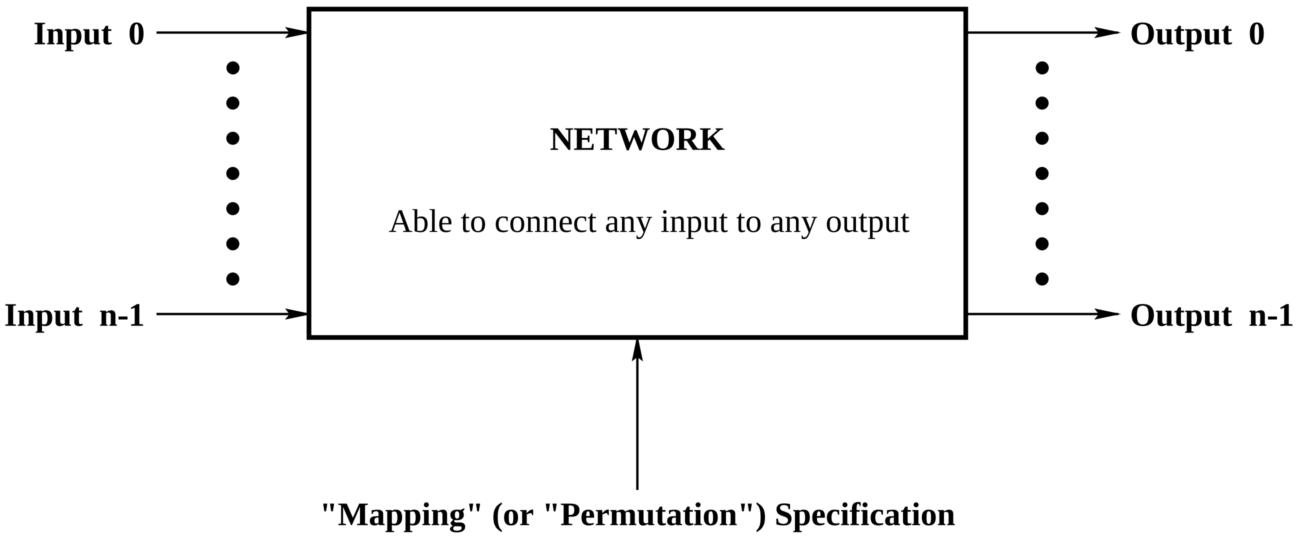
Interconnection Networks I

Permutations and Static Networks

Tópico 6

Interconnection Networks



Questions:

1. What are the characteristics of networks?

2. How does one specify a mapping?

Characteristics of Networks

- "Active" design features [Behavioural]:
  - Operating mode (synchronous or asynchronous)
  - Control strategy (centralised - SIMD or distributed - MIMD)
  - Switching method (circuit or packet switching)

Characteristics of Networks$_{(cont)}$

- "Passive" design features [Architectural]:
  - Degree (of node) and diameter (of network)
  - Bisection width
  - Topology (static or dynamic networks)

Specifying Mappings/Permutations

- Data routing functions

Operating Mode

Refers to whether the communication over the network is *synchronous* or *asynchronous*.

In general:

- synchronous in SIMD;
- asynchronous in MIMD;
- ??? (it depends) in hybrids.

Control Strategy

The nodes in a network need to be provided with "information" in order for them to route messages.

How is this information distributed?

1. *Centralised* - single controller producing multiple control signals. [Mainly SIMD]

2. *Distributed* - each node controls its own routing function. [Mainly MIMD]

3. in hybrids, again it depends.

Switching Method

"Switching" is another word for "routing", but is often used in the context of local routing decisions of messages at nodes [via "throwing switches"].

- *Circuit Switching*
  - Most suitable for SIMD, or bulk data transmission.
  - Low control overhead and simpler logic ("switch box").

Switching Method$_{(cont)}$

- *Packet Switching* (or *Store and Forward*)
  - MIMD, or short data transmission.
  - Packets "self-route", so switch boxes are more complex.
  - Routing conflicts are possible, so there is a need for a conflict resolution strategy.
- Hybrids of the two
  - e.g. *wormhole* or *cut-through* routing.

Node Degree

Basically, the number of wires leaving a processor, or the number of communication channels.

- Reflects the I/O bandwidth of **one** processor.
- Scalability requires constant degree independent of network size. [Why ?]

Network Diameter

The **maximum** shortest path between any two nodes (i.e. the shortest path between the two most distant nodes).

- Sometimes equated to *latency*, or the time to communicate between nodes.

- In any case, *latency* $=$ *f (diameter)* and designers like it to be **small**.

Bisection Width

Cut the machine in half (bisect), clumping each half into a “supernode”. The degree of the supernodes = *bisection width*.

- Basically, denotes the communication bandwidth of the entire network.
- It is the upper bound on the volume of messages movable from one half of the network to the other, simultaneously

High bisection width is better, because in algorithms requiring large amounts of data movements, the size of the data set divided by the bisection width puts a lower bound on the complexity of the parallel algorithm.

Network Topology

The *network* can be represented by a graph where nodes are switch boxes and edges communication wires. The *topology* is the shape of the graph, normally regular in structure and either *static* or *dynamic*.

Static:

- Fixed, point-to-point connections;
- Dedicated links, *passive* switches.

Dynamic:

- Reconfigurable, “virtual” connections;
- “Variable”, *active* switches;
- Can reconfigure a virtual network over the hardware network to match a program’s communication pattern.

Data Routing Functions

Data routing functions (software) specify how data is to be routed on networks (hardware).

Common classes of routing functions include:

- one-to-one (permutation)
- one-to-all (broadcast)
- many-to-many (multicast)
- one-to-many ("personalised")
- many-to-one (reduction)

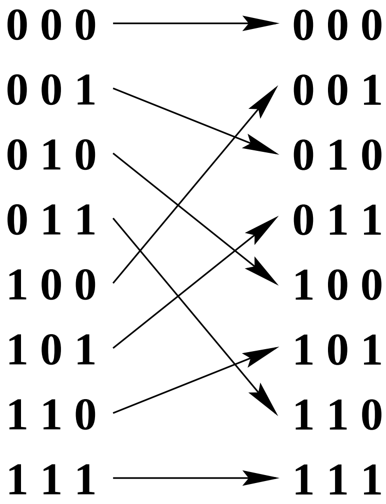
Permutations

Given 2^n input vertices, a *permutation* defines a mapping to a set of 2^n output vertices. Permutations are *simple routing functions*.

Perfect shuffle: $\sigma(x)$

$$\sigma(x) = a_{n-2} \ldots a_0 a_{n-1}$$

where $x = a_{n-1} \ldots a_0$, $0 \leq x < 2^n$, $a_i =$ "bit i".



Left cyclic rotation
i.e. shift one place to the left.

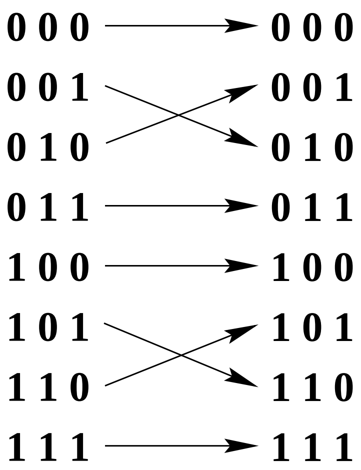
Inverse Shuffle: $\sigma^{-1}(x)$

$\sigma^{-1}(x) = a_0 a_{n-1} .... a_1$ *Right cyclic rotation*

kth Sub-shuffle: $\sigma_k(x)$

$\sigma_k(x) = a_{n-1} ... a_k a_{k-2} ... a_0 a_{k-1}$

E.g. σ_2



Left cyclic rotation on a subset, i.e. the k least significant bits.

kth super-shuffle: $\sigma^k(x)$

$$\sigma^k(x) = a_{n-2}...a_{n-k}a_{n-1}a_{n-k-1}...a_0$$

E.g. σ^2

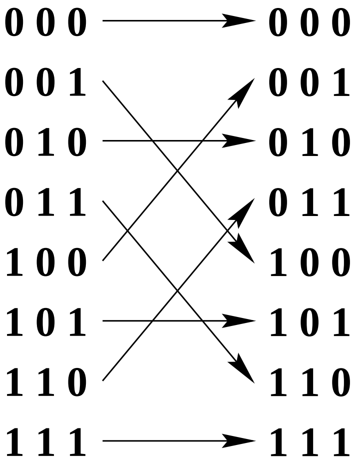
0 0 0 → **0 0 0**
0 0 1 → **0 0 1**
0 1 0 → **1 0 0**
0 1 1 → **1 0 1**
1 0 0 → **0 1 0**
1 0 1 → **0 1 1**
1 1 0 → **1 1 0**
1 1 1 → **1 1 1**

Left cyclic rotation on a subset, i.e. the k most significant bits.

Exchange Permutations: $\epsilon_i(x)$

$\epsilon_i(x) = a_{n-1}...\overline{a_{i-1}}...a_0$ $\qquad$ *Complement* a_{i-1}

Butterfly Permutation $\beta(x)$



$\beta(x) = a_0 a_{n-2} ... a_1 a_{n-1}$

Swap the most and least significant bits.

kth sub-butterfly: $\beta_k(x)$

$\beta_k(x) = a_{n-1} ... a_k a_0 a_{k-2} ... a_1 a_{k-1}$

Swap the least significant bit with bit *k-1*

kth super-butterfly: $\beta^k(x)$

$\beta^k(x) = a_{k-1} a_{n-2} ... a_k a_{n-1} a_{k-2} ... a_0$

Swap the most significant bit with bit *k-1*

Alternative Specification

Recall the perfect shuffle permutation (routing function, albeit a simple one):

$0 \leq x < 2^n$ where x has binary representation

$$a_{n-1}a_{n-1}....a_1a_0$$

Another functionally equivalent way to specify perfect shuffle routing is:

$$\sigma(x) = \begin{cases} 2 \cdot x & \text{for } 0 \leq x < 2^{n-1} \\ 2 \cdot x + 1 - 2^n & \text{for } 2^{n-1} \leq x < 2^n \end{cases}$$

Check this out on paper with an example, e.g. $\{0,1,...,7\}$. Try to do the same for the binary specification of the inverse shuffle.

Concluding Remarks

These input $\rightarrow$ output data mappings have been useful in developing parallel algorithms and simple enough that networks have been built to specifically implement them e.g. shuffle-exchange and butterfly networks.

Concluding Remarks$_{(cont)}$

Can summarise network performance as being influence by the following factors:

- Functionality, i.e. operational characteristics;
- Network Latency - worst-case time delay for unit message transfer (time related);
- Bandwidth - maximum data transfer rate (capacity related);
- Hardware Complexity - implementation costs; and
- Scalability.

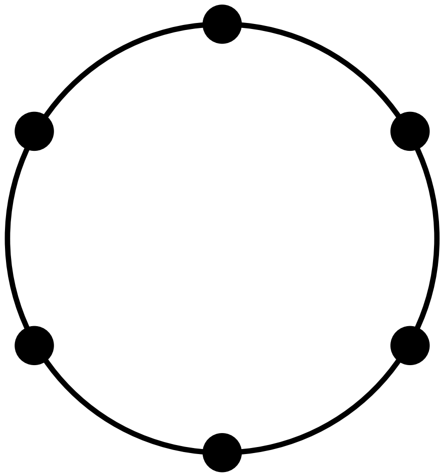
Static Network Topologies

Can be classified according to dimensionality,
[well, at least I have tried to :-)]:

- One-dimensional
- Two-dimensional
- Three-dimensional
- n-dimensional

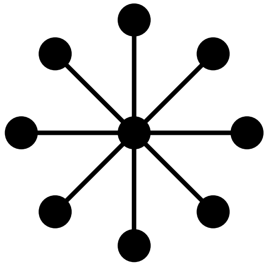
One-dimensional

Linear Array : Number of Nodes = n
Network Diameter = n - 1
Bisection Width = 1
Node Degree = 2



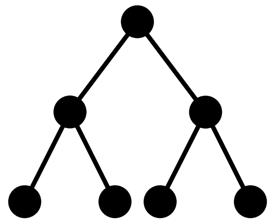
Ring : Number of Nodes = n
Network Diameter = n/2
Bisection Width = 2
Node Degree = 2

Two-dimensional



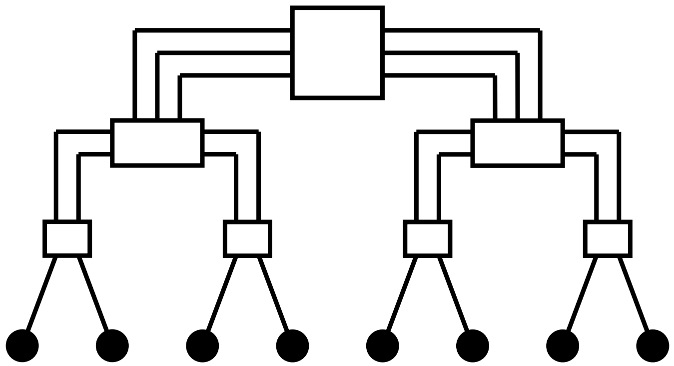
Star :

Number of Nodes = n
Network Diameter = 2
Bisection Width = n/2
Node Degree = n - 1

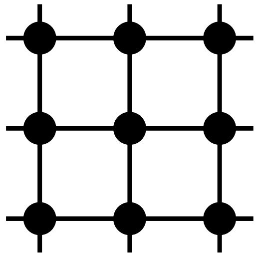


(Binary) Tree :

Number of Nodes = n (= 2^k - 1)
Network Diameter = 2(k - 1)
Bisection Width = 1
Node Degree = 3



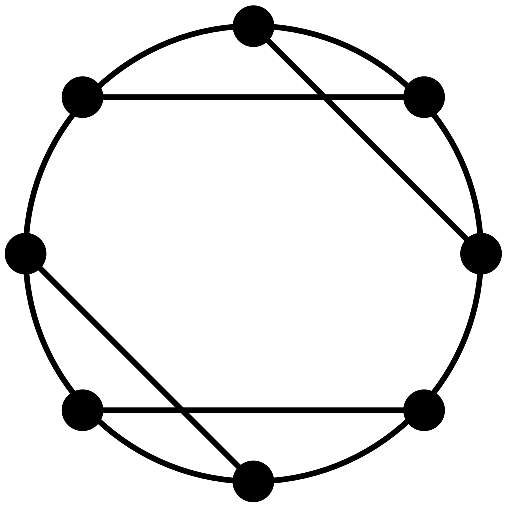
(Binary) Fat Tree

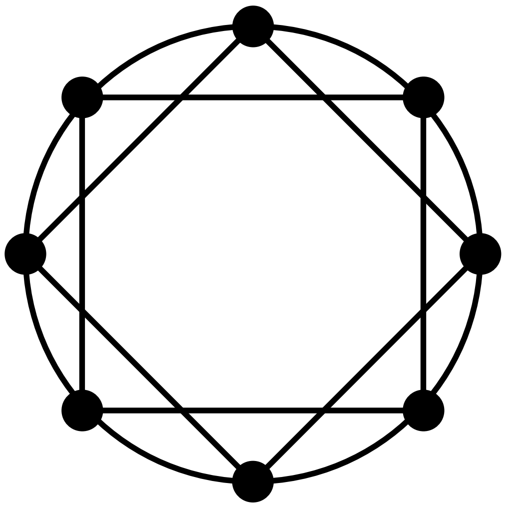


Mesh :

Number of Nodes = n^2
Network Diameter = 2(n - 1)
Bisection Width = n
Node Degree = 4

Two-dimensional$_{(cont)}$

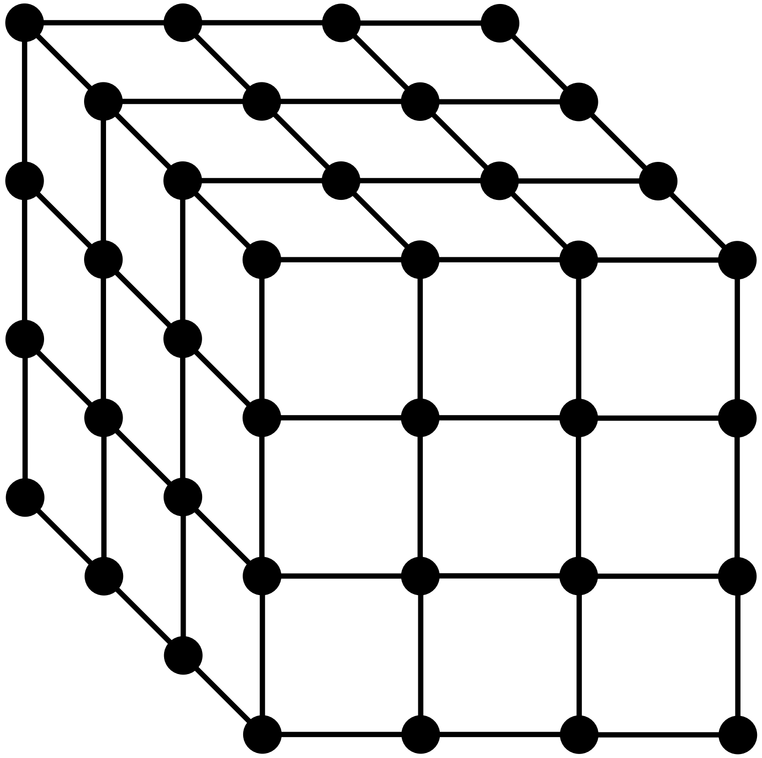


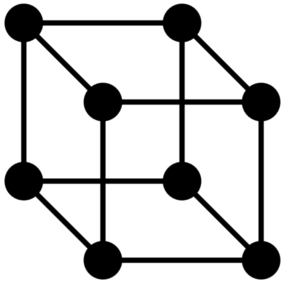


Chordal Rings :

Number of Nodes = 8	Number of Nodes = 8
Network Diameter = 3	Network Diameter = 2
Bisection Width = 2	Bisection Width = 6
Node Degree = 3	Node Degree = 4

Three-dimensional





3-Cube :

Number of Nodes = 8

Network Diameter = 3

Bisection Width = 4

Node Degree = 3

Mesh :

Number of Nodes = n^3

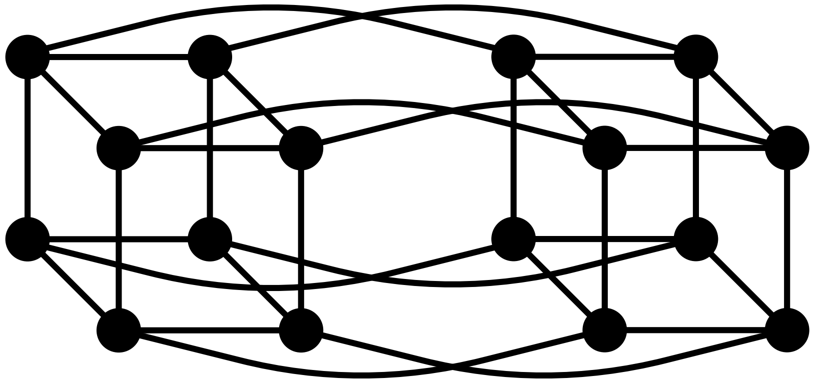
Network Diameter = 3(n - 1)

Bisection Width = n^2

Node Degree = 6

n-dimensional

Mainly n-cubes or variants of them.



n-Cube :

Number of Nodes = N (= 2^n)

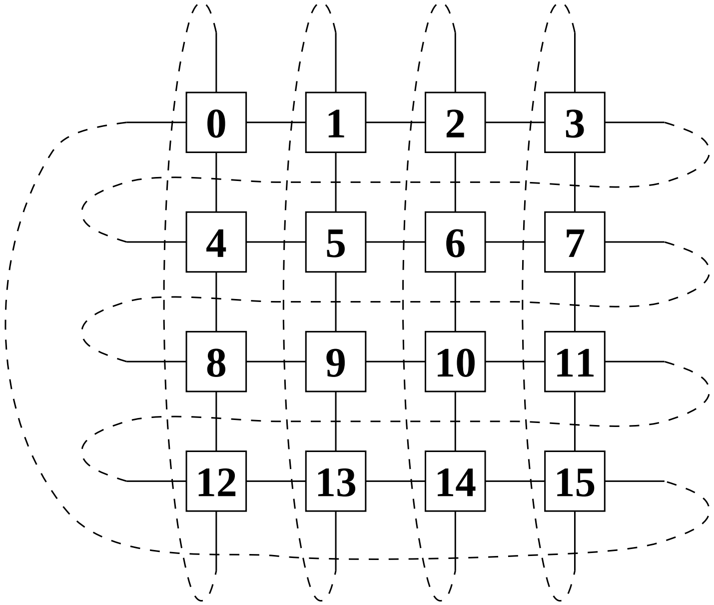
Network Diameter = log(2) N (= n)

Bisection Width = N/2 (= 2^(n-1))

Node Degree = log(2) N (= n)

A Square Mesh

with “skewed wraparound”



Used in ILLIAC IV
(an early SIMD array processor).

Exercise: Show this is equivalent to a *Chordal Ring*.

The Four Routing Functions:

$$\mathbf{f}_N,\ \mathbf{f}_S,\ \mathbf{f}_E,\ \mathbf{f}_W$$

$$\mathrm{f}_N = (X - w) \text{ MOD } n;$$
$$\mathrm{f}_S = (X + w) \text{ MOD } n;$$
$$\mathrm{f}_E = (X + 1) \text{ MOD } n;$$
$$\mathrm{f}_W = (X - 1) \text{ MOD } n;$$

Where :

n = the number of PEs;

$w = \sqrt{n}$, (side length);

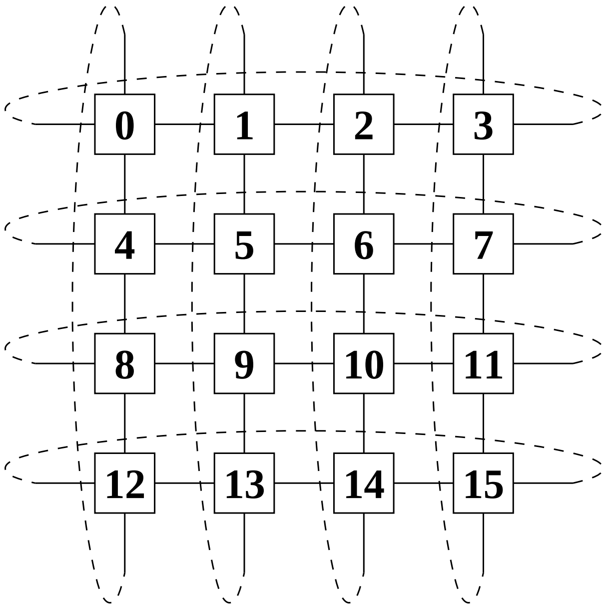
X is the starting node,

(the vertex address); and

f_i is the end node.

A Square Mesh

without Skew (Torus)



Doughnut shaped.

The Four Routing Functions:

$$\mathbf{f}_N,\ \mathbf{f}_S,\ \mathbf{f}_E,\ \mathbf{f}_W$$

$$\mathrm{f}_N = (X - w) \text{ MOD } w^2;$$
$$\mathrm{f}_S = (X + w) \text{ MOD } w^2;$$
$$\mathrm{f}_E = ((X + 1) \text{ MOD } w) + w \cdot \lfloor \tfrac{X}{w} \rfloor;$$
$$\mathrm{f}_W = ((X - 1) \text{ MOD } w) + w \cdot \lfloor \tfrac{X}{w} \rfloor;$$

Where :

n = the number of PEs;

$w = \sqrt{n}$;

X is the starting node; and

f_i is the end node.

k-dimensional Hypercube

Graph G = (V, E), where

V = $\{i : 0 \leq i \leq 2^k\text{-}1\}$,

E = $\{(i,j)$: $i,j \in$ V and

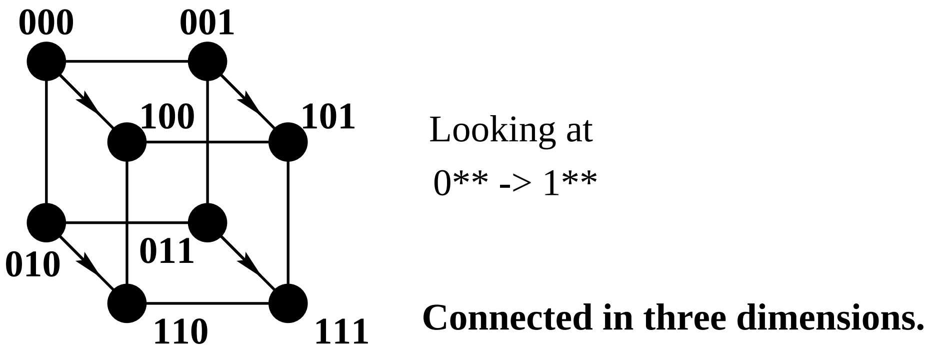
abs$(i - j) = 2^m$, $0 \leq m \leq$ k-1 $\}$

i.e. label the 2^k vertices with integers

in the range 0....2^k-1.

Then a pair of vertices have an edge between them if the binary representation of their addresses differ by **exactly one** bit.

Take a 3-cube

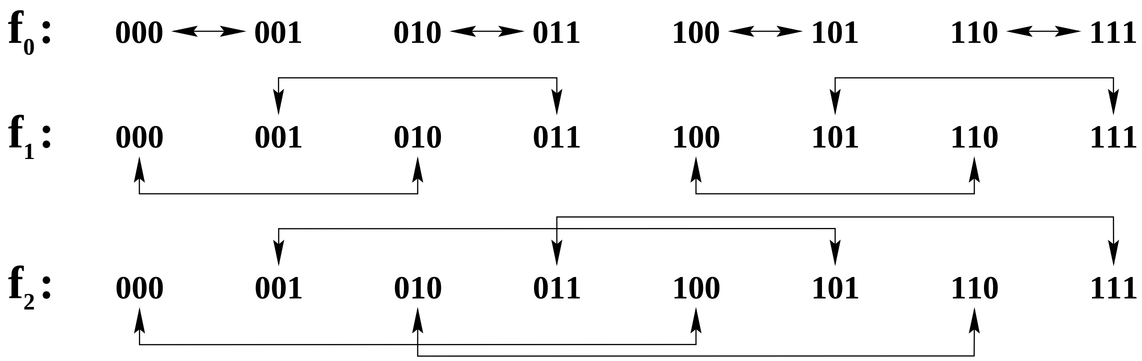


Each bit of the processor address specifies a different *dimension* of the processor's position.

For an n-dimensional hypercube there are n possible routing functions (or permutations).

Complex routing can be done via the composition of multiple routing functions.

Example: n-cube (n=3)



Formally, we define the routing function f_i, for $N = 2^n$ processors in a n-cube to be:

$$f_i(a_{n-1}, ..., a_0) = a_{n-1}...\overline{a_i}a_{i-1}...a_0$$

Informally, just invert the bit at position i in all node addresses to perform the routing.

How do you use these n functions to route from **any** node to any other?

Finishing up with Static Networks

- One last thing to note, with respect to k-ary n-cubes, is **Latency** has 2 components:

  1. the distance the message has to travel (the number of hops); and,

  2. the time a message takes to make one hop $= \frac{message\ length}{channel\ width}$

- *Network Throughput* is the total number of messages a network can handle per time unit. "Related" to bisection bandwidth via reference to network capacity. Typically both are functions of capacity, e.g. $throughput = \frac{1}{Constant} \times capacity$

Finishing up with Static Networks$_{(cont)}$

- *"Hot Spot"* is another buzzword - this is a "bottleneck area" (traffic jam) in a network, and is inherently a *locality* concept or quantity. *Hot spot throughput* is the maximum rate at which messages can be sent from one node to another; the nodes being specifically chosen (usually the worst case pair, thus creating the "largest" hot spot).

- Fat tree and k-ary n-cube networks are considered universal in the sense that they can simulate any other network of the same volume.

Static Networks

- Simple and comparatively cheap, but lack flexibility.
- Scalable systems require small constant node degrees. Network diameter has become less critical with the invention of hardware routing.
- *A priori* matching of the network topology and the application's communication pattern is needed.
- “Fixed” communication patterns.

Dynamic Networks

- Allows "arbitrary" communication patterns.
- Reconfigurable communication paths via "switching" at nodes.
  Can sort of look at this as a "virtual" network overlying a fixed/static one.
- Design based on throughput, cost and switching methodology.
- Expensive.