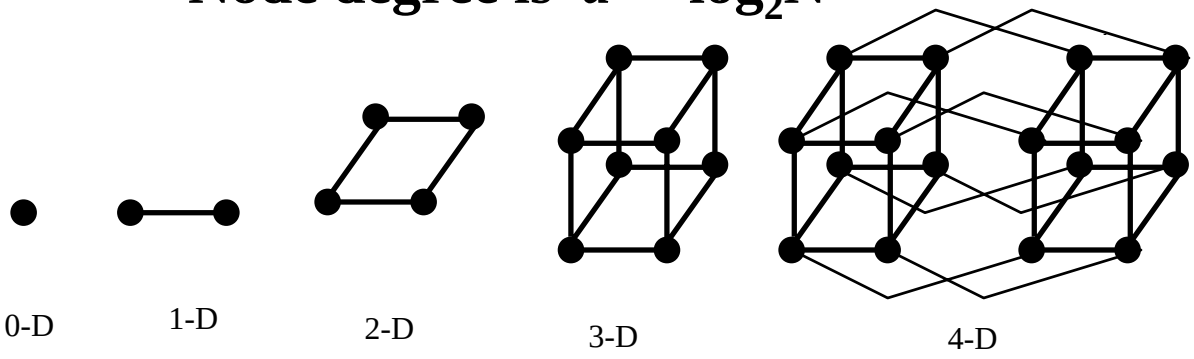
Hypercubes

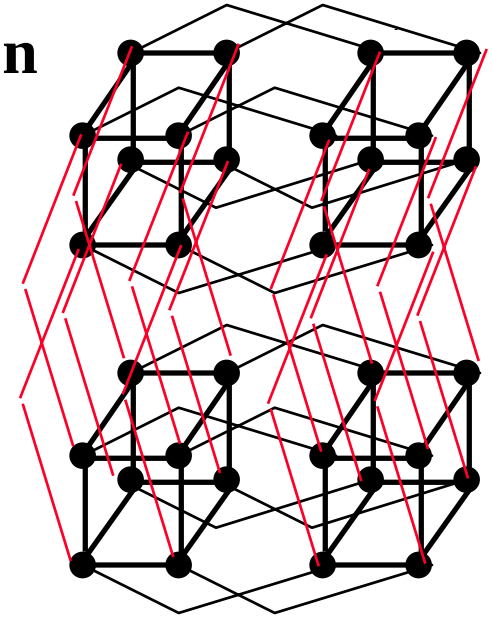
***k*-ary *d*-cubes or tori with k =2 in each dimension**

Or: Binary d-cube
2-ary d-torus
Binary d-torus
Binary d-mesh
2-ary d-mesh?

- **Also called binary *d*-cubes (*2*-ary *d*-cube)**
- **Dimension = $d = \log_2 N$**
- **Number of nodes = $N = 2^d$**
- **Diameter: $O(\log_2 N)$ hops = d= Dimension**
- **Good bisection width: N/2**
- **Complexity:**
  - **Number of links: $N(\log_2 N)/2$**
  - **Node degree is $d = \log_2 N$**

O(N Log$_2$ N)

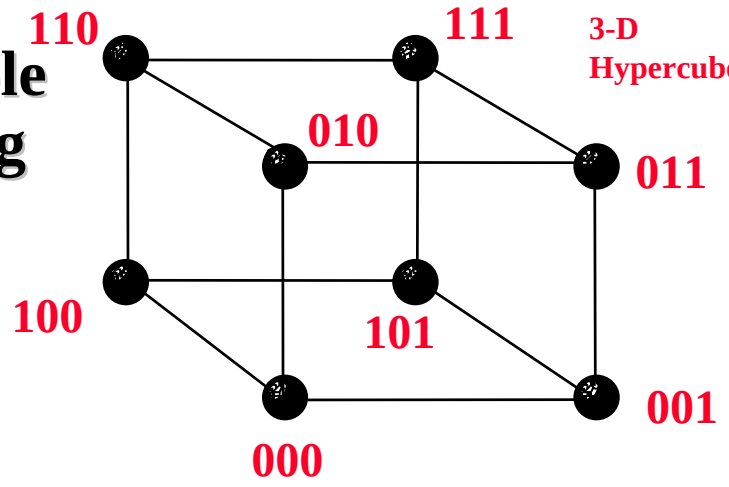




Connectivity: A node is directly connected to d nodes with addresses that differ from its address in only one bit

3-D Hypercube Static Routing Example

Message Routing Functions Example
Dimension-order (E-Cube) Routing



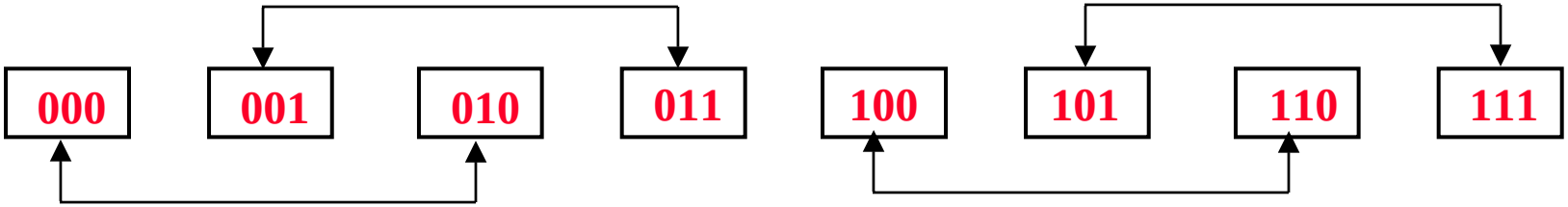
Network Topology:
3-dimensional static-link hypercube
Nodes denoted by $C_2C_1C_0$

Routing by least significant bit C_0

1st Dimension

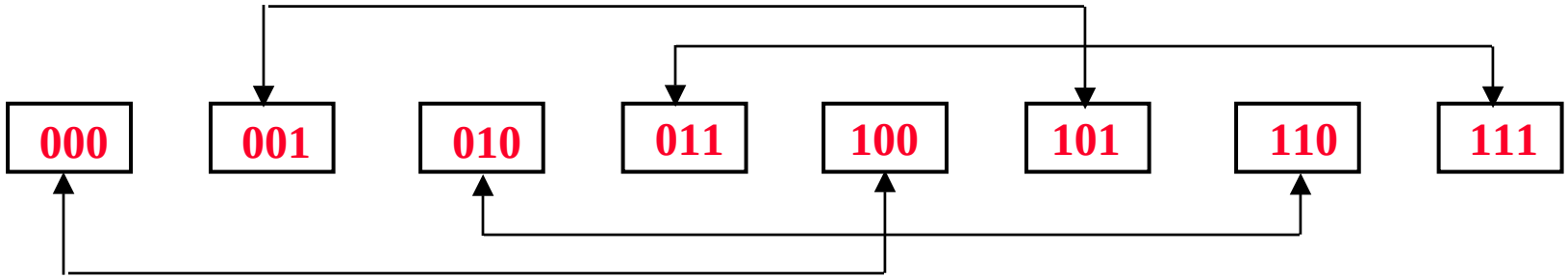
Routing by middle bit C_1

2nd Dimension



Routing by most significant bit C_2

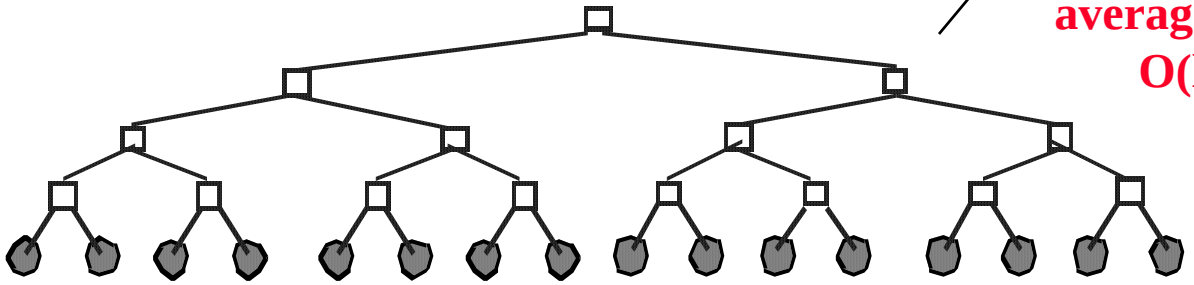
3rd Dimension



For Hypercubes: Diameter = max hops = d here d =3

Static Connection Networks Examples:

Trees



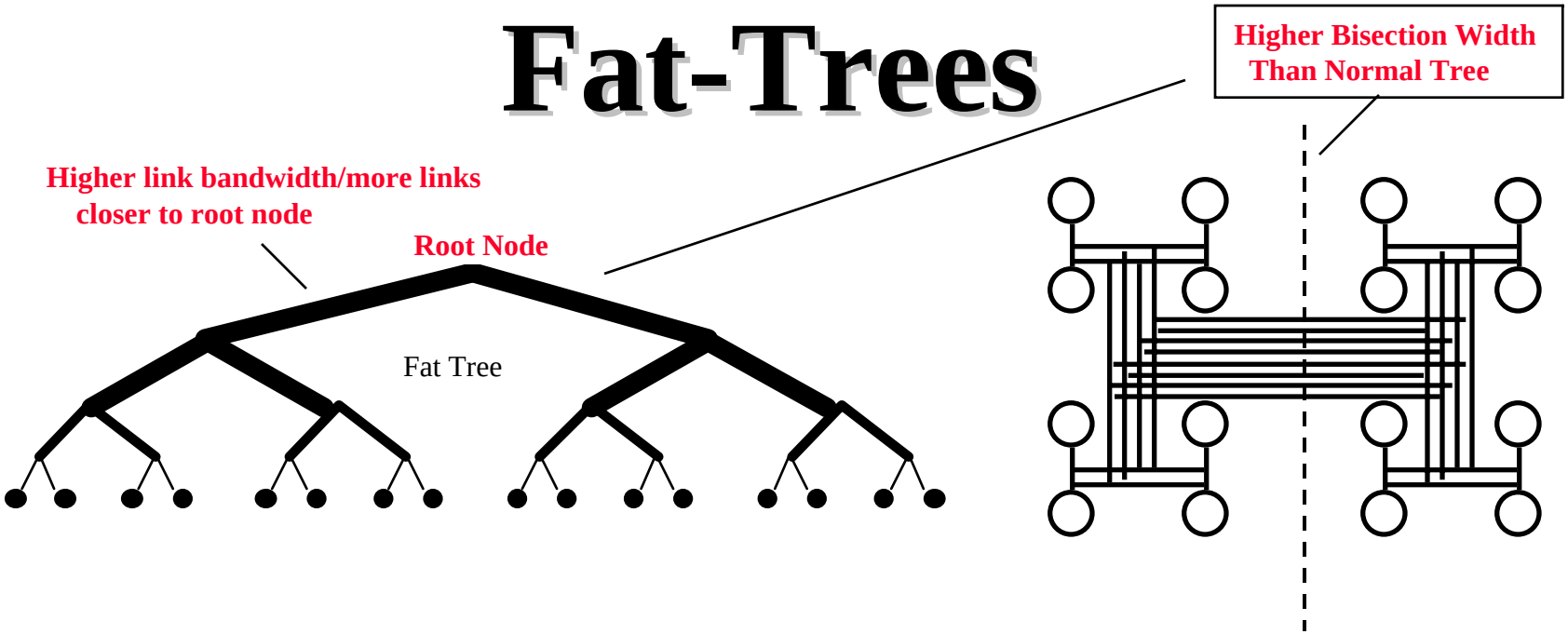
Binary Tree k=2
Height/diameter/
average distance:
$O(\log_2 N)$

- Diameter and average distance are logarithmic.
  - *k*-ary tree, height $d = \log_k N$
  - Address specified d-vector of radix k coordinates describing path down from root.
- Fixed degree *k*. (Not for leaves, for leaves degree = 1)
- Route up to common ancestor and down:
  - R = B XOR A
  - Let *i* be position of most significant 1 in R, route up *i*+1 levels
  - Down in direction given by low *i*+1 bits of B
- H-tree space is O(N) with O(√N) long wires.
- Low Bisection Width = 1

Good? Or Bad?

Static Connection Networks Examples:

Fat-Trees



- **"Fatter" higher bandwidth links (more connections in reality) as you go up, so bisection bandwidth scales with number of nodes N.**

  Why? To fix low bisection width problem in normal tree topology

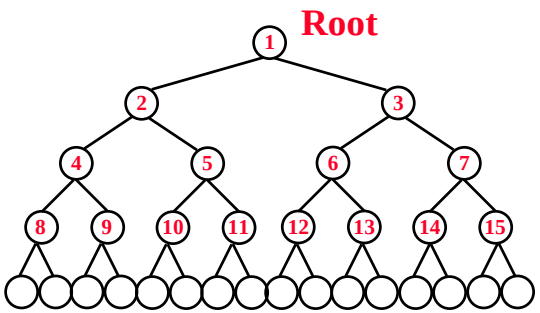
- **Example: Network topology used in Thinking Machine CM-5**

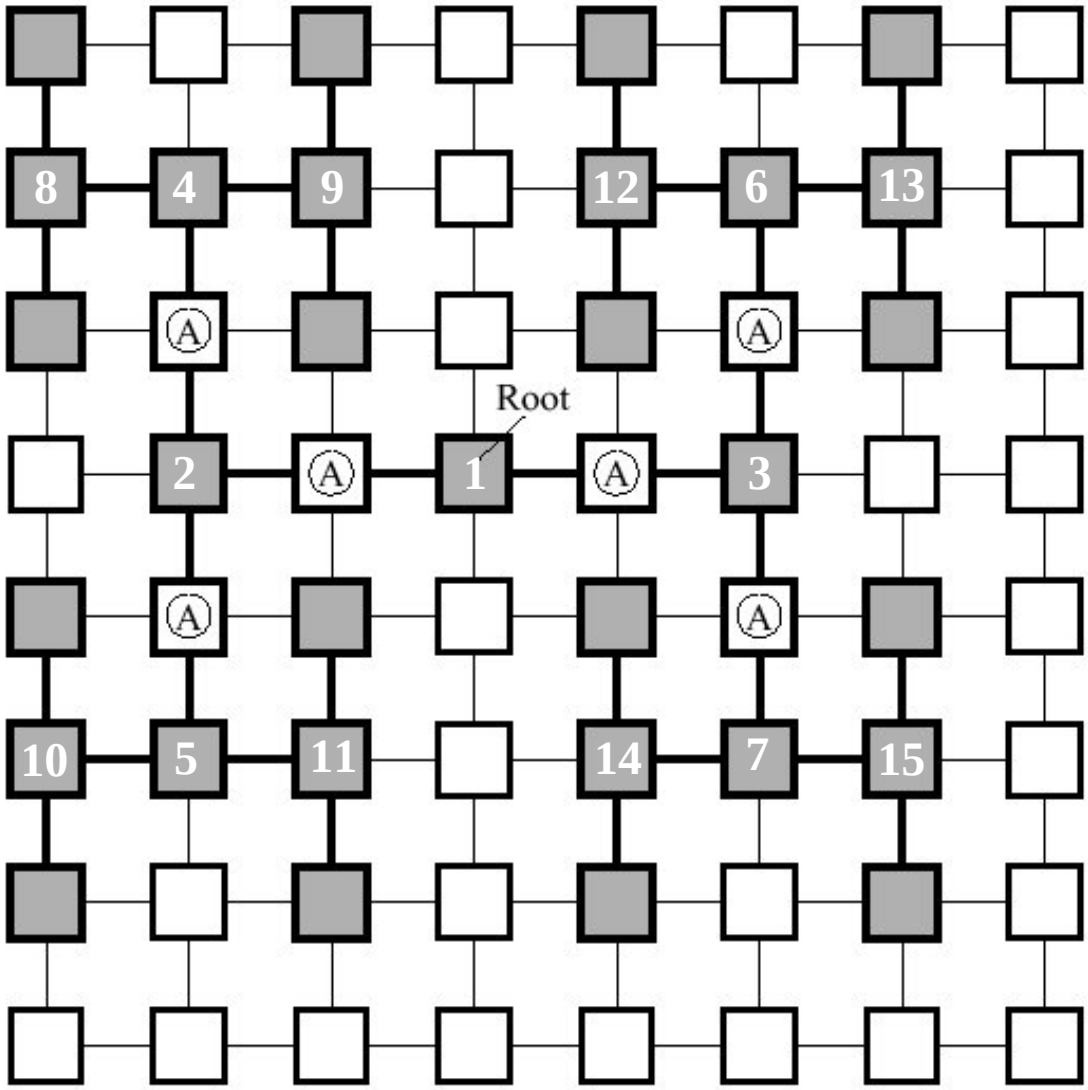
Embedding A Binary Tree Onto A 2D Mesh

Embedding:
In static networks refers to mapping nodes of one network (or task graph?) onto another network while attempting to minimize extra hops.

Graph Matching?

H-Tree Configuration to embed binary tree onto a 2D mesh





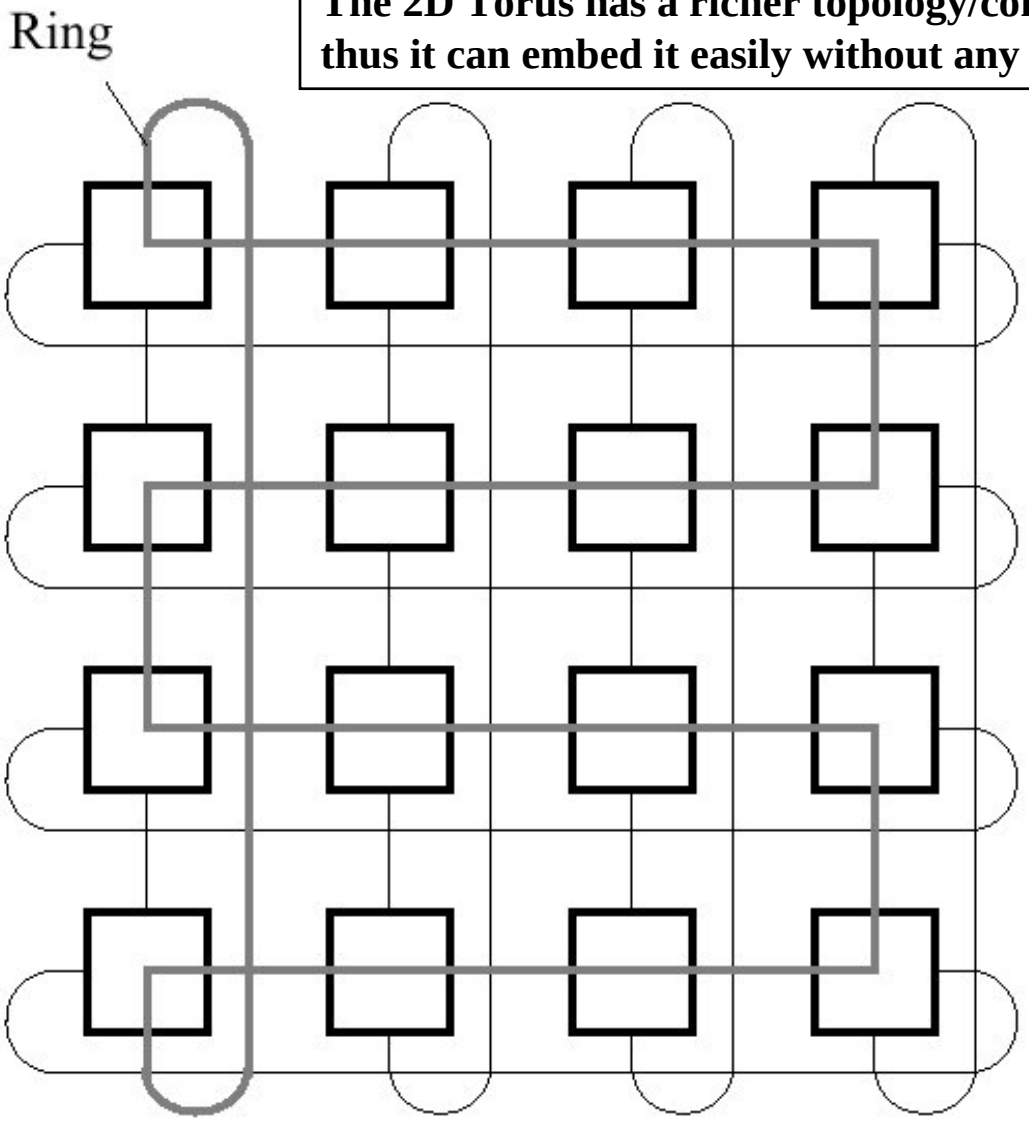
Ⓐ = Additional nodes added to form the tree

i.e Extra hops

(PP, Chapter 1.3.2)

Embedding A Ring Onto A 2D Torus

The 2D Torus has a richer topology/connectivity than a ring, thus it can embed it easily without any extra hops needed



Ring:

Node Degree = 2
Diameter = ⌊N/2⌋
Links = N
Bisection = 2

Here
N = 16
Diameter = 8
Links = 16

2D Torus:

Node Degree = 4
Diameter = 2⌊k/2⌋
Links = 2N = 2 k^2
Bisection = 2k

Here k = 4
Diameter = 4
Links = 32
Bisection = 8

Extra Hops Needed?

Also: Embedding a binary tree onto a Hypercube is done without any extra hops

Dynamic Connection Networks

- **Switches are usually used to dynamically implement connection paths or virtual circuits between nodes instead of fixed point-to-point connections.**
- **Dynamic connections are established by configuring switches based on communication demands.**
- **Such networks include:**

1 – Bus systems. Shared links/interconnects

e.g. Wireless Networks?

2 – Multi-stage Interconnection Networks (MINs):

- **Omega Network.**
- **Baseline Network**
- **Butterfly Network, etc.**

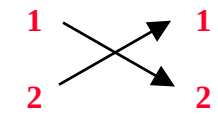
3 – Single-stage Crossbar switch networks.

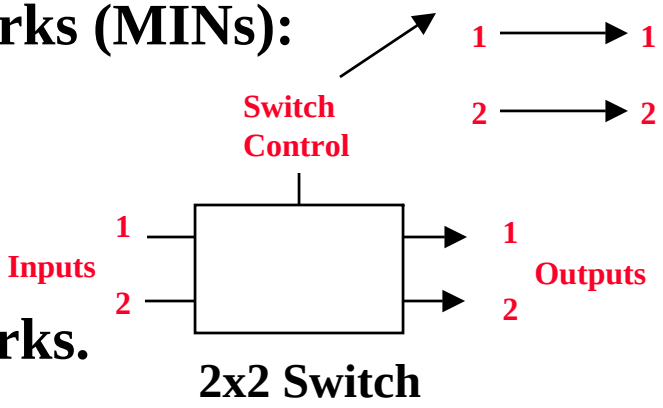
(one N x N large switch)

$O(N_2)$ Complexity?

N = Size of Network = Number of Nodes

e.g





2x2 Switch

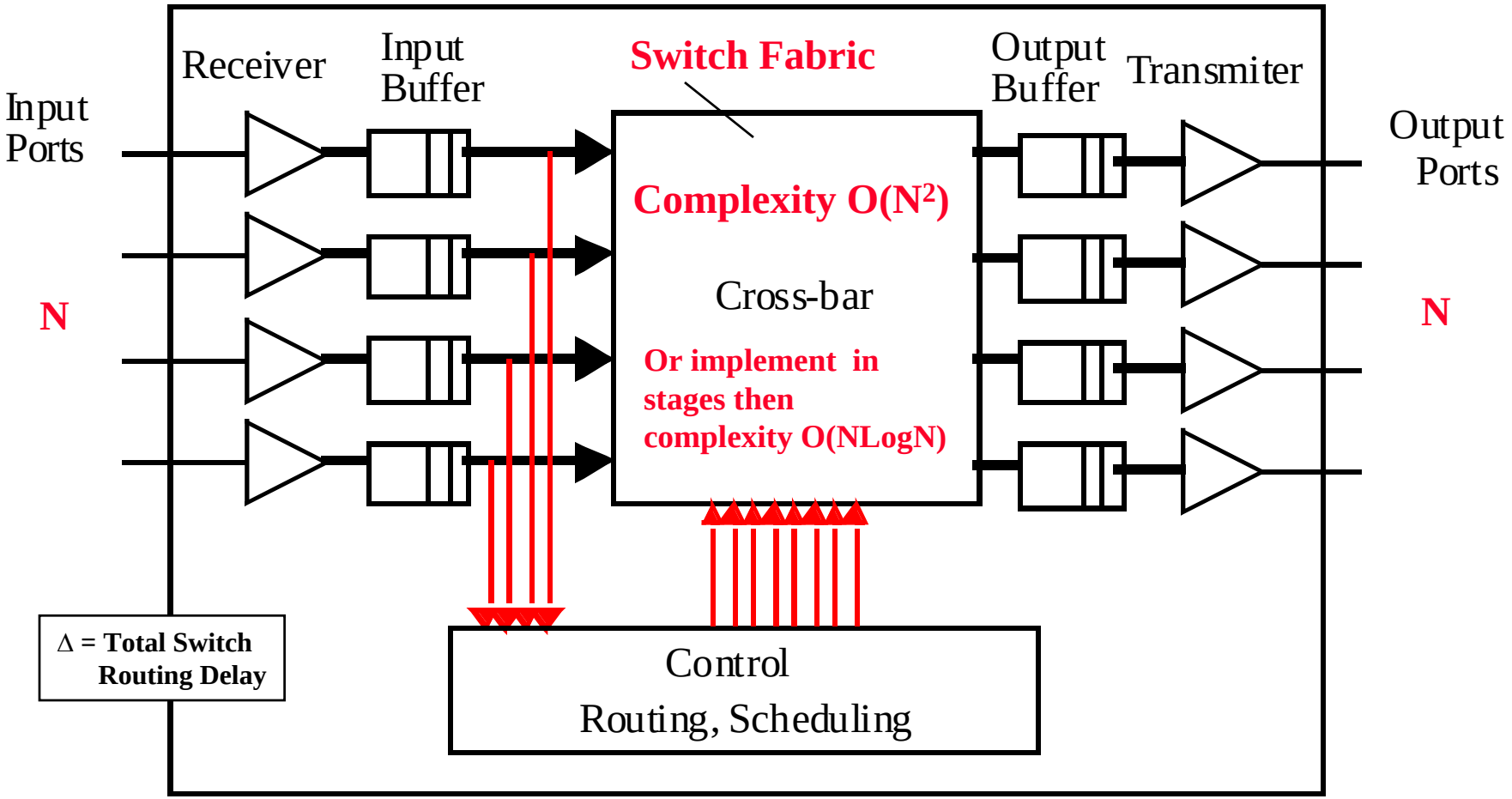
A possible MINS Building Block

Dynamic Networks Definitions

- **Permutation networks:** Can provide any one-to-one mapping between sources and destinations.
- **Strictly non-blocking:** Any attempt to create a valid connection succeeds. These include Clos networks and the crossbar.
- **Wide Sense non-blocking:** In these networks any connection succeeds if a careful routing algorithm is followed. The Benes network is the prime example of this class.
- **Rearrangeably non-blocking:** Any attempt to create a valid connection eventually succeeds, but some existing links may need to be rerouted to accommodate the new connection. Batcher's bitonic sorting network is one example.
- **Blocking:** Once certain connections are established it may be impossible to create other specific connections. The Banyan and Omega networks are examples of this class.
- **Single-Stage networks:** Crossbar switches are single-stage, strictly non-blocking, and can implement not only the N! permutations, but also the N^N combinations of non-overlapping broadcast.

Dynamic Network Building Blocks:

Crossbar-Based NxN Switches



Implemented using one large N x N switch or by using multiple stages of smaller switches

Switch Components

- **Output ports:**
  - Transmitter (typically drives clock and data).
- **Input ports:**
  - Synchronizer aligns data signal with local clock domain.
  - FIFO buffer.
- **Crossbar:** i.e switch fabric
  - Switch fabric connecting each input to any output.
  - Feasible degree limited by area or pinout, $O(n^2)$ complexity.
- **Buffering** (input and/or output).
- **Control logic:**
  - Complexity depends on routing logic and scheduling algorithm.
  - Determine output port for each incoming packet.
  - Arbitrate among inputs directed at same output.
  - May support quality of service constraints/priority routing.

for n x n crossbar

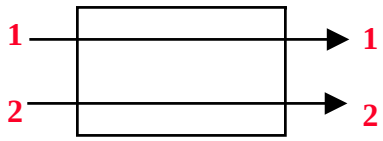
Switch Size And Legitimate States

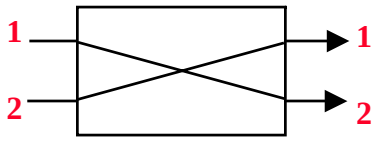
Switch Size	All Legitimate States (includes broadcasts)	Permutation Connections (i.e only one-to-one mappings no broadcast connections)
2 X 2	$2^2 = 4$	$2! = 2$
4 X 4	$4^4 = 256$	$4! = 24$
8 X 8	$8^8 = 16,777,216$	$8! = 40,320$
n X n	n^n	$n!$

Input size

Output size

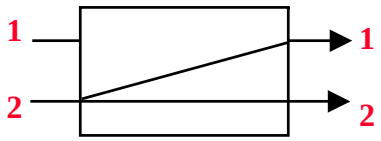
Example: Four states for 2x2 switch

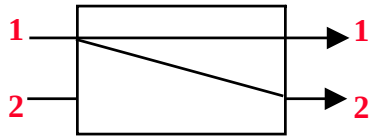




(2 permutation connections)

(2 broadcast connections)





For n x n switch: Complexity = $O(n^2)$ n= number of input or outputs

Permutations

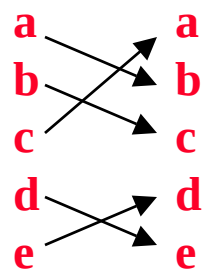
AKA Bijections (one to one mappings)

- For n objects there are $n!$ permutations by which the n objects can be reordered.
- The set of all permutations form a permutation group with respect to a composition operation.
- One can use cycle notation to specify a permutation function.

For Example:

One Cycle

The permutation $\pi = (a, b, c)(d, e)$



stands for the bijection (one to one) mapping:

$$a \rightarrow b, \quad b \rightarrow c \ , \ c \rightarrow a \ , \ d \rightarrow e \ , \quad e \rightarrow d$$

in a circular fashion.

The cycle (a, b, c) has a period of 3 and the cycle (d, e) has a period of 2. Combining the two cycles, the permutation π has a cycle period of $2 \times 3 = 6$. If one applies the permutation π six times, the identity mapping

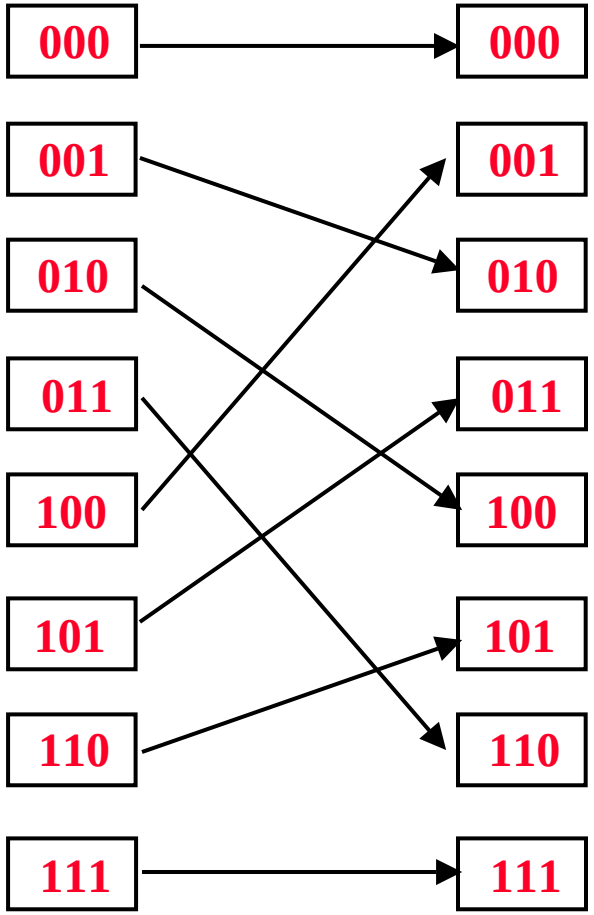
$I = (a)(b)(c)(d)(e)$ is obtained.

Perfect Shuffle

- Perfect shuffle is a special permutation function suggested by Harold Stone (1971) for parallel processing applications.
- Obtained by rotating the binary address one position left.
- The perfect shuffle and its inverse for 8 objects are shown here:

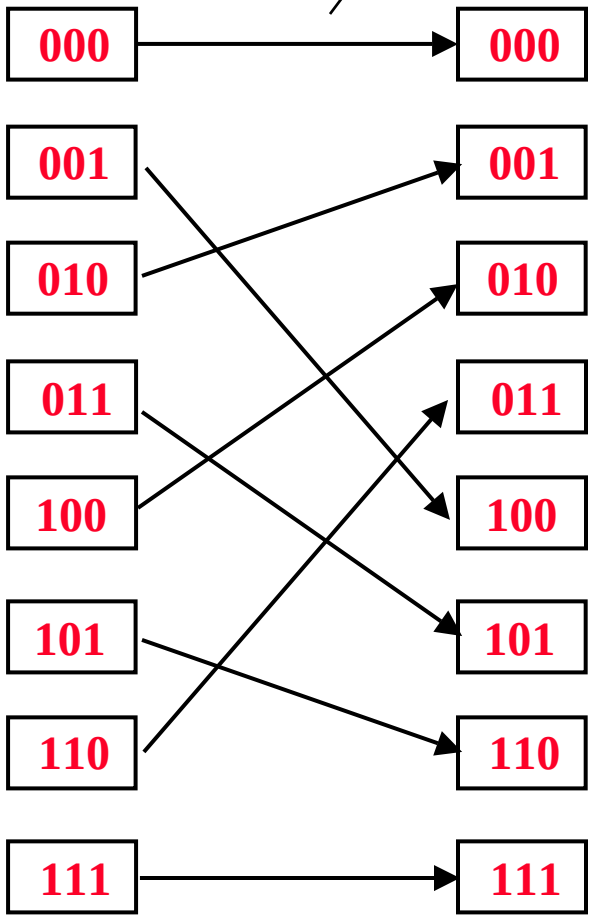
Inverse Perfect Shuffle: rotate binary address one position right

e.g. For N = 8



Perfect Shuffle

(circular shift left one position)



Inverse Perfect Shuffle

Generalized Structure of Multistage Interconnection Networks (MINS)

Fig 2.23 page 91

Kai Hwang ref.

See handout

Multi-Stage Networks (MINS) Example: The Omega Network Ω

- In the Omega network, perfect shuffle is used as an inter-stage connection (ISC) pattern for all $\log_2 N$ stages.
- Routing is simply a matter of using the destination's address bits to set switches at each stage.
- The Omega network is a single-path network: There is just one path between an input and an output.
- It is equivalent to the Banyan, Staran Flip Network, Shuffle Exchange Network, and many others that have been proposed.
- The Omega can only implement $N^{N/2}$ of the N! permutations between inputs and outputs in one pass, so it is possible to have permutations that cannot be provided in one pass (i.e. paths that can be blocked).
  - For N = 8, there are $8^4/8! = 4096/40320 = 0.1016 = 10.16\%$ of the permutations that can be implemented in one pass.
- It can take $\log_2 N$ passes of reconfiguration to provide all links. Because there are $\log_2 N$ stages, the worst case time to provide all desired connections can be $(\log_2 N)^2$.

ISC

N = size of network

2x2 switches used Log_2 N stages

ISC patterns used define MIN topology/connectivity
Here, ISC used for Omega network is perfect shuffle

Multi-Stage Networks:
The Omega Network

ISC = Perfect Shuffle
a= b = 2 (i.e 2x2 switches used)

Node Degree = 1 bi-directional link or 2 uni-directional links
Diameter = $\log_2 N$ (i.e number of stages)
Bisection width = N/2

N/2 switches per stage, $\log_2 N$ stages, thus:
Complexity = $O(N \log_2 N)$

Fig 2.24 page 92

Kai Hwang ref.

See handout (for figure)

MINs Example: Baseline Network

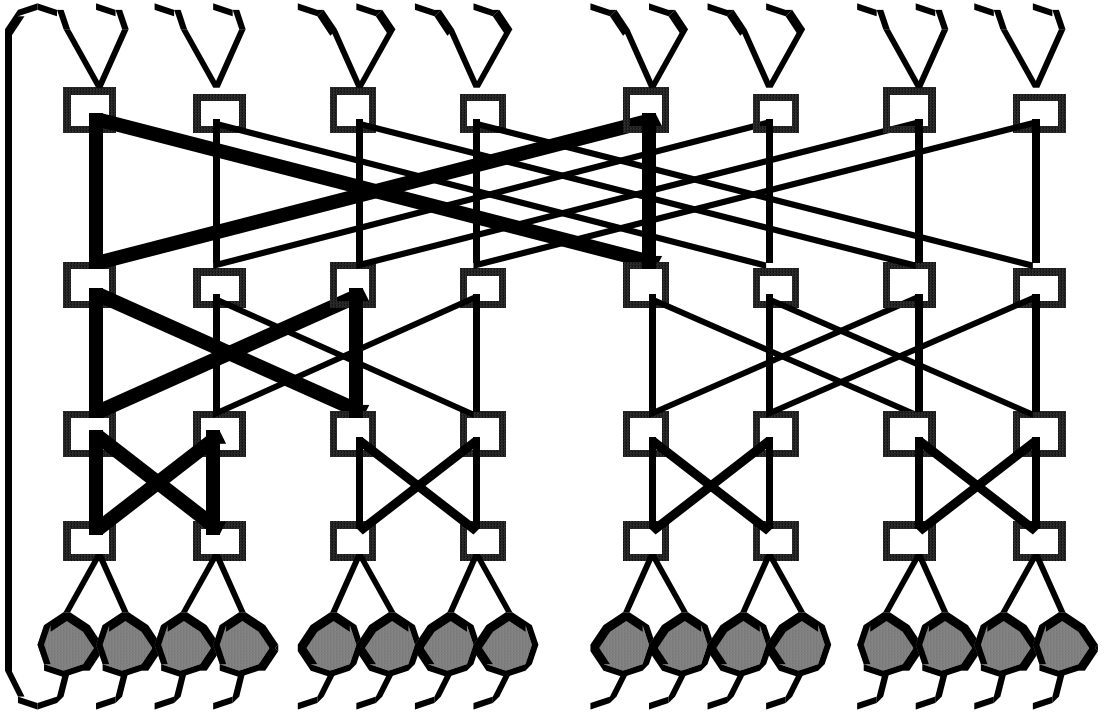
Fig 2.25 page 93

Kai Hwang ref.

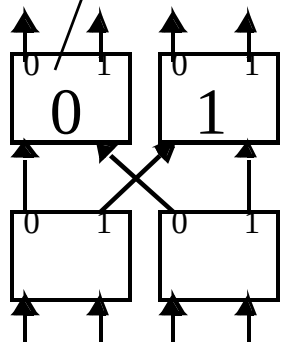
See handout

MINs Example: Butterfly Network

Constructed by connecting 2x2 switches doubling the connection distance at each stage
Can be viewed as a tree with multiple roots



Distance Doubles



Building block

Example: N = 16

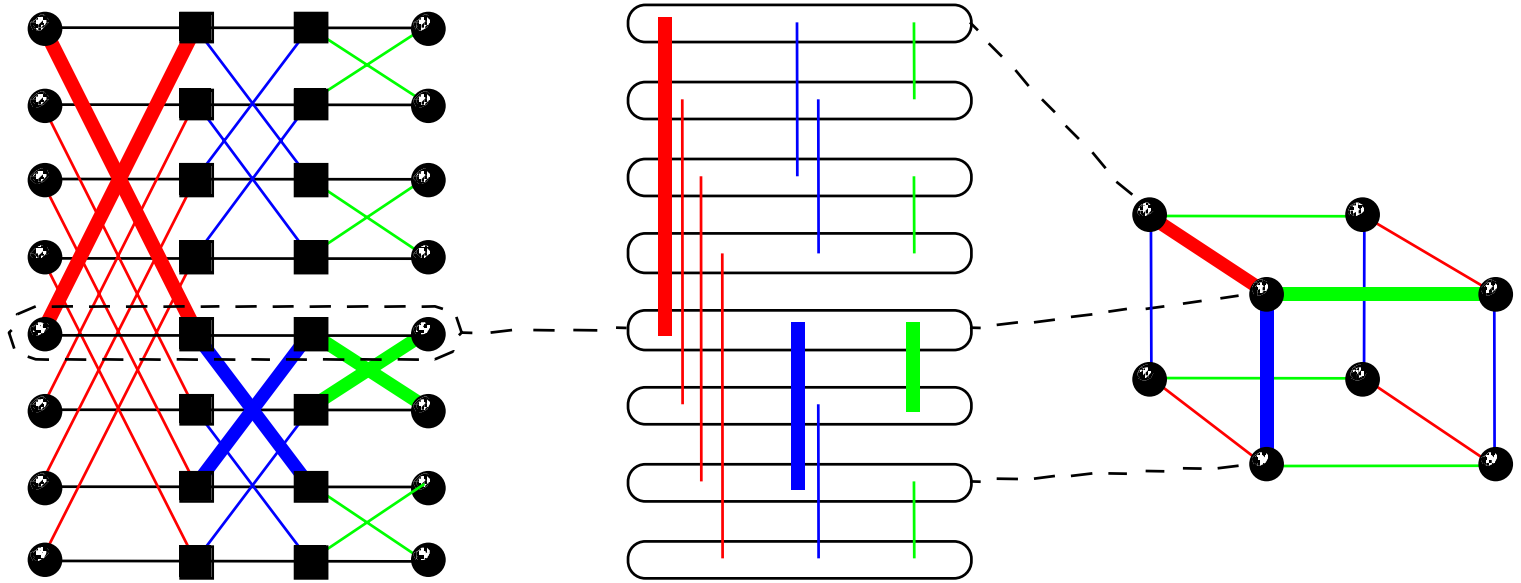
- **Complexity: N/2 x $\log_2 N$ (# of switches in each stage x # of stages)**
- **Exactly one route from any source to any destination node.**
- **R = A XOR B, at level *i* use 'straight' edge if r_i=0, otherwise cross edge**
- **Bisection width = N/2**
- **Diameter = Average Distance = $\log_2 N$ = Number of stages**

i.e O(N $\log_2$ N)

Complexity = O(N $\log_2$ N)

N = Number of nodes

Relationship Between Butterfly Network & Hypercubes



Relationship:

- **The connection patterns in the two networks are isomorphic (identical).**
  - **Except that Butterfly always takes $log_2 n$ steps.**

MIN Network Latency Scaling Example

$O(\log_2 N)$ Stage N-node MIN using 2x2 switches:

Cost or Complexity = $O(N \log_2 N)$

i.e. # of stages

- **Max distance:** $\log_2 N$ (good latency scaling)
- **Number of switches:** 1/2 N log N (good complexity scaling)
- overhead = o = 1 us, BW = 64 MB/s, Δ = 200 ns per hop

Switching/routing delay per hop

- Using pipelined or cut-through routing:
- $T_{64}(128)$ = 1.0 us + 2.0 us + 6 hops * 0.2 us/hop = 4.2 us — N= 64 nodes
- $T_{1024}(128)$ = 1.0 us + 2.0 us + 10 hops * 0.2 us/hop = 5.0 us — N= 1024 nodes

Message size n = 128 bytes

Only 20% increase in latency for 16x network size increase

Good latency scaling

- Store and Forward — h, n/B, Δ
- $T_{64}^{sf}(128)$ = 1.0 us + 6 hops * (2.0 + 0.2) us/hop = 14.2 us — N= 64 nodes
- $T_{1024}^{sf}(128)$ = 1.0 us + 10 hops * (2.0 + 0.2) us/hop = 23 us — N= 1024 nodes

o

~ 60% increase in latency for 16x network size increase

Latency when sending n = 128 bytes for N = 64 and N = 1024 nodes

Summary of Static Network Characteristics

Table 2.2 page 88

Kai Hwang ref.

See handout

Summary of Dynamic Network Characteristics

Table 2.4 page 95

Kai Hwang ref.

See handout

Example Networks: Cray MPPs

Distributed Memory SAS

Both networks used in T3D and T3E are: Point-to-point (static) using the 3D Torus topology

- **T3D: Short, Wide, Synchronous (300 MB/s).**
  - **3D bidirectional torus up to 1024 nodes, dimension order, virtual cut-through, packet switched routing.**
  - **24 bits: 16 data, 4 control, 4 reverse direction flow control**
  - **Single 150 MHz clock (including processor).**
  - **flit = phit = 16 bits.**
  - **Two control bits identify flit type (idle and framing).**
    - **No-info, routing tag, packet, end-of-packet.**
- **T3E: long, wide, asynchronous (500 MB/s)**
  - **14 bits, 375 MHz**
  - **flit = 5 phits = 70 bits**
    - **64 bits data + 6 control**
  - **Switches operate at 75 MHz.**
  - **Framed into 1-word and 8-word read/write request packets.**

Parallel Machine Network Examples

$\tau = 1/f$ **W or Phit** Δ

i.e basic unit of flow-control (frame size)

Machine	Topology	Cycle Time (ns)	Channel Width (bits)	Routing Delay (cycles)	Flit (data bits)
nCUBE/2	Hypercube	25	1	40	32
TMC CM-5	Fat-Tree	25	4	10	4
IBM SP-2	Banyan	25	8	5	16
Intel Paragon	2D Mesh	11.5	16	2	16
Meiko CS-2	Fat-Tree	20	8	7	8
CRAY T3D	3D Torus	6.67	16	2	16
DASH	Torus	30	16	2	16
J-Machine	3D Mesh	31	8	2	8
Monsoon	Butterfly	20	16	2	16
SGI Origin	Hypercube	2.5	20	16	160
Myricom	Arbitrary	6.25	16	50	16