# Scalable Interconnection Networks

# Goals

Latency as small as possible

As many concurrent transfers as possible

- operation bandwidth
- data bandwidth

Cost as low as possible

# Basic Definitions

Network interface

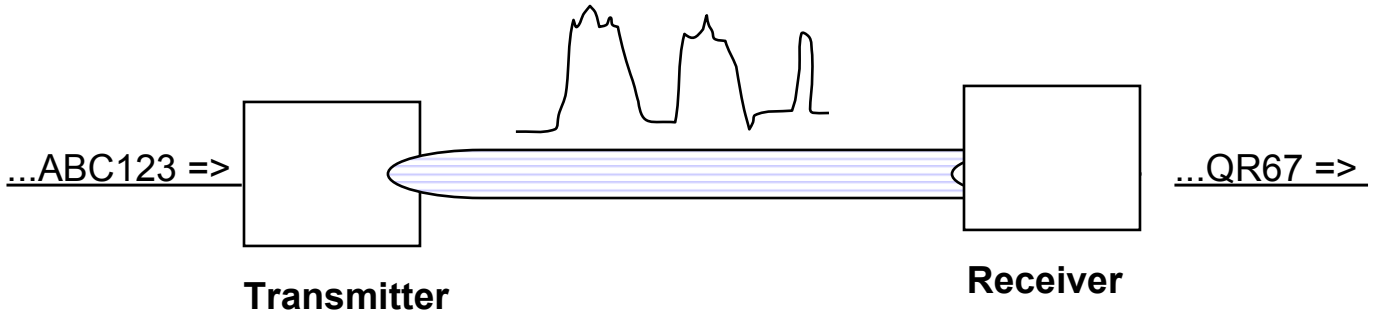
Links

- bundle of wires or fibers that carries a signal

Switches

- connects fixed number of input channels to fixed number of output channels

# Links and Channels




*transmitter* converts stream of digital symbols into signal that is driven down the link

*receiver* converts it back

- tran/rcv share *physical protocol*

trans + link + rcv form *Channel* for digital info flow between switches

*link-level protocol* segments stream of symbols into larger units: packets or messages (framing)

*node-level protocol* embeds commands for dest communication assist within packet

# Formalism

network is a graph V = {switches and nodes} connected by communication channels $C \subseteq V \times V$

Channel has width w and signaling rate $f = 1/\tau$

- channel bandwidth $b = wf$
- phit (physical unit) data transferred per cycle
- flit - basic unit of flow-control

Number of input (output) channels is switch *degree*

Sequence of switches and links followed by a message is a *route*

# What characterizes a network?

Topology (what)

- physical interconnection structure of the network graph
- direct: node connected to every switch
- indirect: nodes connected to specific subset of switches

Routing Algorithm (which)

- restricts the set of paths that msgs may follow
- many algorithms with different properties
  - gridlock avoidance?

Switching Strategy (how)

- how data in a msg traverses a route
- circuit switching vs. packet switching

Flow Control Mechanism (when)

- when a msg or portions of it traverse a route
- what happens when traffic is encountered?

# Topological Properties

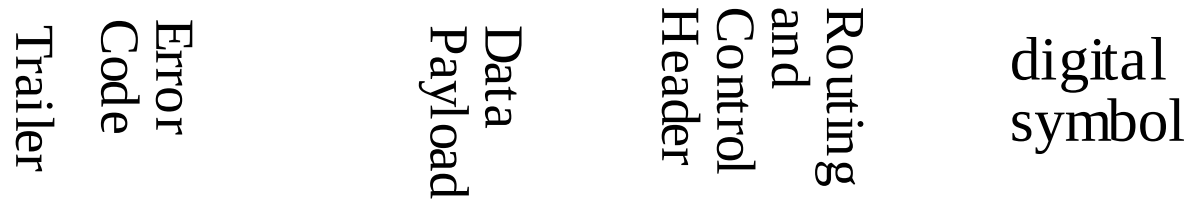
*Routing Distance* - number of links on route

*Diameter* - maximum routing distance

*Average Distance*

A network is *partitioned* by a set of links if their removal disconnects the graph

# Typical Packet Format




Sequence of symbols transmitted over a channel

Two basic mechanisms for abstraction

- encapsulation
- fragmentation

# Communication Perf: Latency

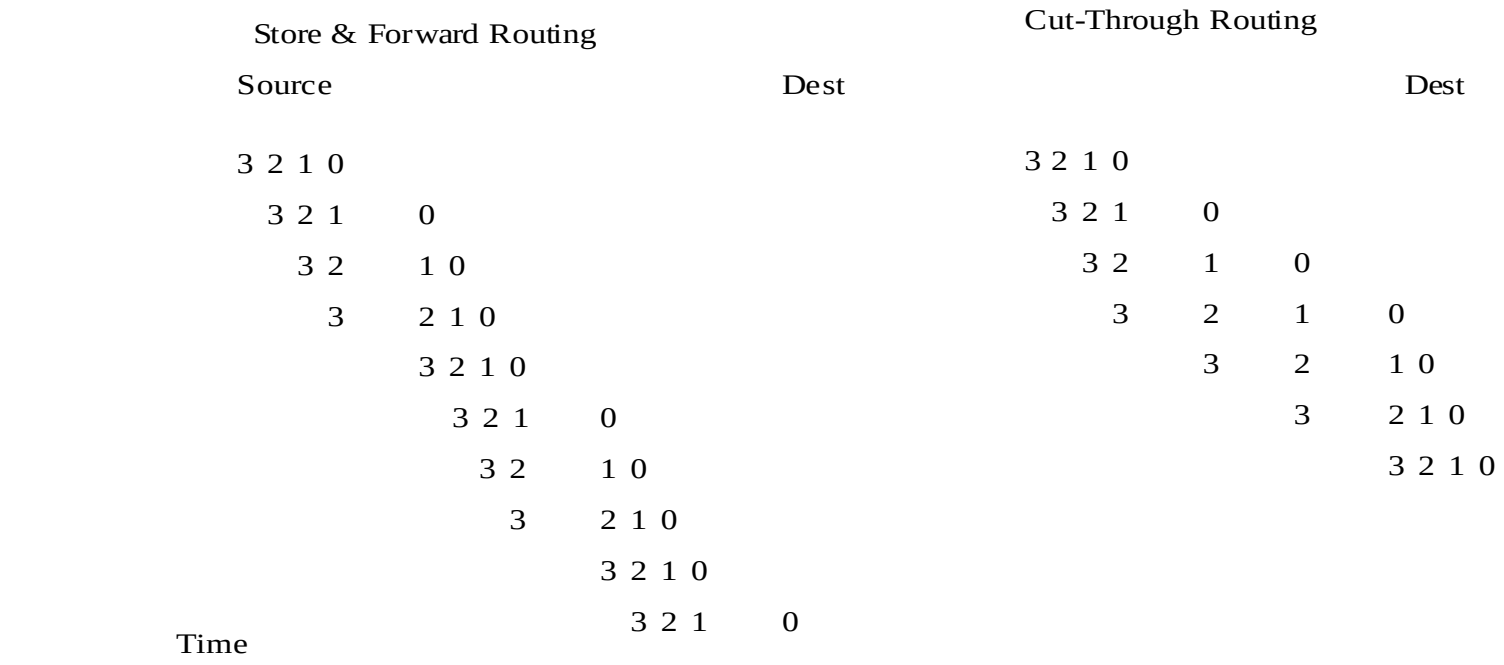
**$Time(n)_{s\text{-}d}$ = overhead + routing delay + channel occupancy + contention delay**

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

**Routing delay?**

**Contention?**

# Store&Forward vs Cut-Through Routing

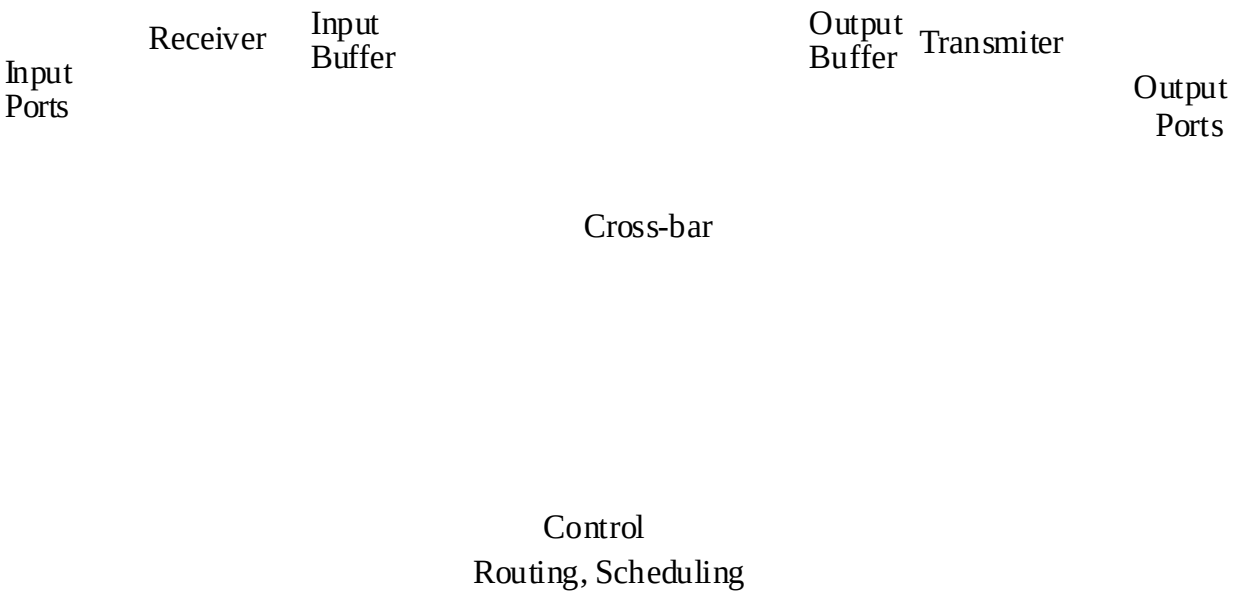



$h(n/b + \Delta)$ vs $n/b + h\Delta$

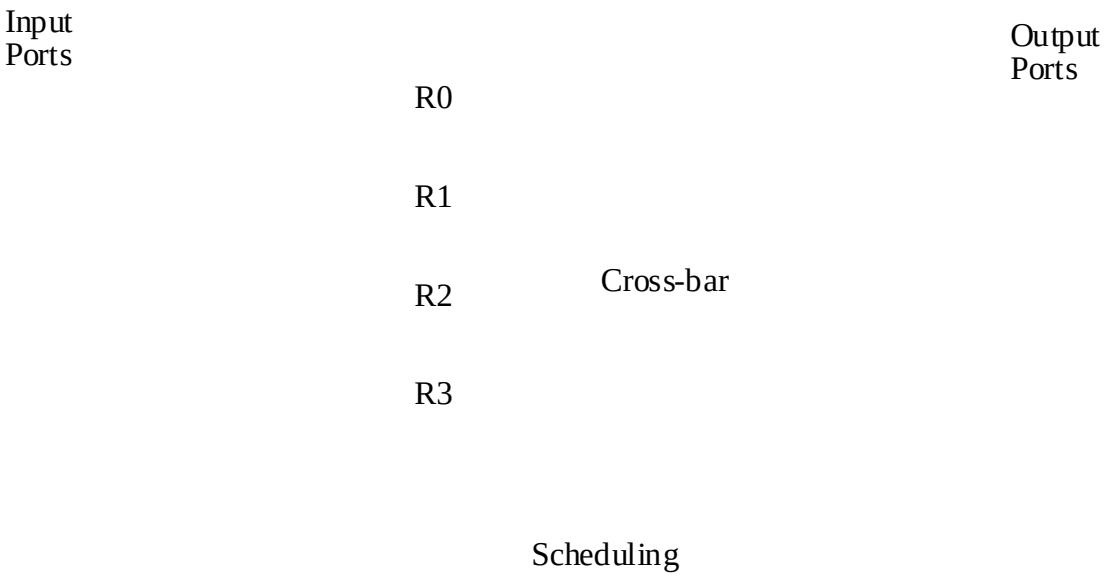
what if message is fragmented?

wormhole vs virtual cut-through

# Switch Design



# Input buffered swtich




Independent routing logic per input

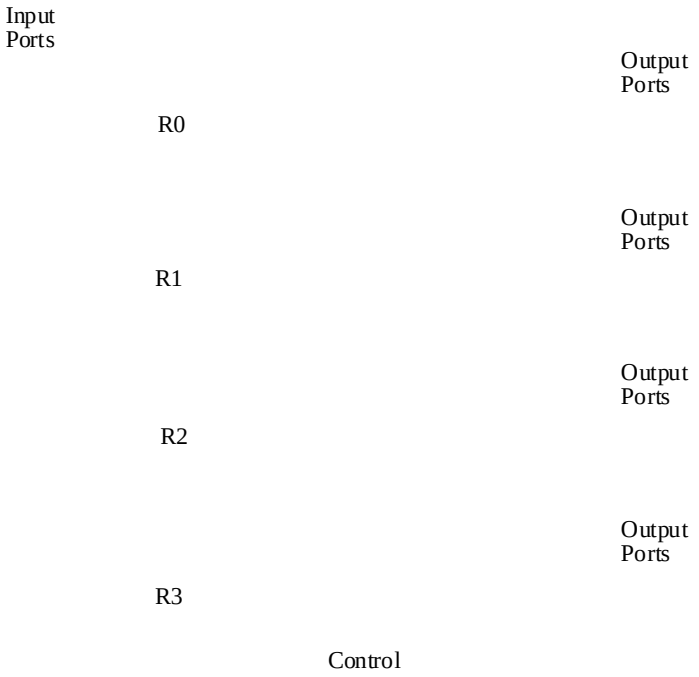
- FSM

Scheduler logic arbitrates each output

- priority, FIFO, random

Head-of-line blocking problem

# Output Buffered Switch




How would you build a shared pool?

# Output scheduling


n independent arbitration problems?

- static priority, random, round-robin

Simplifications due to routing algorithm?

General case is max bipartite matching

# Flow Control

What do you do when push comes to shove?

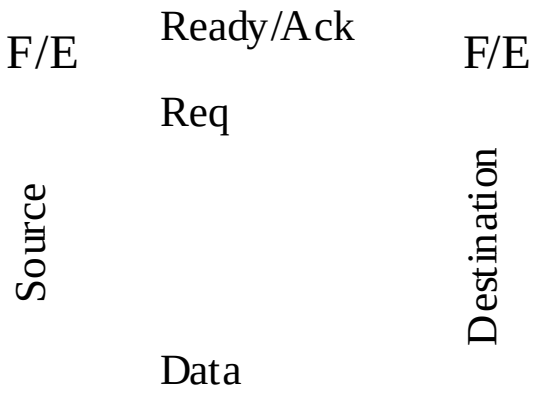
- ethernet: collision detection and retry after delay
- FDDI, token ring:  arbitration token
- TCP/WAN: buffer, drop, adjust rate
- any solution must adjust to output rate

Link-level flow control

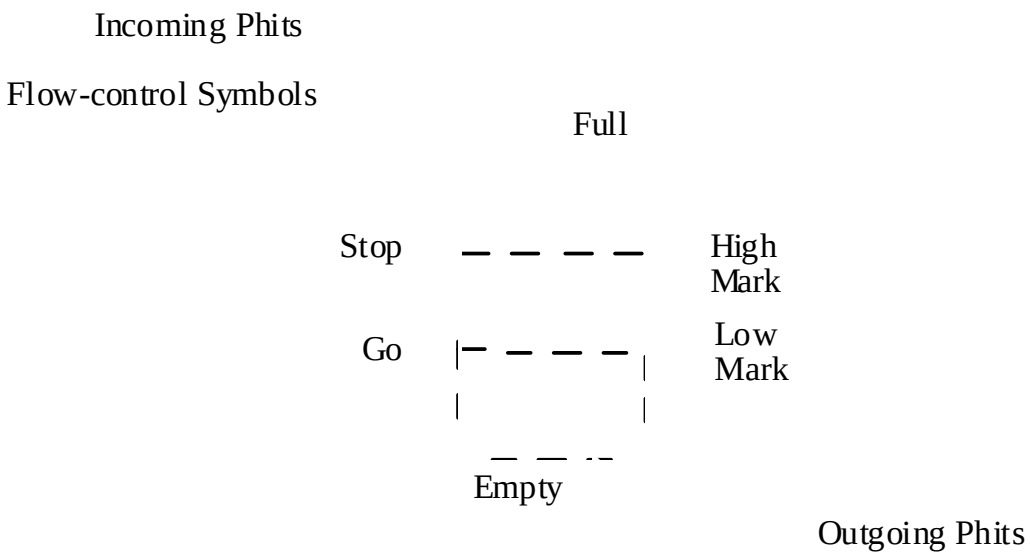
Ready

**Data**

# Examples

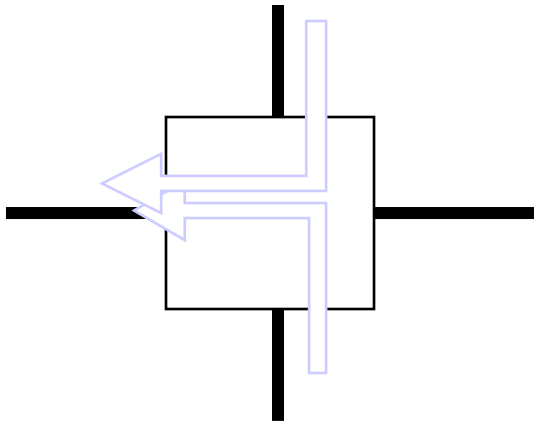


# Smoothing the flow




How much slack do you need to maximize bandwidth?

# Contention



Two packets trying to use the same link at same time

- limited buffering
- drop?

Most parallel mach. networks block in place

- link-level flow control
- tree saturation

Closed system - offered load depends on delivered

# Bandwidth

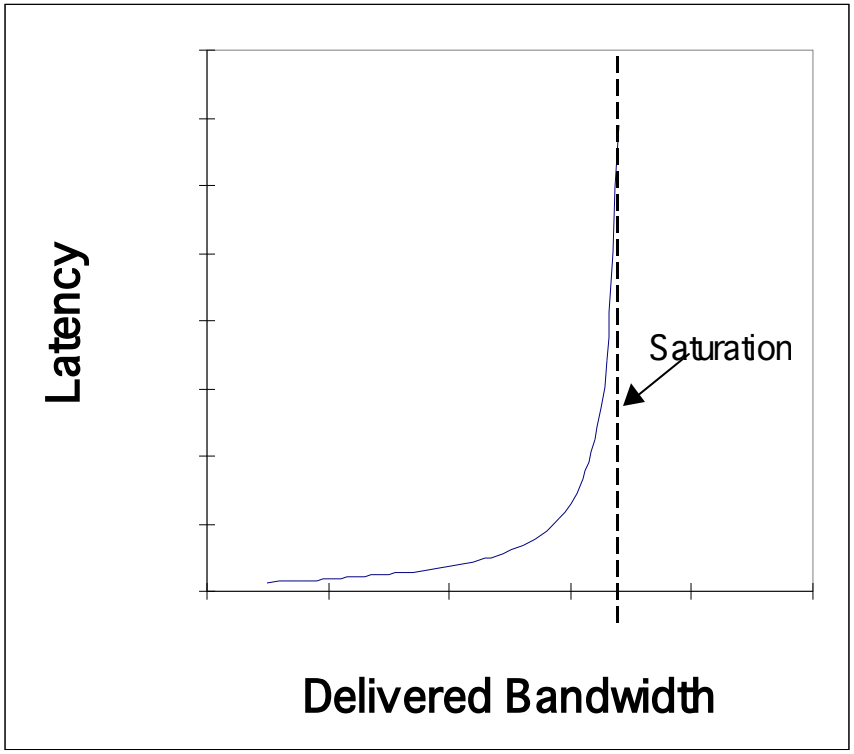
What affects local bandwidth?

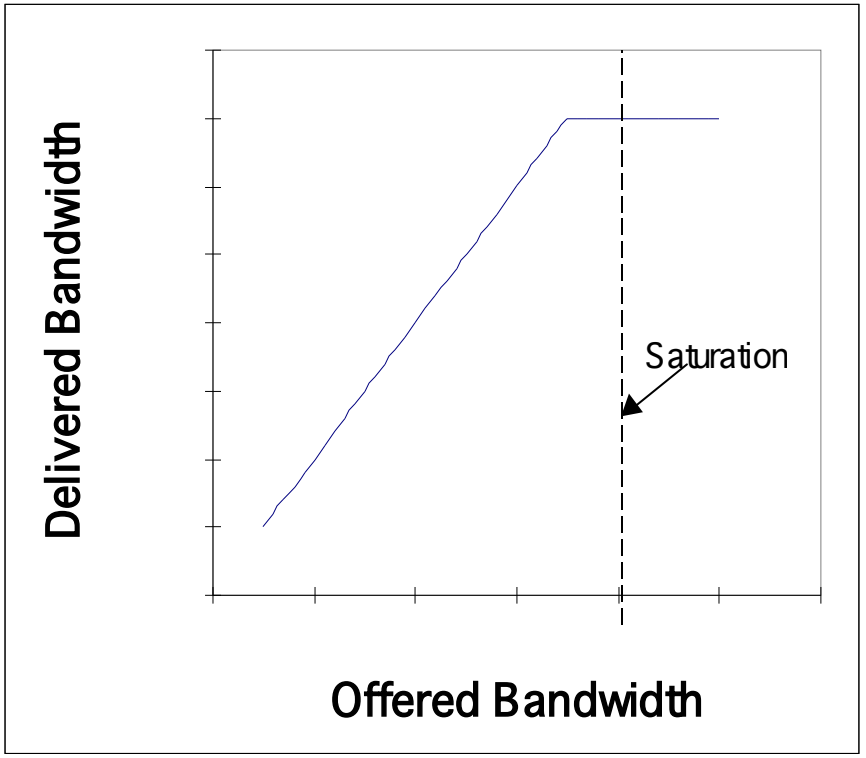
- packet density b x n/(**n + $n_e$**)
- routing delay b x n / (**n + $n_e$ + wΔ**)
- contention
  - endpoints
  - within the network

Aggregate bandwidth

- bisection bandwidth
  - sum of bandwidth of smallest set of links that partition the network
- total bandwidth of all the channels: Cb

# Saturation


Latency
Saturation
Delivered Bandwidth


Delivered Bandwidth
Saturation
Offered Bandwidth

# Interconnection Topologies

Logical Properties:

- distance, degree

Physcial properties

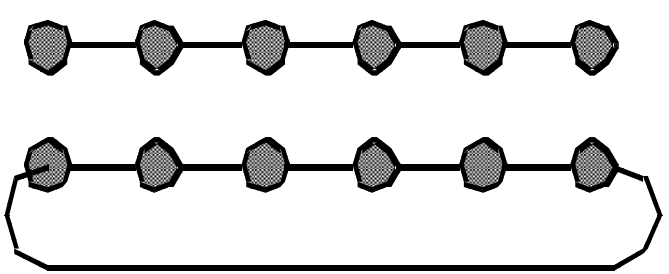
- length, width

Fully connected network

- diameter = 1
- degree = N
- cost?
  - bus => O(N), but BW is O(1)
  - crossbar => O($N^2$) for BW O(N)

VLSI technology determines switch degree

# Linear Arrays and Rings



Linear Array

Torus

Linear Array

- Diameter?
- Average Distance?
- Bisection bandwidth?
- Route A -> B given by relative address R = B-A

Torus?

Examples: FDDI, SCI, FiberChannel Arbitrated Loop, KSR1

# Multidimensional Meshes and Tori

2D Grid

3D Cube

*d*-dimensional array

- n = $k_{d-1}$ X ...X $k_O$ nodes
- described by *d*-vector of coordinates ($i_{d-1}$, ..., $i_O$)

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

*d*-dimensional *k*-ary torus (or *k*-ary *d*-cube)?

# Properties

Routing

- 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
- *dimension-order routing*

Average Distance

- d x 2k/3 for mesh
- dk/2 for cube

Wire Length?

Degree?

Bisection bandwidth?

- $k^{d-1}$ bidirectional links

Partitioning?

Physical layout?

- 2D in O(N) space
- higher dimension?

Short wires

# Embeddings in two dimensions

**6 x 3 x 2**

Embed multiple logical dimension in one physical dimension using long wires

# Trees

Diameter and avg. distance are logarithmic
- k-ary tree, height d = $\log_k N$

Fixed degree

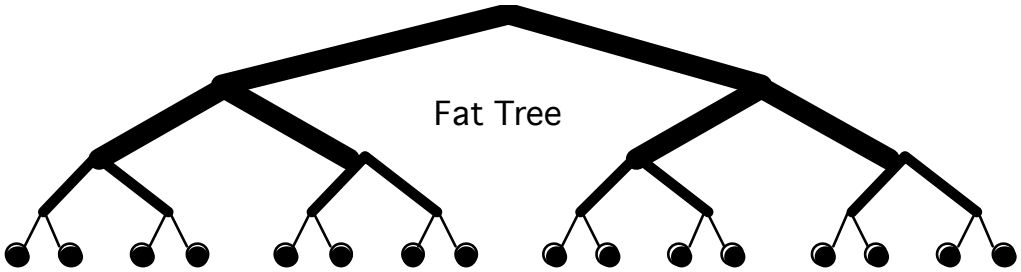
Routing

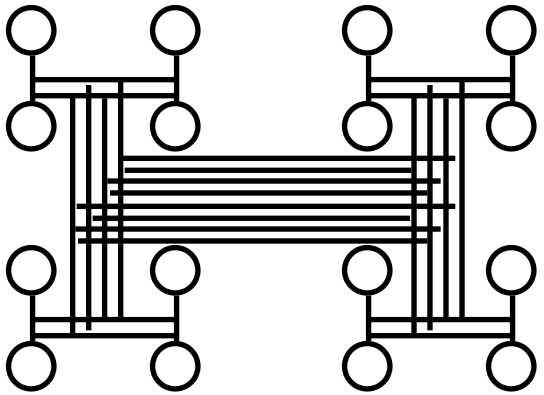
- R = B xor A
  - let i be position of most significant 1 in R, route up i+1 levels
  - down in direction given by low i+1 bits of B

H-tree space is O(N) with O(√N) long wires

Bisection BW?

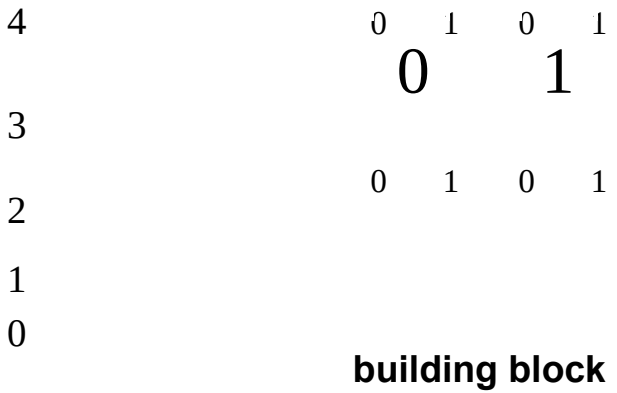
# Fat-Trees



Fatter links (really more of them) as you go up, so bisection BW scales with N

# Butterflies




**16 node butterfly**

Tree with lots of roots!

N log N (actually N/2 x logN)

Exactly one route from any source to any dest

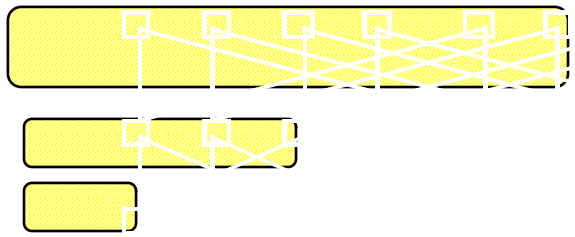
R = A xor B, at level i use ‘straight’ edge if $r_i$=0, otherwise cross edge

Bisection N/2

# Benes network and Fat Tree

16-node Benes Network (Unidirectional)

16-node 2-ary Fat-Tree (Bidirectional)



Back-to-back butterfly can route all permutations

- off line

# Hypercubes

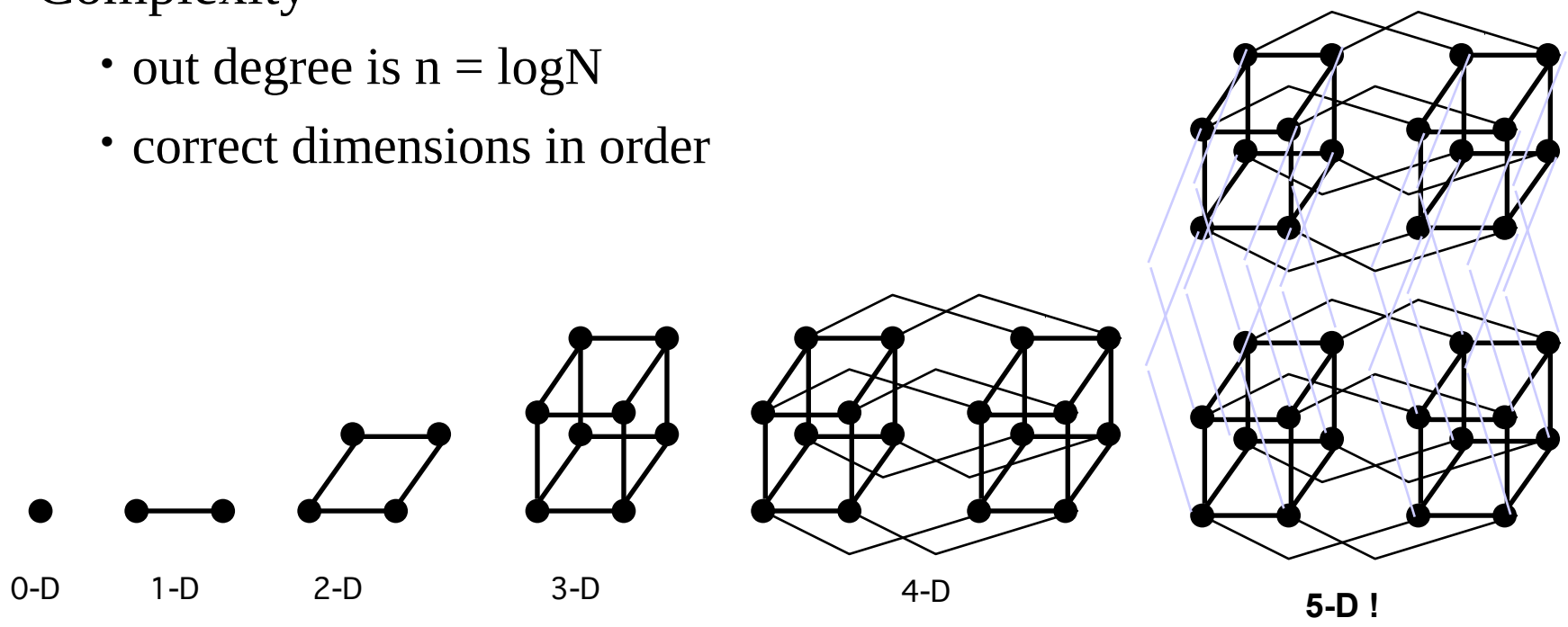
Also called binary n-cubes. # of nodes = N = $2^n$

O(logN) hops

Good bisection BW

Complexity

- out degree is n = logN
- correct dimensions in order



# Properties of Some Topologies

| Topology | Degree | Diameter | Ave Dist | Bisection | D (D ave) @ P=1024 | |
|---|---|---|---|---|---|---|
| 1D Array | 2 | N-1 | N / 3 | 1 | huge | |
| 1D Ring | 2 | N/2 | N/4 | 2 | | |
| 2D Mesh | 4 | $2 (N^{1/2} - 1)$ | $2/3\ N^{1/2}$ | $N^{1/2}$ | 63 (21) | |
| 2D Torus | 4 | $N^{1/2}$ | $1/2\ N^{1/2}$ | $2N^{1/2}$ | 32 (16) | |
| k-ary n-cube | 2n | nk/2 | nk/4 | nk/4 | 15 (7.5) | @n=3 |
| Hypercube | n =log N | | n | n/2 | N/2 | 10 (5) |

All have some “bad permutations”

- many popular permutations are very bad for meshes (transpose)
- ramdomness in wiring or routing makes it hard to find a bad one!

# Real Machines

| 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 |

# How Many Dimensions in Network?

n = 2 or n = 3

- Short wires, easy to build
- Many hops, low bisection bandwidth
- Requires traffic locality

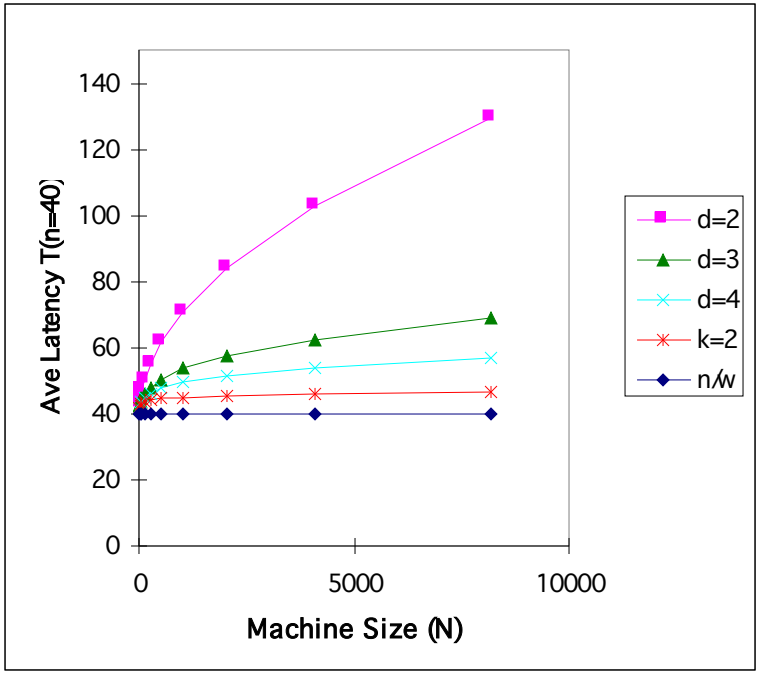
n >= 4

- Harder to build, more wires, longer average length
- Fewer hops, better bisection bandwidth
- Can handle non-local traffic

k-ary d-cubes provide a consistent framework for comparison

- $N = k^d$
- scale dimension (d) or nodes per dimension (k)
- assume cut-through

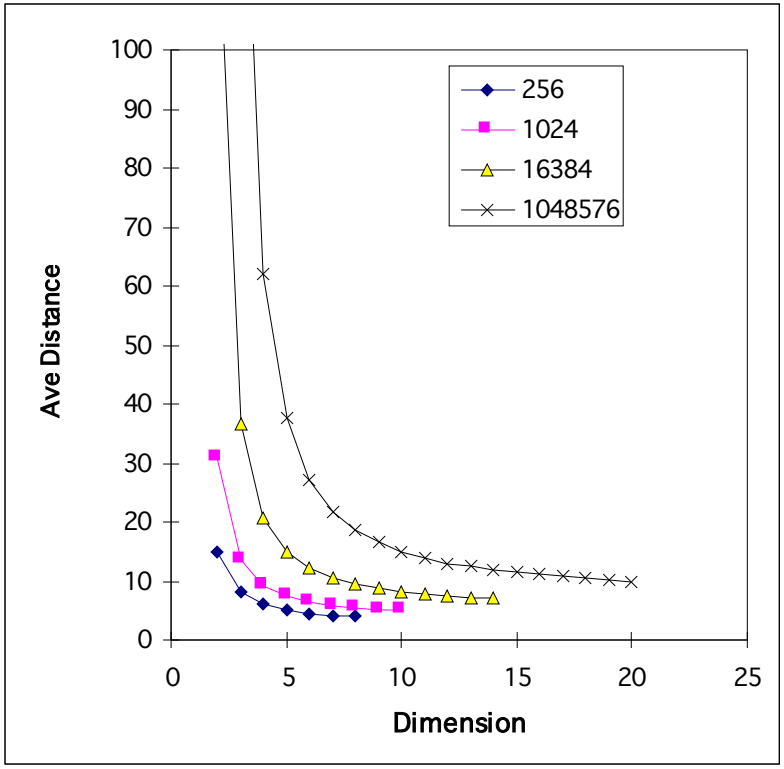
# Traditional Scaling: Latency(P)




Assumes equal channel width

- independent of node count or dimension
- dominated by average distance

# Average Distance



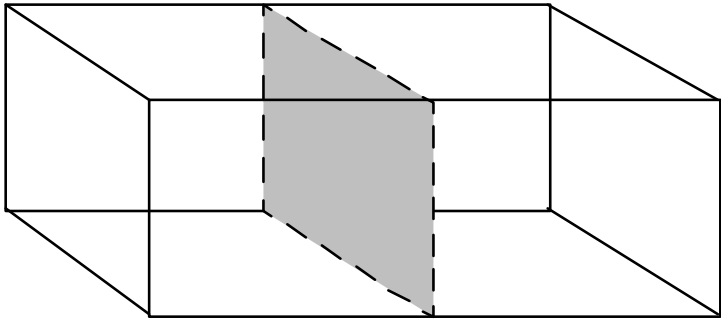

Avg. distance = d (k-1)/2

but, equal channel width is not equal cost!

Higher dimension => more channels

# In the 3-D world

For n nodes, bisection area is $O(n^{2/3})$



For large n, bisection bandwidth is limited to $O(n^{2/3})$

- Dally, IEEE TPDS, [Dal90a]
- For fixed bisection bandwidth, low-dimensional k-ary n-cubes are better (otherwise higher is better)
- i.e., a few short fat wires are better than many long thin wires

# Equal cost in k-ary n-cubes

Equal number of nodes?

Equal number of pins/wires?

Equal bisection bandwidth?

Equal area? Equal wire length?

What do we know?

switch degree: d diameter = d(k-1)

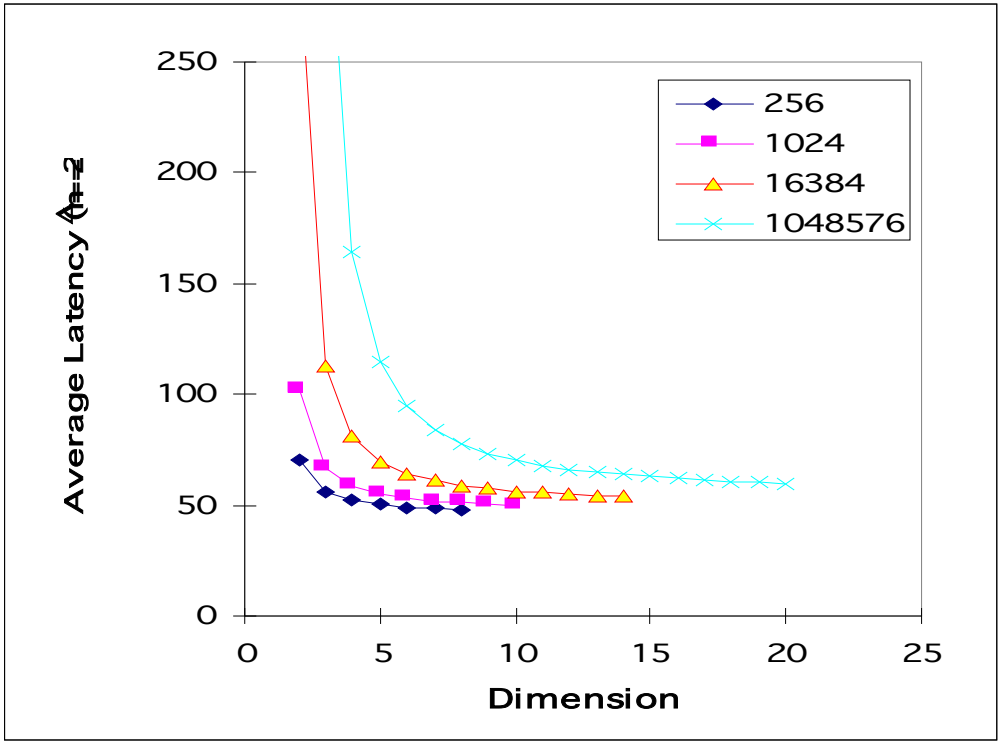
total links = Nd

pins per node = 2wd

bisection = $k^{d-1}$ = N/k links in each directions

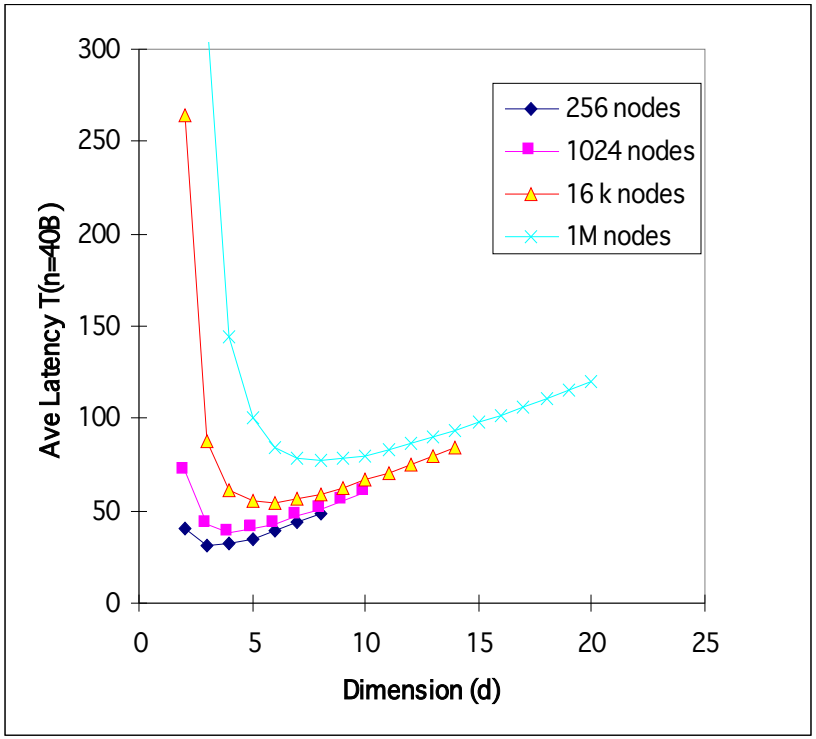
2Nw/k wires cross the middle

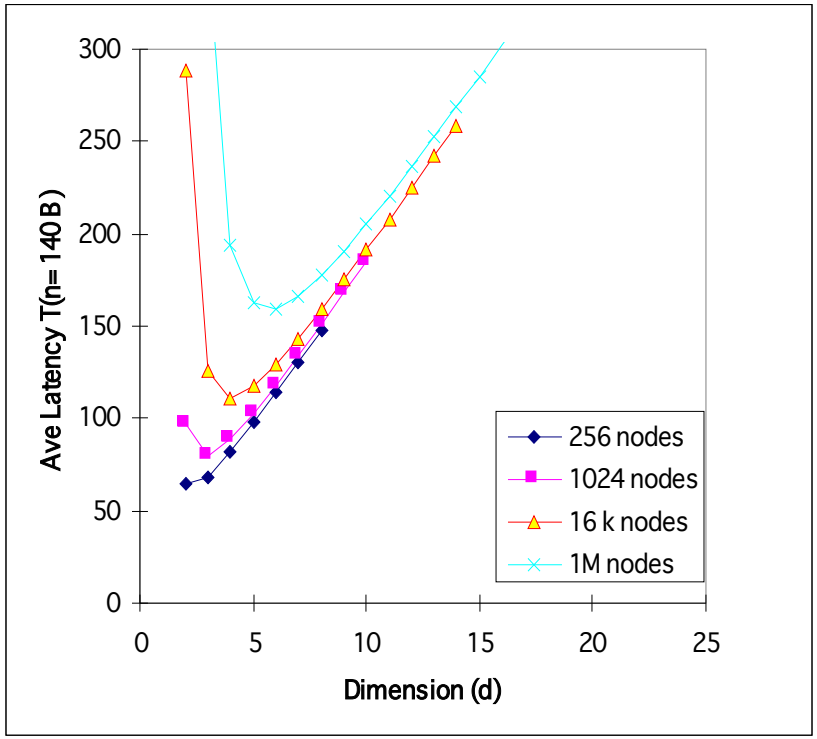
# Latency(d) for P with Equal Width




total links(N) = Nd

# Latency with Equal Pin Count



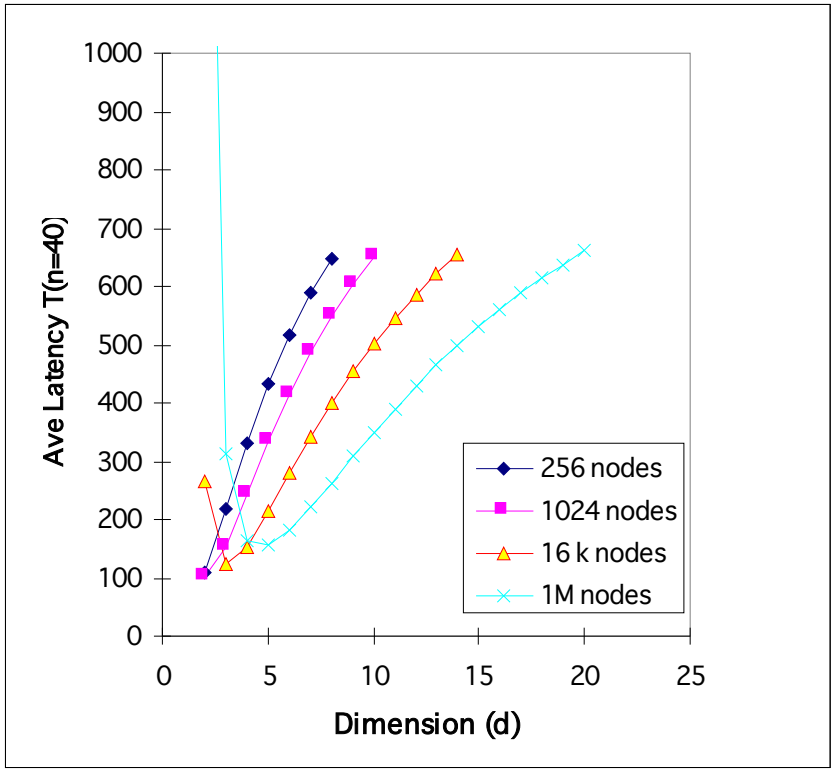




Baseline d=2, has w = 32 (128 wires per node)

fix 2dw pins => w(d) = 64/d

distance up with d, but channel time down

# Latency with Equal Bisection Width



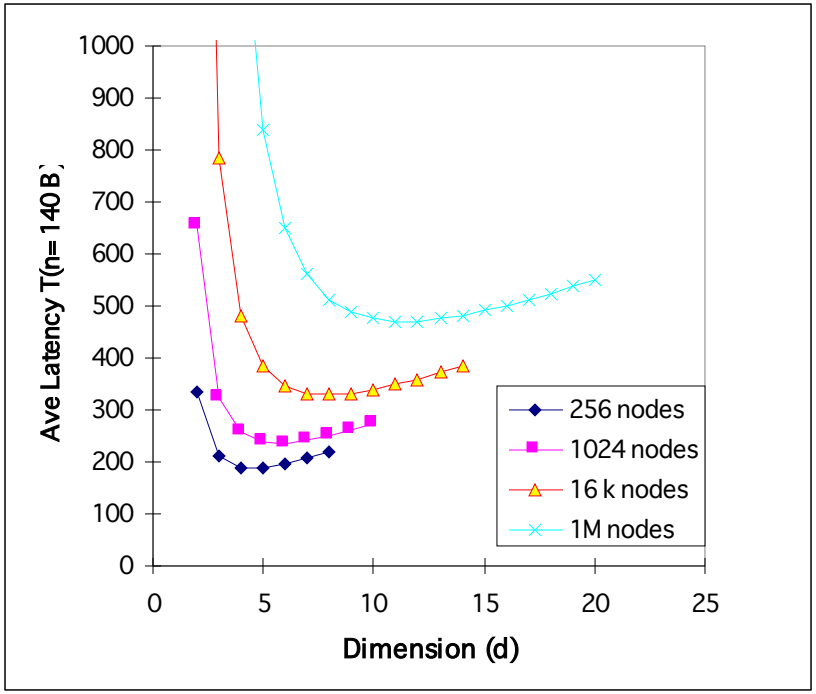

N-node hypercube has N bisection links

2d torus has $2N^{1/2}$

Fixed bisection => w(d) = $N^{1/d} / 2 = k/2$

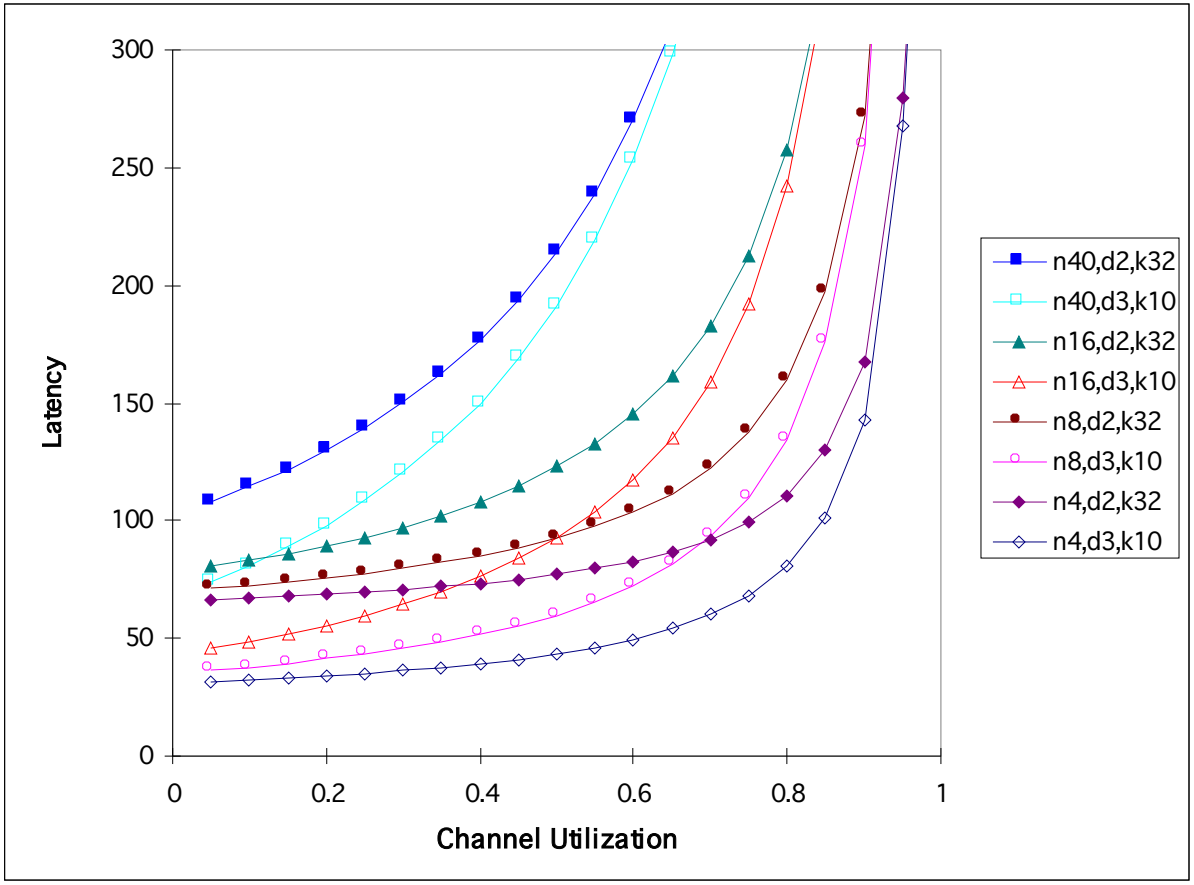
1 M nodes, d=2 has w=512!

# Larger Routing Delay (w/ equal pin)




Dally's conclusions strongly influenced by assumption of small routing delay

# Latency under Contention




Optimal packet size? Channel utilization?

# Topology Summary

Rich set of topological alternatives with deep relationships

Design point depends heavily on cost model

- nodes, pins, area, ...
- Wire length or wire delay metrics favor small dimension
- Long (pipelined) links increase optimal dimension

Need a consistent framework and analysis to separate opinion from design

Optimal point changes with technology

# Routing and Switch Design

Routing

Switch Design

Flow Control

Case Studies

# Routing

Recall: routing algorithm determines

- which of the possible paths are used as routes
- how the route is determined
- R: N x N -> C, which at each switch maps the destination node $n_d$ to the next channel on the route

Issues:

- Routing mechanism
  - arithmetic
  - source-based port select
  - table driven
  - general computation
- Properties of the routes
- Deadlock feee

# Routing Mechanism

need to select output port for each input packet

- in a few cycles

Simple arithmetic in regular topologies

- ex: $\Delta x$, $\Delta y$ routing in a grid
  - west (-x) $\Delta x < 0$
  - east (+x) $\Delta x > 0$
  - south (-y) $\Delta x = 0, \Delta y < 0$
  - north (+y) $\Delta x = 0, \Delta y > 0$
  - processor $\Delta x = 0, \Delta y = 0$

Reduce relative address of each dimension in order

- Dimension-order routing in k-ary d-cubes
- e-cube routing in n-cube

# Routing Mechanism (cont)


Source-based

- message header carries series of port selects
- used and stripped en route
- *CRC? Packet Format?*
- CS-2, Myrinet, MIT Artic

Table-driven

- message header carried index for next port at next switch
  - o = R[i]
- table also gives index for following hop
  - o, I' = R[i ]
- ATM, HPPI

# Properties of Routing Algorithms

Deterministic

- route determined by (source, dest), not intermediate state (i.e. traffic)

Adaptive

- route influenced by traffic along the way

Minimal

- only selects shortest paths

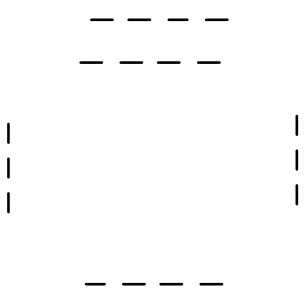
Deadlock free

- no traffic pattern can lead to a situation where no packets mover forward

# Deadlock Freedom

How can it arise?

- necessary conditions:
  - shared resource
  - incrementally allocated
  - non-preemptible
- think of a channel as a shared resource that is acquired incrementally
  - source buffer then dest. buffer
  - channels along a route



How do you avoid it?

- constrain how channel resources are allocated
- ex: dimension order

How do you prove that a routing algorithm is deadlock free

# Proof Technique

Resources are logically associated with channels

Messages introduce dependences between resources as they move forward

Need to articulate possible dependences between channels

Show that there are no cycles in Channel Dependence Graph

- find a numbering of channel resources such that every legal route follows a monotonic sequence

=> no traffic pattern can lead to deadlock

Network need not be acyclic, on channel dependence graph

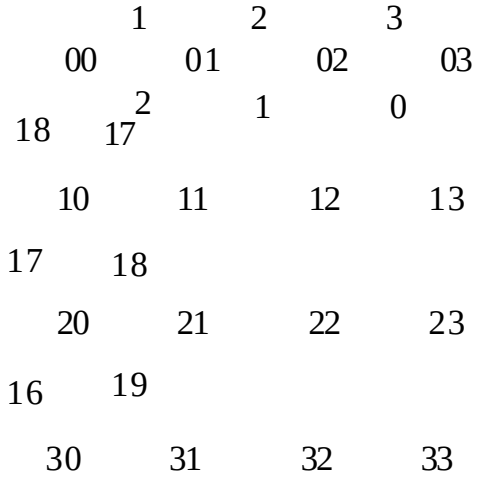
# Example: k-ary 2D array

Theorem: x,y routing is deadlock free

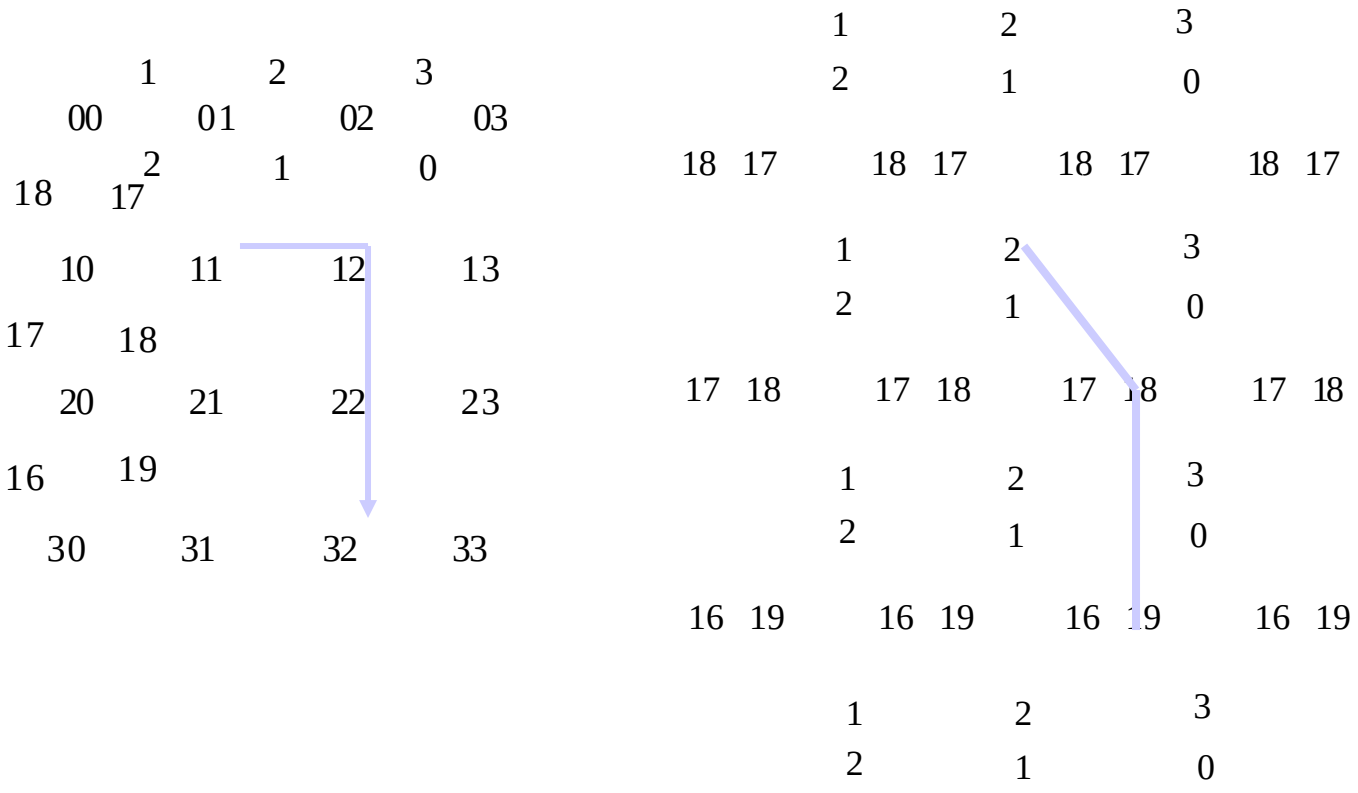
Numbering

- +x channel (i,y) -> (i+1,y) gets i
- similarly for -x with 0 as most positive edge
- +y channel (x,j) -> (x,j+1) gets N+j
- similary for -y channels

Any routing sequence: x direction, turn, y direction is increasing



# Channel Dependence Graph



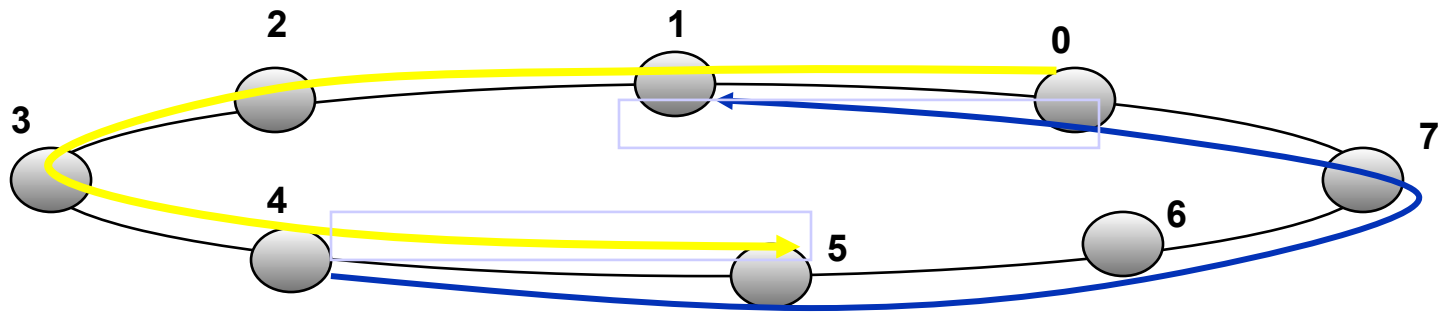
# More examples

Why is the obvious routing on X deadlock free?

- butterfly?
- tree?
- fat tree?

Any assumptions about routing mechanism? amount of buffering?

What about wormhole routing on a ring?



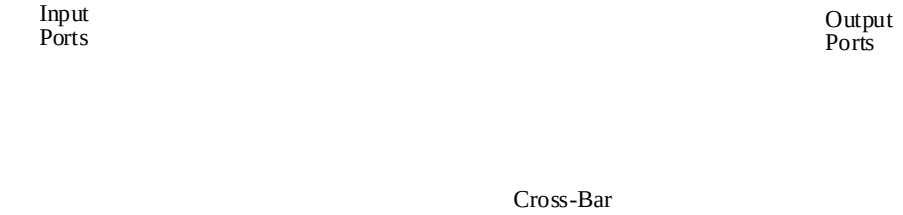
# Deadlock free wormhole networks?

Basic dimension-order routing doesn't work for k-ary d-cubes

- only for k-ary d-arrays (bi-directional)

Idea: add channels!

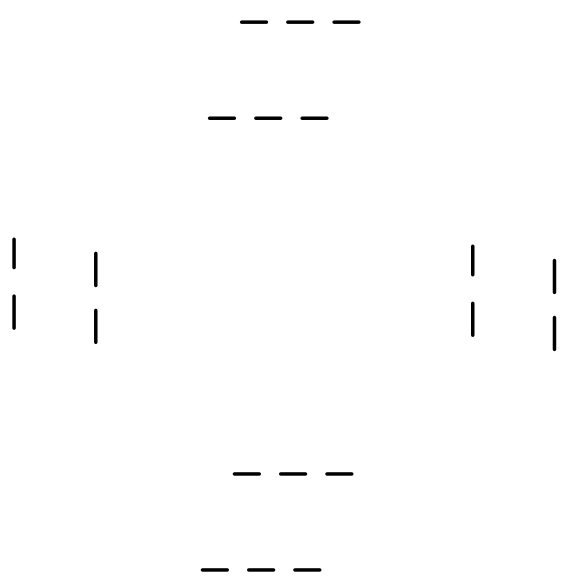
- provide multiple "virtual channels" to break dependence cycle
- good for BW too!




- Don't need to add links, or xbar, only buffer resources

This adds nodes the the CDG, remove edges?

# Breaking deadlock with virtual channels



Packet switches
from lo to hi channel

# Up*-Down* routing

Given any bidirectional network

Construct a spanning tree

Number of the nodes increasing from leaves to roots

UP increase node numbers

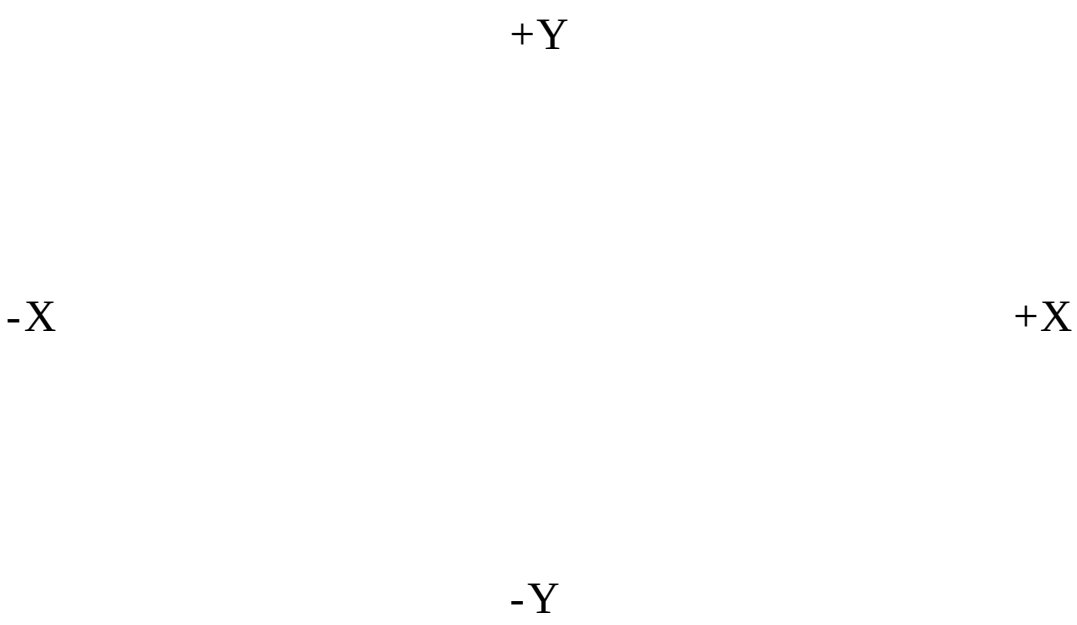
Any Source -> Dest by UP*-DOWN* route

- up edges, single turn, down edges

Performance?

- Some numberings and routes much better than others
- interacts with topology in strange ways

# Turn Restrictions in X,Y




XY routing forbids 4 of 8 turns and leaves no room for adaptive routing

Can you allow more turns and still be deadlock free

# Minimal turn restrictions in 2D

+y

-x

+x

West-first

north-last

-y

negative first

# Example legal west-first routes

Can route around failures or congestion

Can combine turn restrictions with virtual channels

# Adaptive Routing

R: C x N x Σ -> C

Essential for fault tolerance

- at least multipath

Can improve utilization of the network

Simple deterministic algorithms easily run into bad permutations

Fully/partially adaptive, minimal/non-minimal

Can introduce complexity or anomolies

Little adaptation goes a long way!

# Routing and Switch Design Summary

Routing Algorithms restrict the set of routes within the topology

- simple mechanism selects turn at each hop
- arithmetic, selection, lookup

Deadlock-free if channel dependence graph is acyclic

- limit turns to eliminate dependences
- add separate channel resources to break dependences
- combination of topology, algorithm, and switch design

Deterministic vs adaptive routing

Switch design issues

- input/output/pooled buffering, routing logic, selection logic

Flow control

Real networks are a ‘package’ of design choices