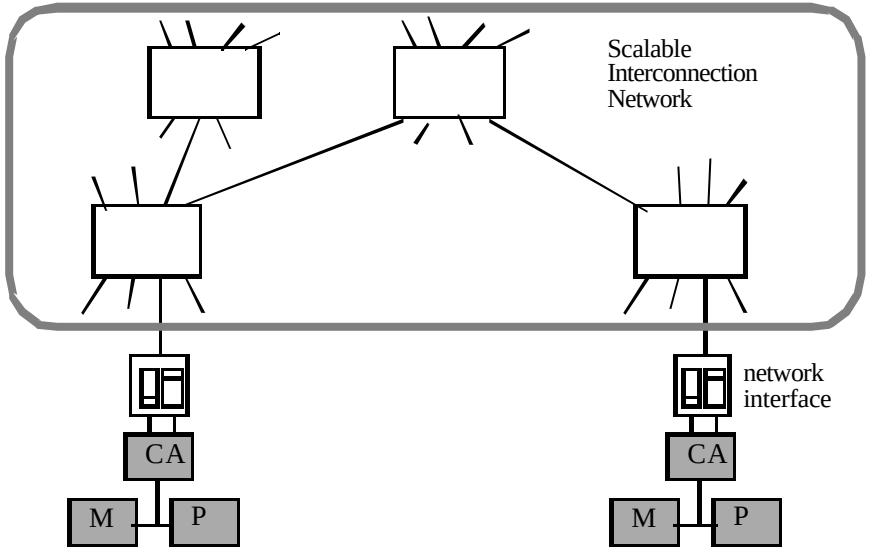
Interconnection Network Design

- Interconnection networks: what holds our parallel machines together - at the core of parallel computer arch.
- Shares basic concept with LAN/WAN, but very different trade-offs due to very different time scale/requirements



Interconnection Network Design

- Considerations and trade-offs at many levels
  - Topology (elegant mathematical structure)
  - Deep relationships to algorithm structure
  - Managing many traffic flows
  - Electrical / Optical link properties
- Little consensus
  - interactions across levels
  - Performance metrics?
  - Cost metrics?
  - Workload?

=> need holistic understanding

Requirements for Interconnect Design

- Communication-to-computation ratio

  => bandwidth that must be sustained for given computational rate

  – traffic localized or dispersed?

  – bursty or uniform?

- Programming Model

  – protocol

  – granularity of transfer

=> job of an interconnection network is to transfer information from source node to dest. node in support of network transactions that realize the programming model

  – latency as small as possible

  – as many concurrent transfers as possible

  – cost as low as possible

Basic Definitions

- Network interface
- Links
  - bundle of wires or fibers that carries a signal
  - *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
- Switches
  - connects fixed number of input channels to fixed number of output channels

Some Formal Definitions

- Interconnection 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 channel followed by a message is a *route*

- Think streets and intersections

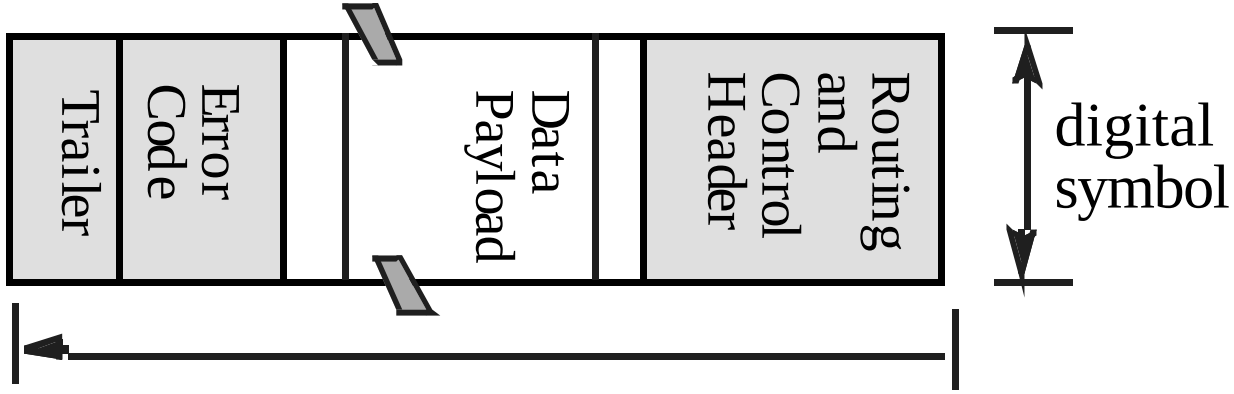
What characterizes an interconnection net?

- 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 (e.g. 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?

Properties of a Topology

- *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



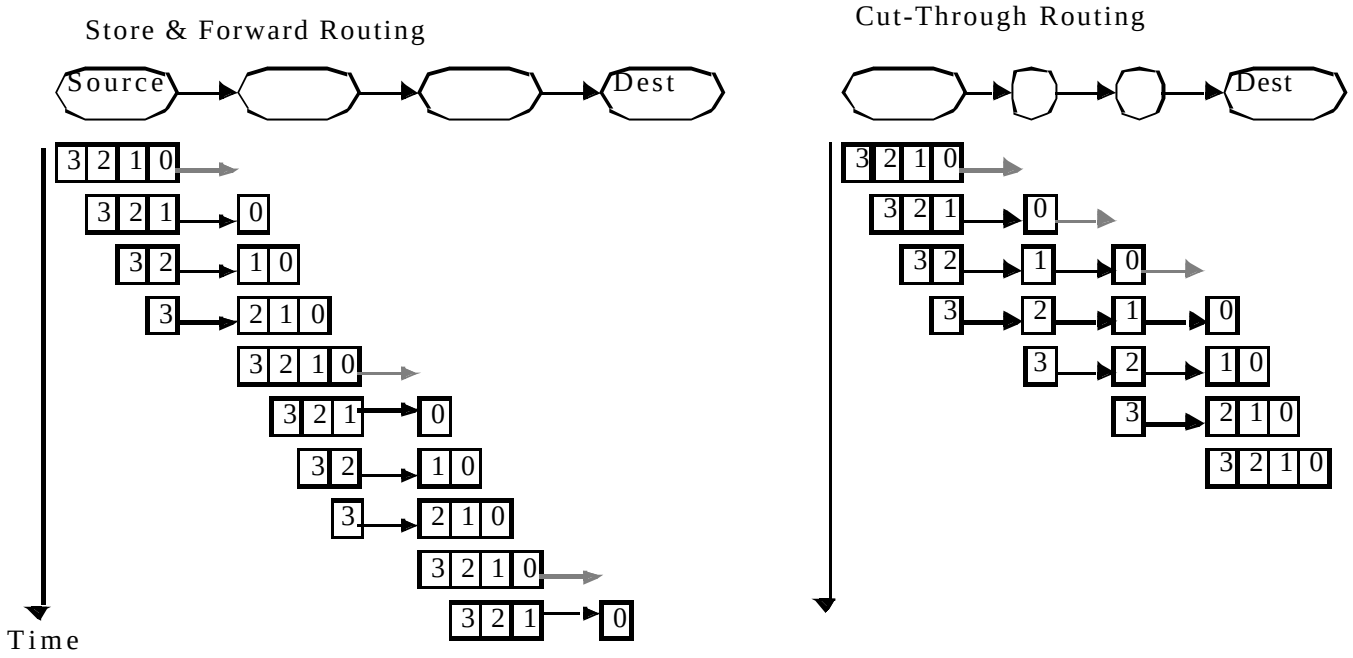
Sequence of symbols transmitted over a channel

- Two basic mechanisms for abstraction (much shallower than IP for example)
  - encapsulation
  - fragmentation

Basic Communication Performance: Latency

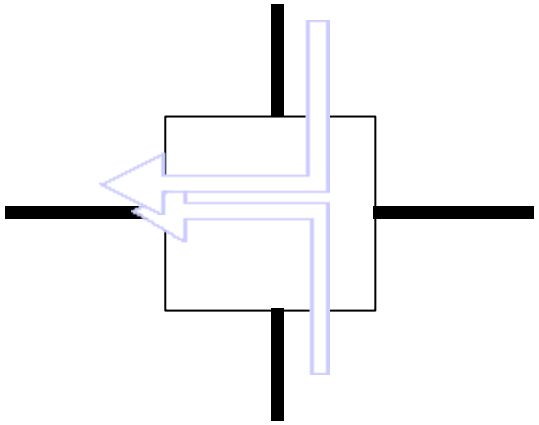
- $Time(n)_{s\text{-}d}$ = overhead + routing delay + channel occupancy + contention delay
- occupancy = $(n + n_e) / b$
  - where n= size of data, n_e= size of packet overhead, b= bandwidth=f*W
- Routing delay
  - function of routing distance and switch delay
  - depends on topology, routing algorithm, communicating nodes, switching strategy
- Contention
  - Given channel can only be occupied by one message
  - Affected by topology, switching strategy, routing algorithm

Store&Forward vs Cut-Through Routing



- h(n/b + D) vs n/b + h D
  - where h= routing distance, D= switch delay or routing delay per hop
- what if message can be fragmented?
- wormhole vs virtual cut-through

Contention



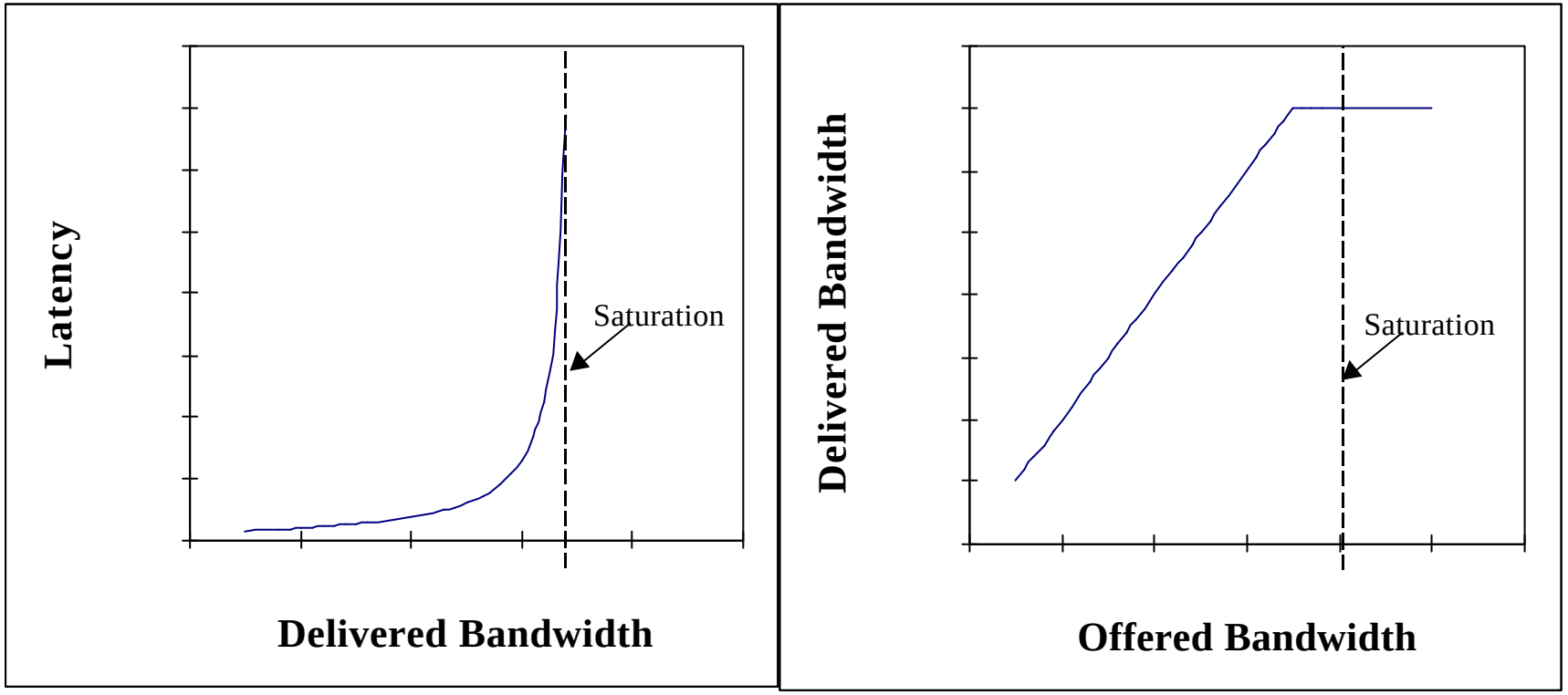
- Two packets trying to use the same link at same time
  - limited buffering
  - drop?
- Most parallel mach. networks block in place
  - link-level flow control
  - tree saturation
- Closed system

Basic Communication Performance: Bandwidth

- What affects local bandwidth?
  - packet density $b \times n/(n + n_E)$
  - routing delay $b \times n / (n + n_E + w\Delta)$
  - contention
- Aggregate bandwidth
  - bisection bandwidth

    -> sum of bandwidth of smallest set of links that partition the network
  - total bandwidth of all the channels: Cb
  - suppose N hosts issue packet every M cycles with average distance h
    - each msg occupies h channels for l = n/w cycles each
    - link utilization $\rho = Cb/(Nhlb/M) = MC/Nhl < 1$
    - C/N channels available per node

Saturation

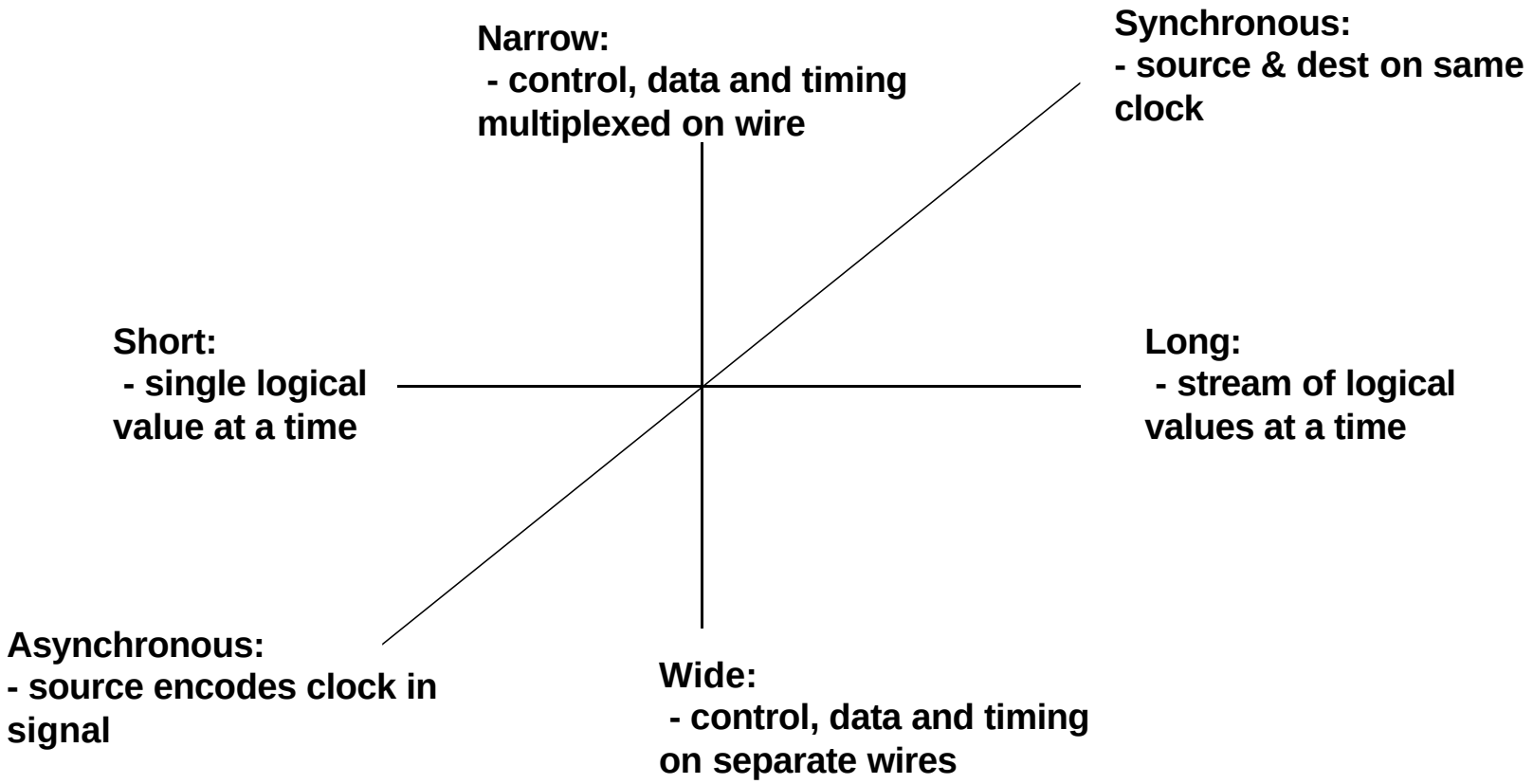

Latency
Saturation
Delivered Bandwidth
Delivered Bandwidth
Saturation
Offered Bandwidth

Organizational Structure

- Processors
  - datapath + control logic
  - control logic determined by examining register transfers in the datapath
- Networks
  - links

    -> Cable of one or more wires/fibers with connectors at the ends attached to switches or interfaces
  - switches
  - network interfaces

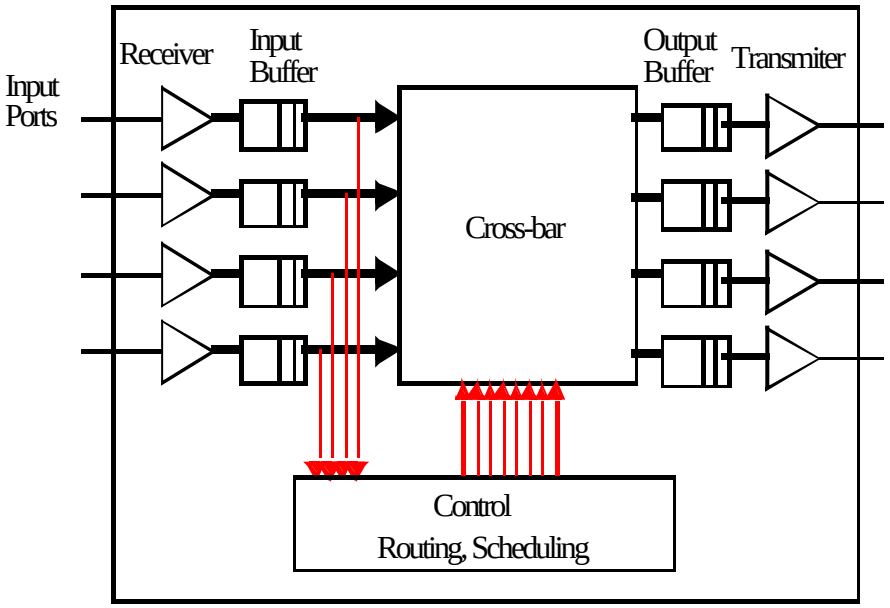
Link Considerations



Example: Cray MPPs

- T3D: Short, Wide, Synchronous (300 MB/s)
  - 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, LVDS
  - 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
- Cost = f(length, width) ?

Switches



- Output ports
  - transmitter (typically drives clock and data)
- Input ports
  - synchronizer aligns data signal with local clock domain
  - essentially FIFO buffer
- Crossbar
  - connects each input to any output
  - degree limited by area or pinout

- Buffering
- Control logic
  - complexity depends on routing logic and scheduling algorithm
  - determine output port for each incoming packet
  - arbitrate among inputs directed at same output

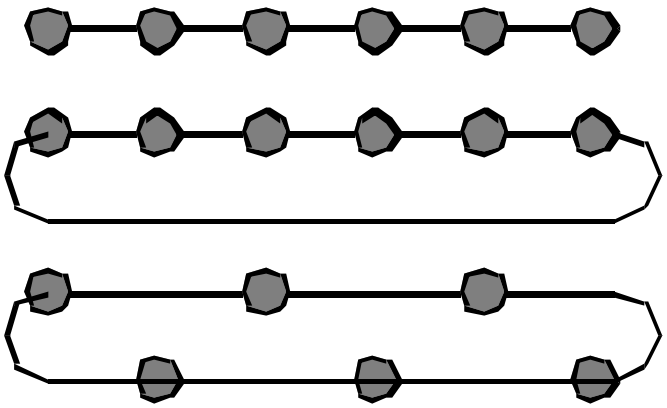
- Details later...

Interconnection Topologies

- Classes of networks scaling with N
- Logical Properties:
  - distance, degree
- Physical properties
  - length, width

- Fully connected network
  - diameter = 1
  - degree = N
  - cost?
    - bus => O(N), but BW is O(1) - actually worse
    - crossbar => $O(N^2)$ for BW O(N)
- VLSI technology determines switch degree

Linear Arrays and Rings



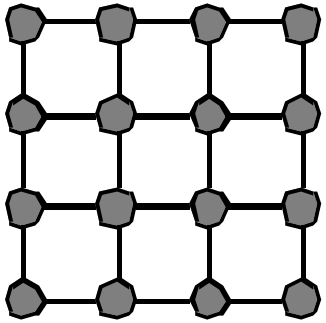
Linear Array

Torus

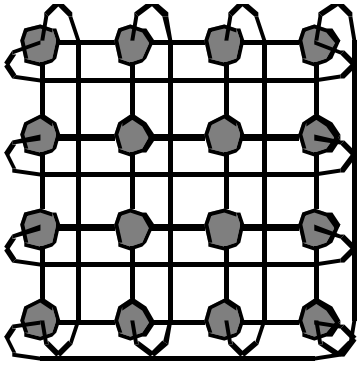
Torus arranged to use short wires

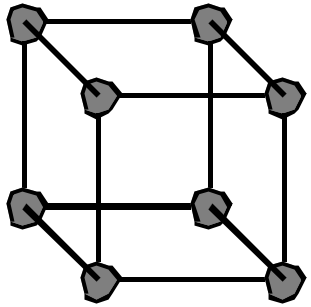
- Linear Array
  - Diameter?
  - Average Distance?
  - Bisection bandwidth?
  - Route A -> B given by relative address R = B-A
- Torus?
- Examples: FDDI, SCI, FiberChannel Arbitrated Loop, KSR1

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: $N = k^d$
  - $k = \sqrt[d]{N}$
  - described by *d*-vector of radix k coordinate
- *d*-dimensional *k*-ary torus (or *k*-ary *d*-cube)?

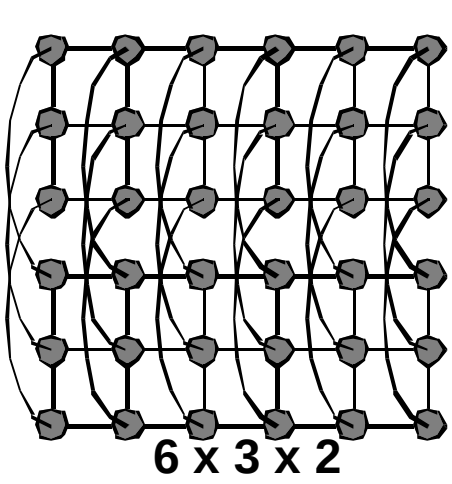
Multidimensional Meshes and Tori: Properties

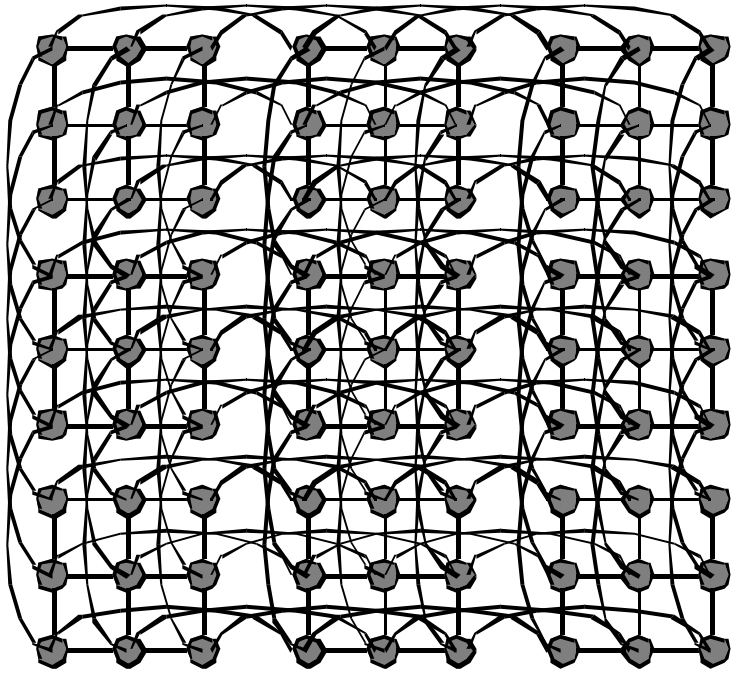
- Routing
  - relative distance: $R = (b_{d-1} - a_{d-1}, \ldots, b_0 - a_0)$
  - traverse $r_i = b_i - a_i$ hops in each dimension
  - *dimension-order routing*
- Average Distance
  - d x 2k/3 for mesh
  - dk/2 for cube
- Degree?
- Bisection bandwidth?
  - k^{d-1} bidirectional links

Wire Length?

Partitioning?

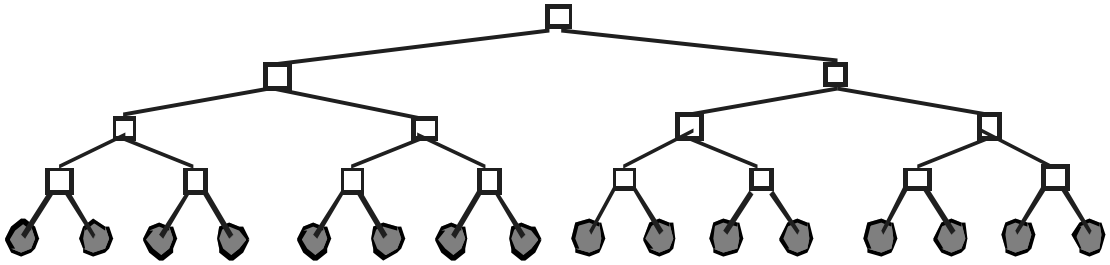
Multidimensional Meshes and Tori: Embeddings in lesser dimensions

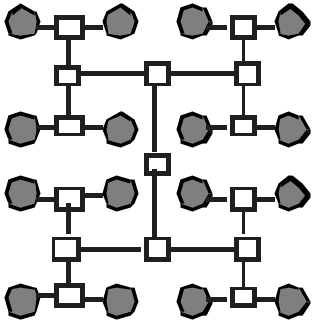




- Embed multiple logical dimension in one physical dimension using long wires

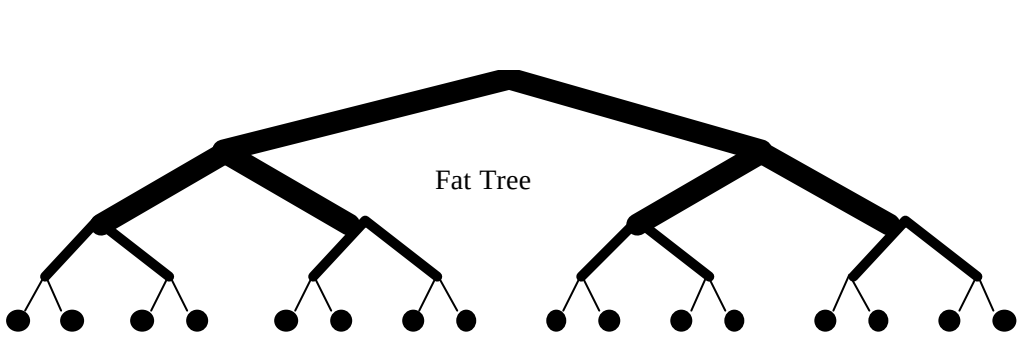
Trees

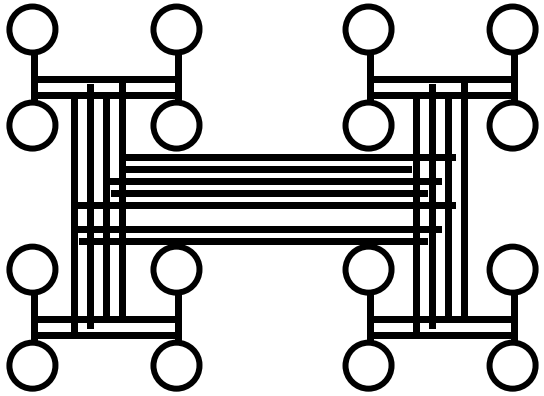




- Diameter and average distance logarithmic
  - k-ary tree, height d = $\log_k$ N
  - address specified d-vector of radix k coordinates describing path down from root
- Fixed degree
- Route up to common ancestor and down
  - R = B xor A
  - let i be position of most significant 1 in R, route up i+1 levels
  - down in direction given by low i+1 bits of B
- H-tree space is O(N) with O($\sqrt{N}$) long wires
- Bisection BW?

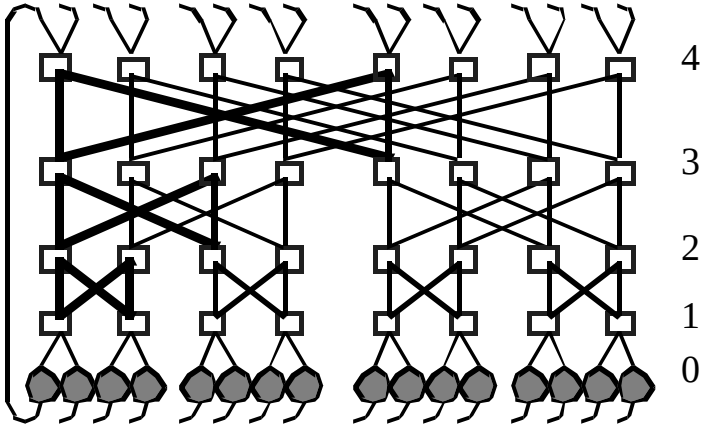
Fat-Trees



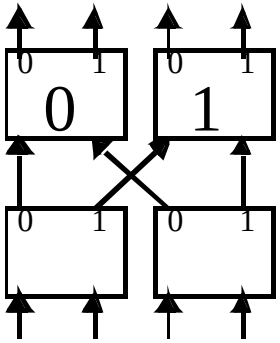


- Fatter links (really more of them) as you go up, so bisection BW scales with N

Butterflies



16 node butterfly

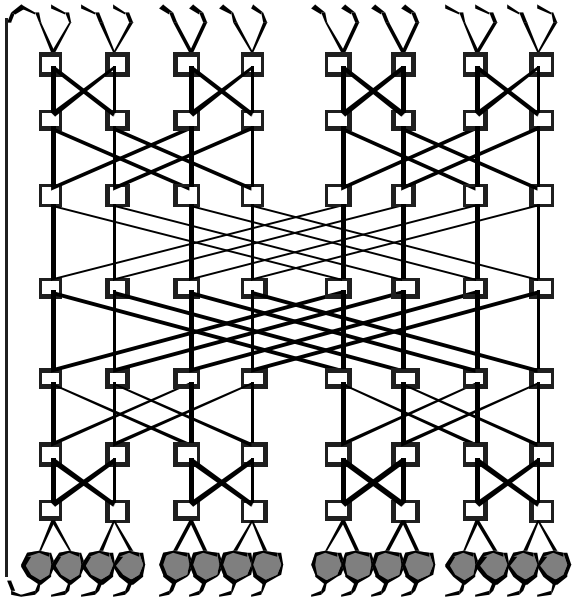


building block

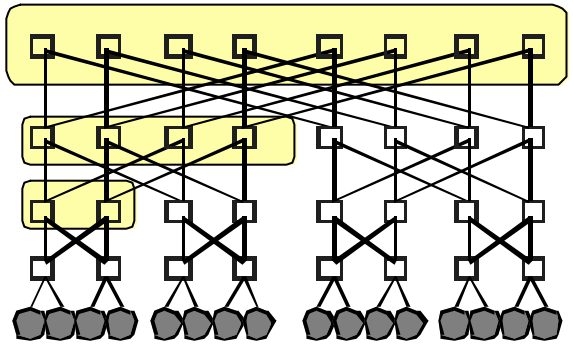
- Tree with lots of roots!
- N log N (actually N/2 x logN)
- Exactly one route from any source to any dest
- R = A xor B, at level i use ‘straight’ edge if r_i=0, otherwise cross edge
- Bisection N/2 vs $n^{(d-1)/d}$ (d-mesh) vs 1 (tree)

Benes network and Fat Tree

16-node Benes Network (Unidirectional)



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



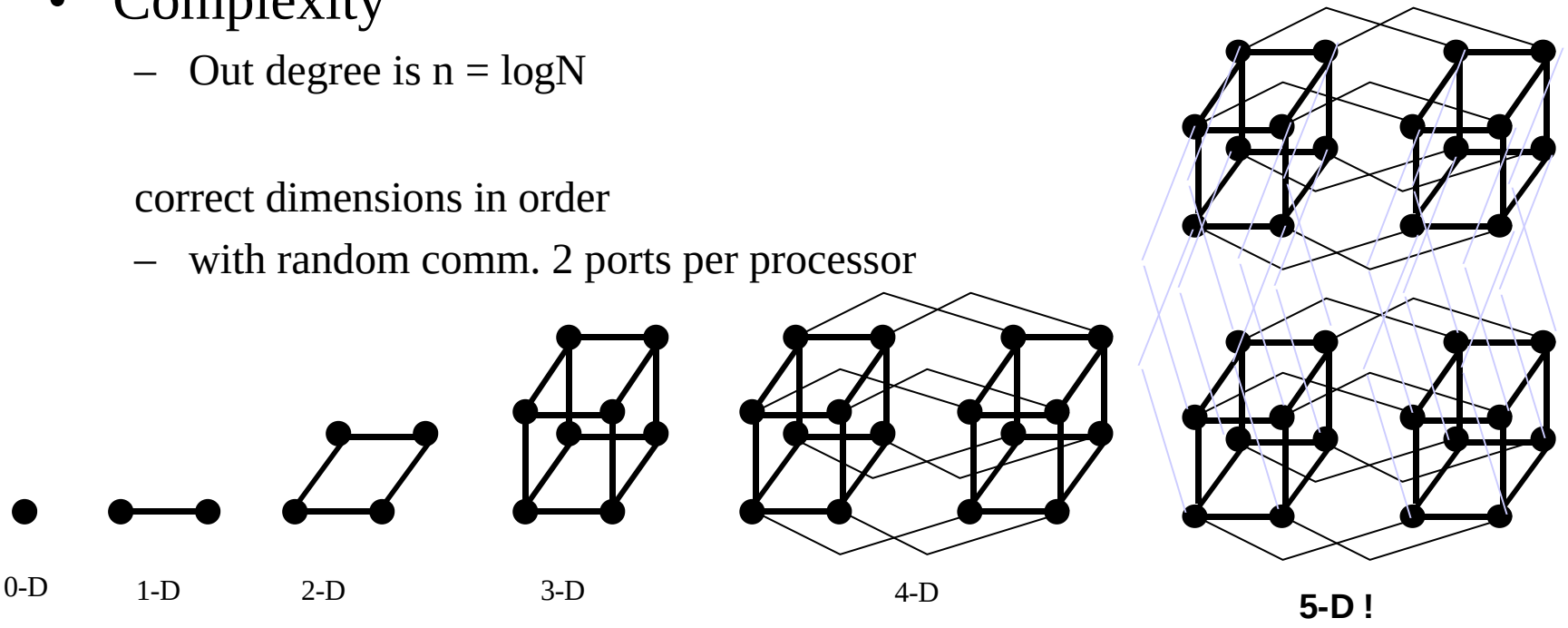
- Back-to-back butterfly can route all permutations
  - off line
- What if you just pick a random mid point?

Hypercubes

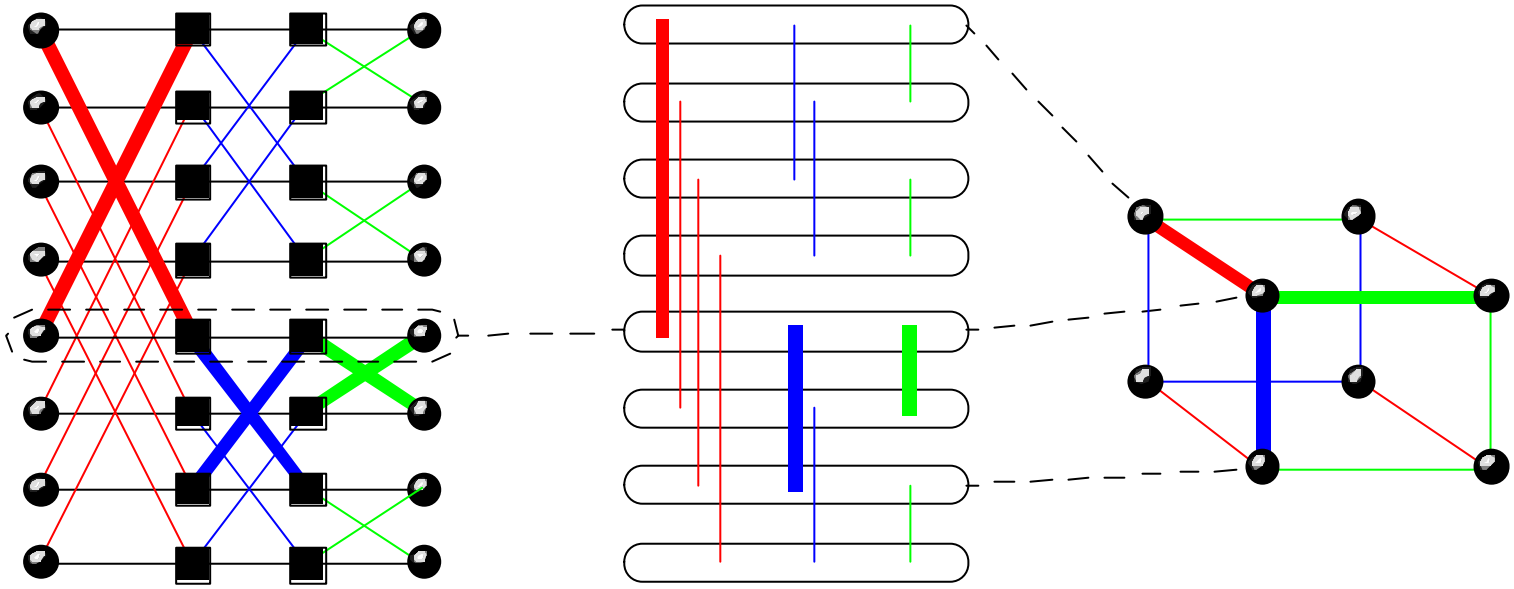
- Also called binary n-cubes. # of nodes = N = 2^n.
- O(logN) Hops
- Good bisection BW
- Complexity
  - Out degree is n = logN

correct dimensions in order

- with random comm. 2 ports per processor



ButterFlies & Hypercubes



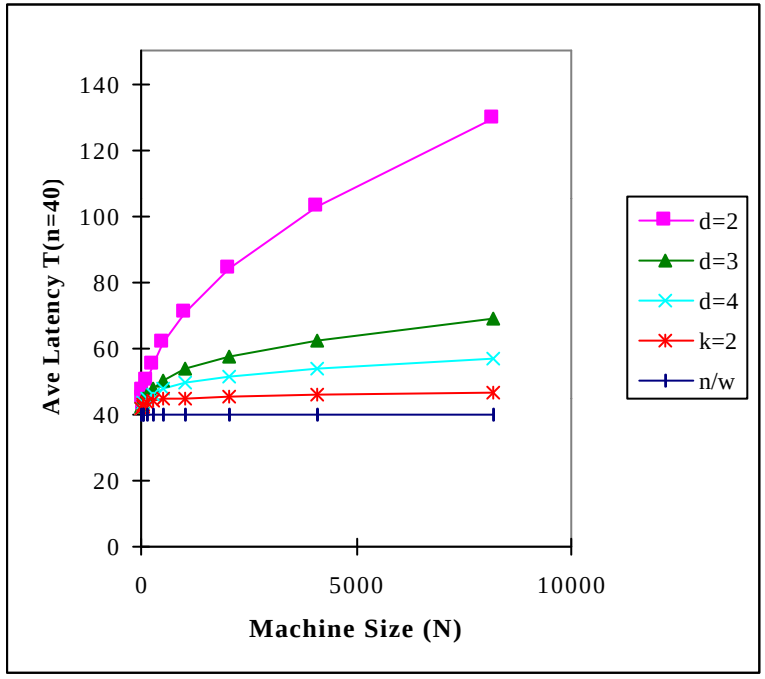
- Wiring is isomorphic
- Except that Butterfly always takes log n steps

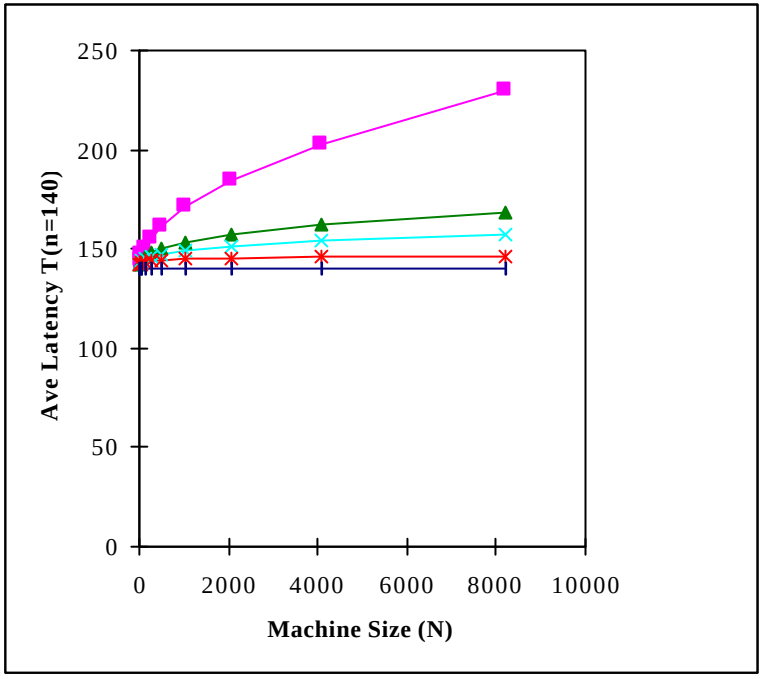
Performance Issues in Topology

- d = 2 or d = 3
  - Short wires, easy to build
  - Many hops, low bisection bandwidth
  - Requires traffic locality
- d >= 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

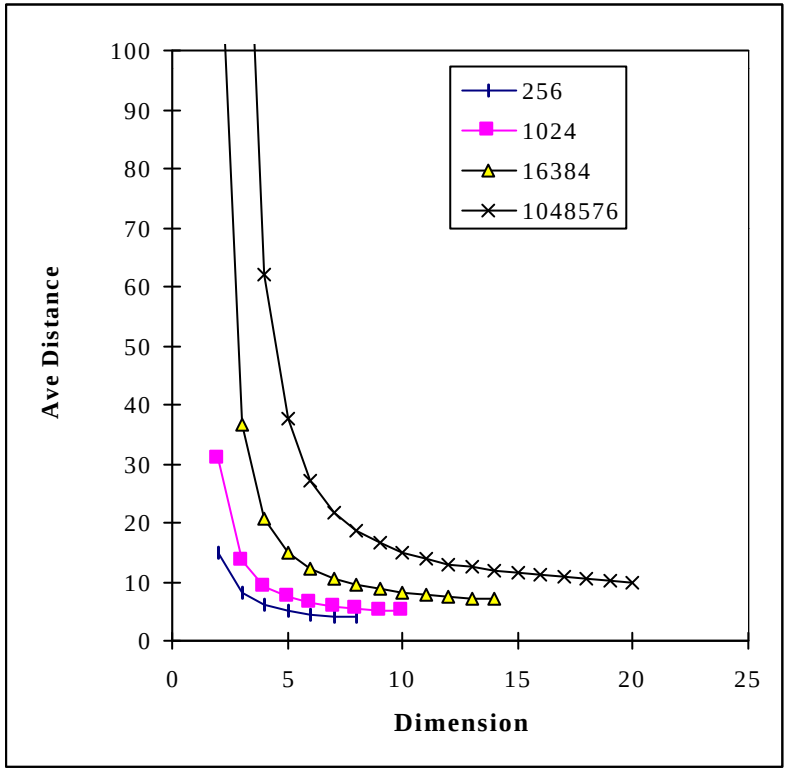
Traditional Scaling: Latency(P)





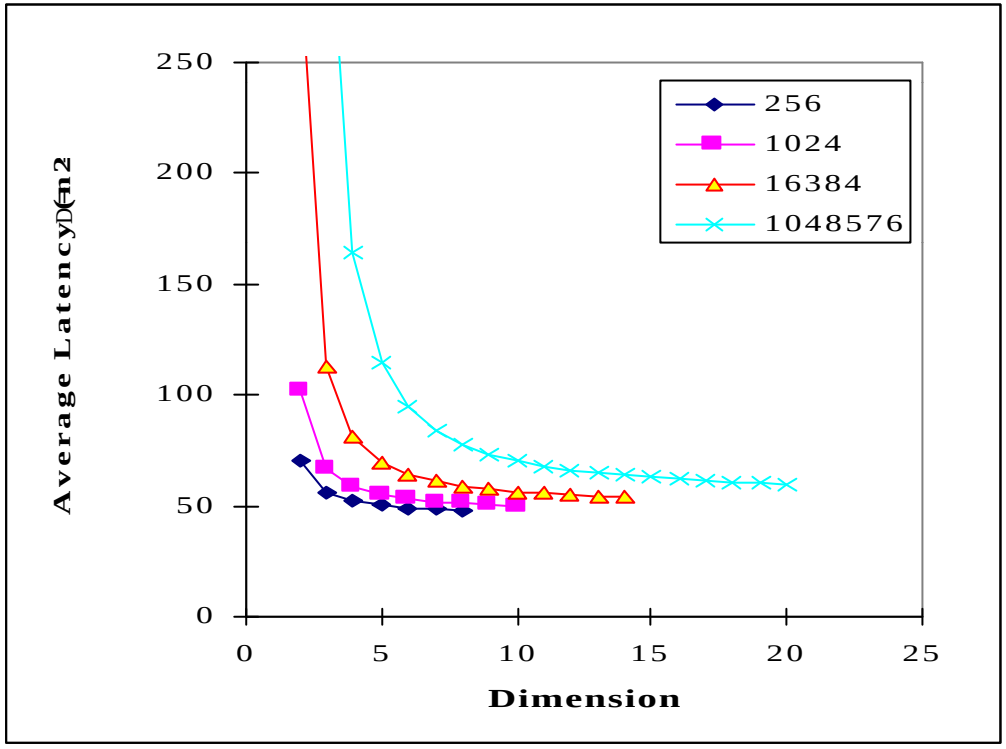
- Assumes equal channel width
  - independent of node count or dimension
  - dominated by average distance

Average Distance



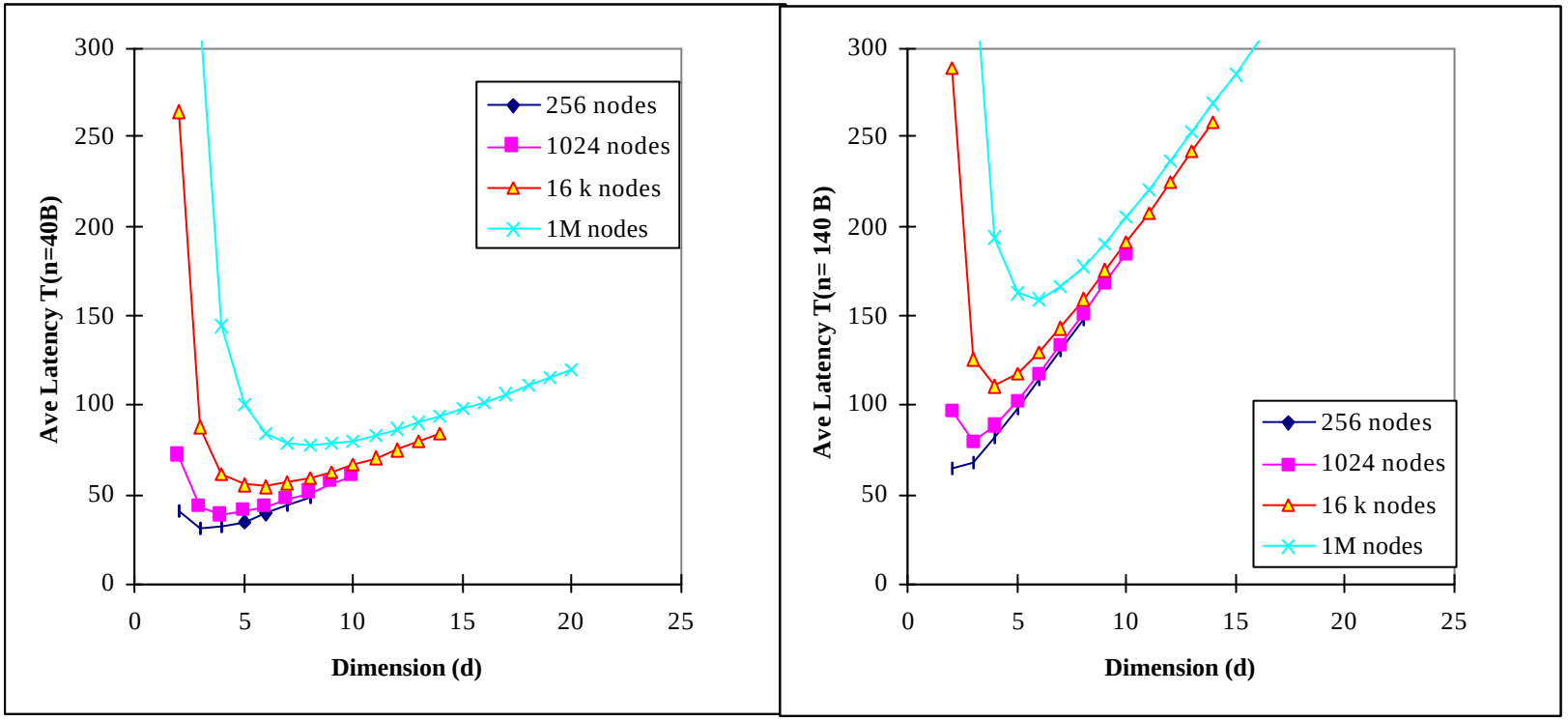
ave dist = d (k-1)/2

Latency(d) for P with Equal Width



- but, equal channel width is not equal cost!
- Higher dimension => more channels

Latency with Equal Pin Count

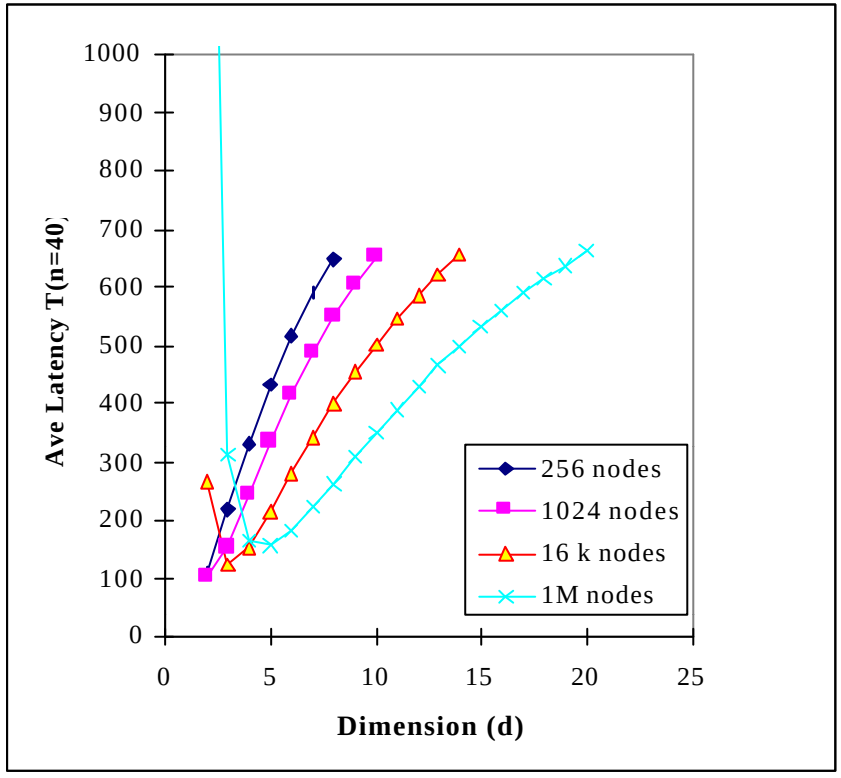


- Baseline d=2, has w = 32 (128 wires per node)
- fix 2dw pins => w(d) = 64/d
- distance up with lower d, but channel time down

Real Machine Channel Width

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

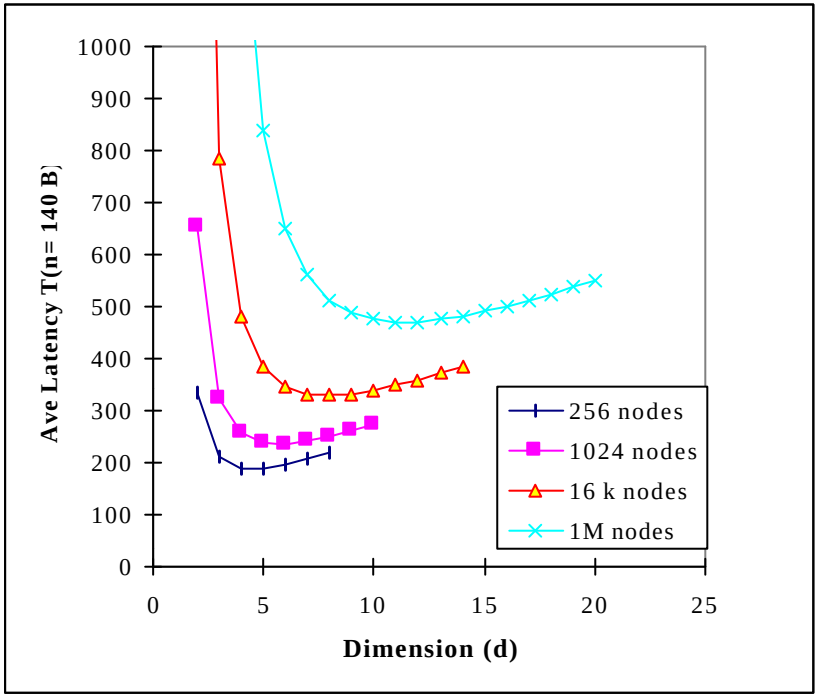
Latency with Equal Bisection Width



- N-node hypercube has N bisection links
- 2d torus has $2N^{1/2}$
- Fixed bisection => w(d) = $N^{1/d} / 2 = k/2$

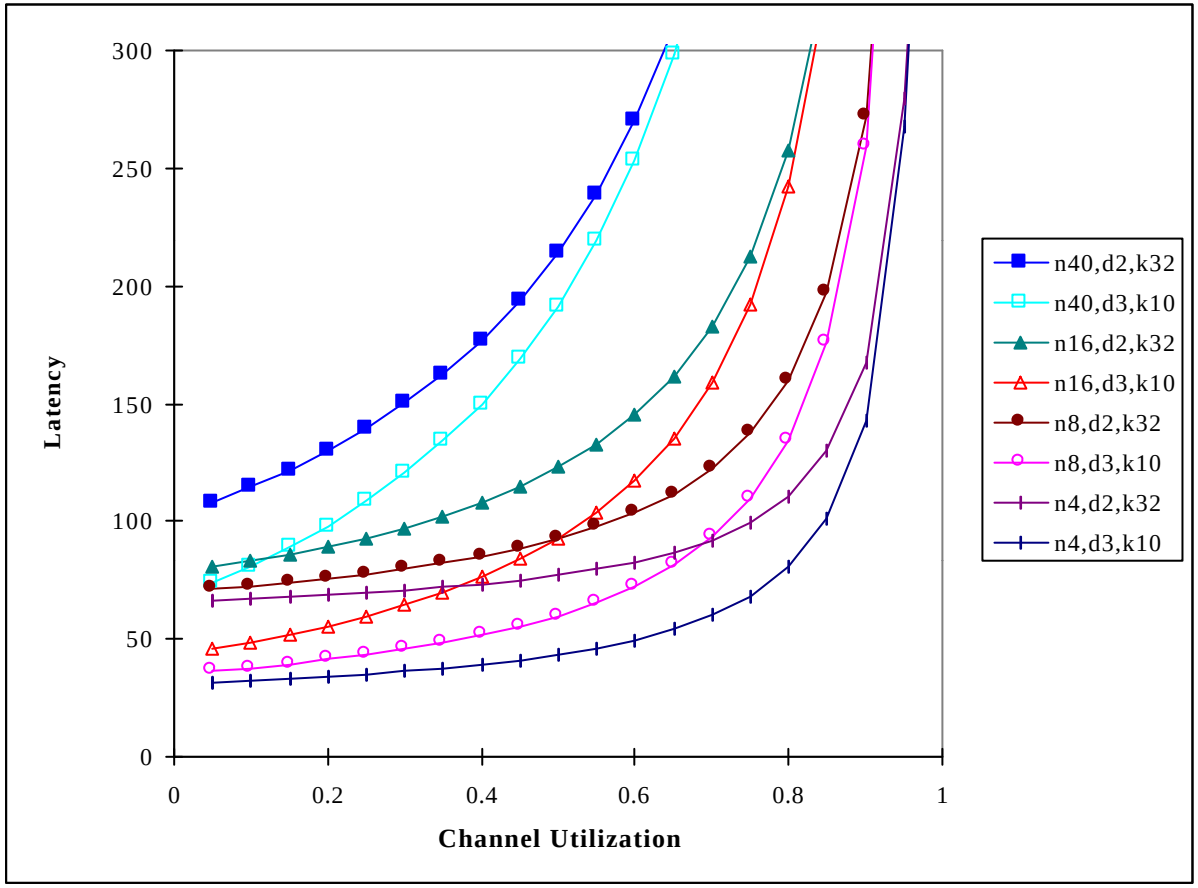
- 1 M nodes, d=2 has w=512!

Larger Routing Delay (w/ equal pin)



- If routing delay=20, optimal point shifts to higher dimension

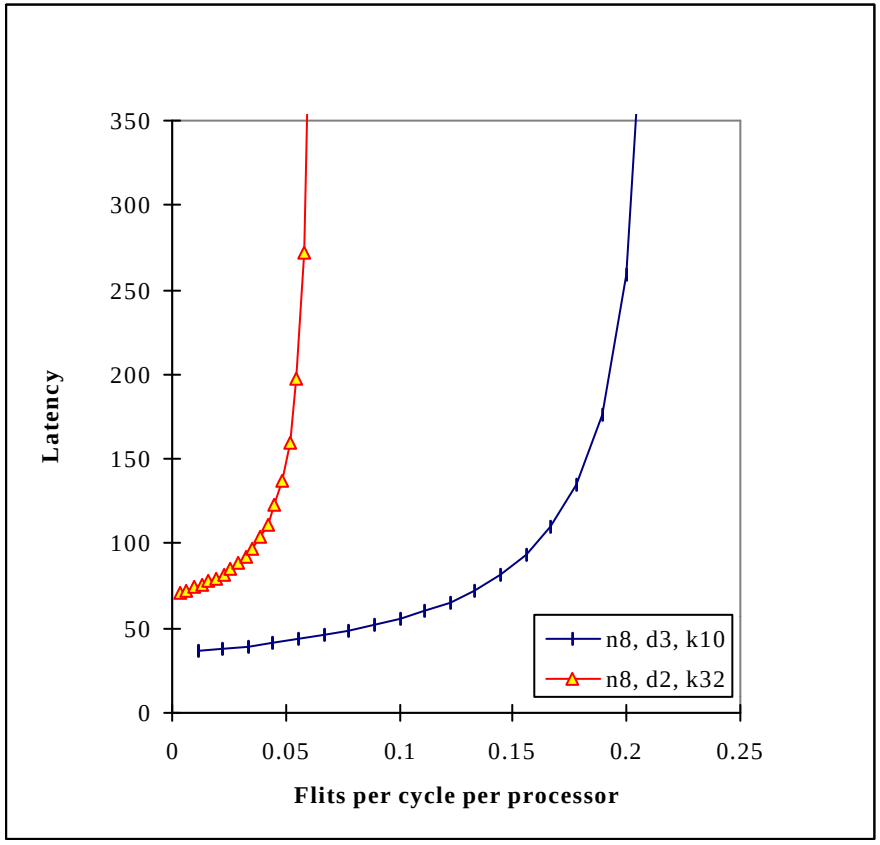
Latency under Contention



- Dimension has no effect?

Phits per Cycle (Delivered Bandwidth)

- higher degree network has larger available bandwidth



Summary of Performance/Topology

- Rich set of topological alternatives with deep relationships
- Design point depends heavily on cost model
  - nodes, pins, area, ...
- Also, wire delay comes into effect
  - Wire length or wire delay metrics favor small dimension
  - Long (pipelined) links increase optimal dimension
- Optimal point changes with technology

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: Δx, Δy 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)

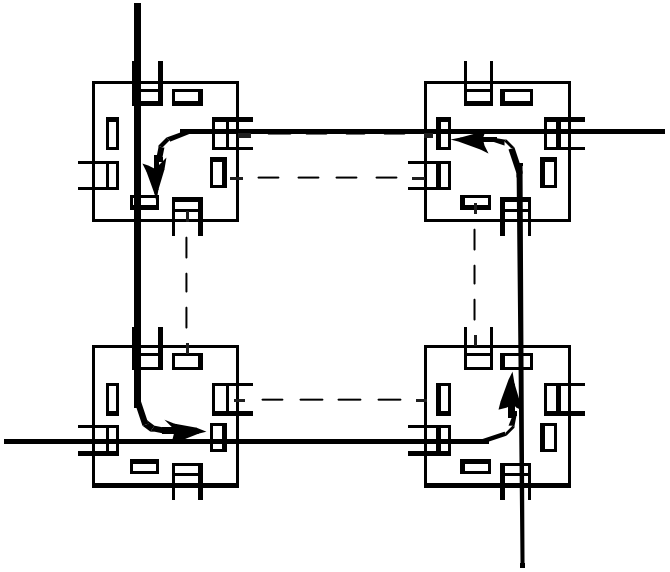
- Source-based
  - message header carries series of port selects
  - used and stripped en route
  - All route computation in the host nodes. Disadv.?
  - CS-2, Myrinet, MIT Artic
- Table-driven
  - message header carried index for next port at next switch
    - o = R[i]
  - table also gives index for 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

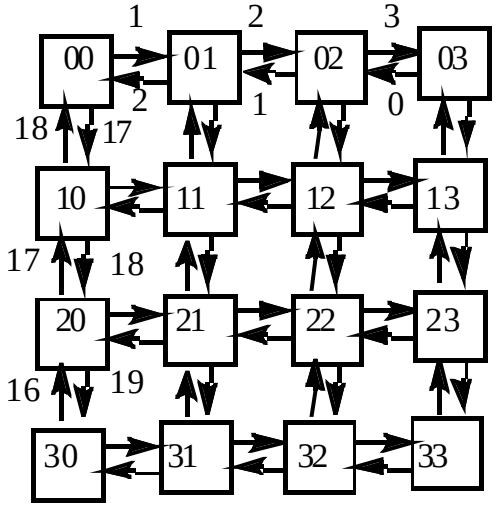
- Resources are logically associated with channels
- Messages introduce dependences between resources as they move forward
- Need to articulate the possible dependences that can arise between channels
- Show that there are no cycles in Channel Dependence Graph
  - find a numbering of channel resources such that every legal route follows a monotonic sequence

  => no traffic pattern can lead to deadlock

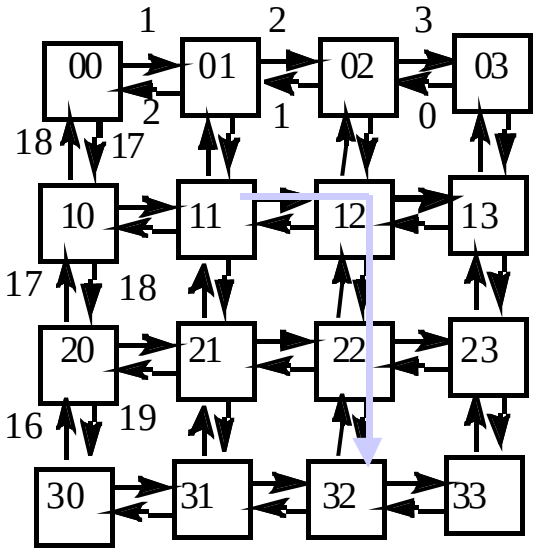
- network need not be acyclic, on channel dependence graph

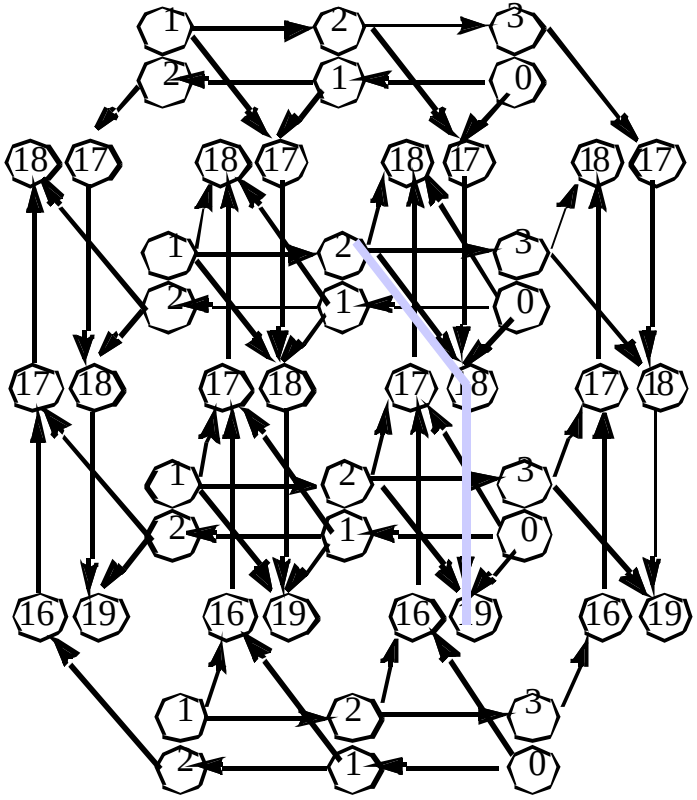
Example: k-ary 2D array

- The Δx,Δy 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



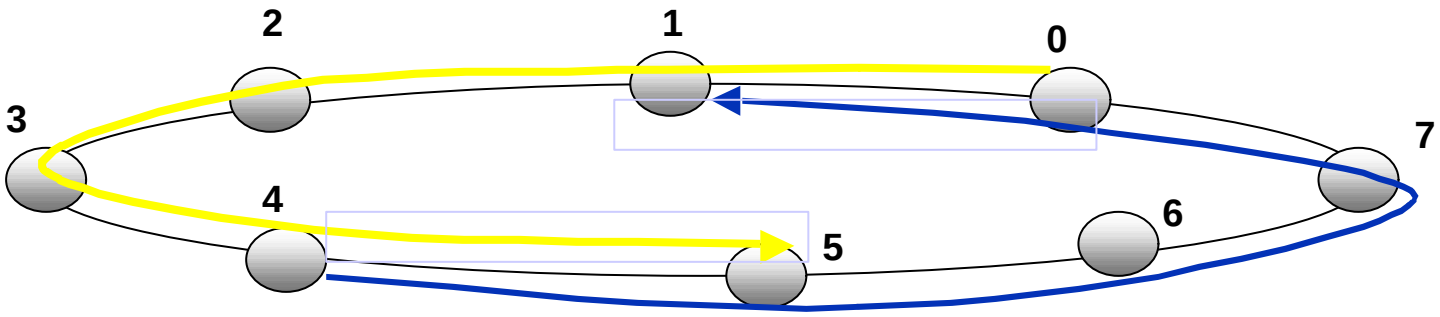
Channel Dependence Graph





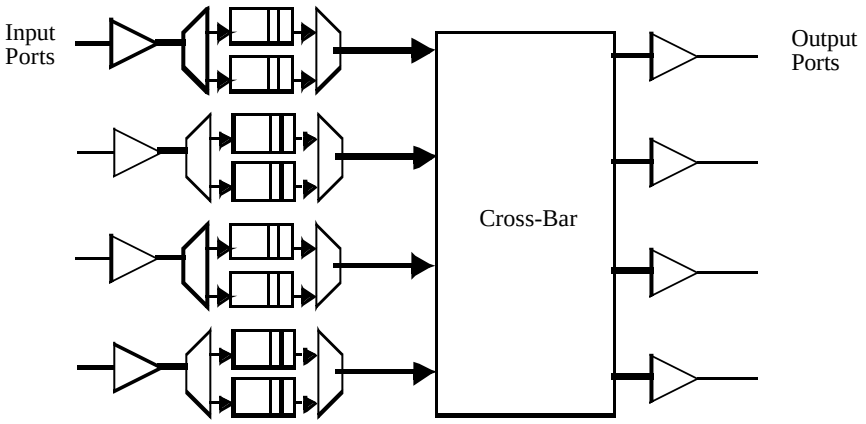
More Examples

- 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?



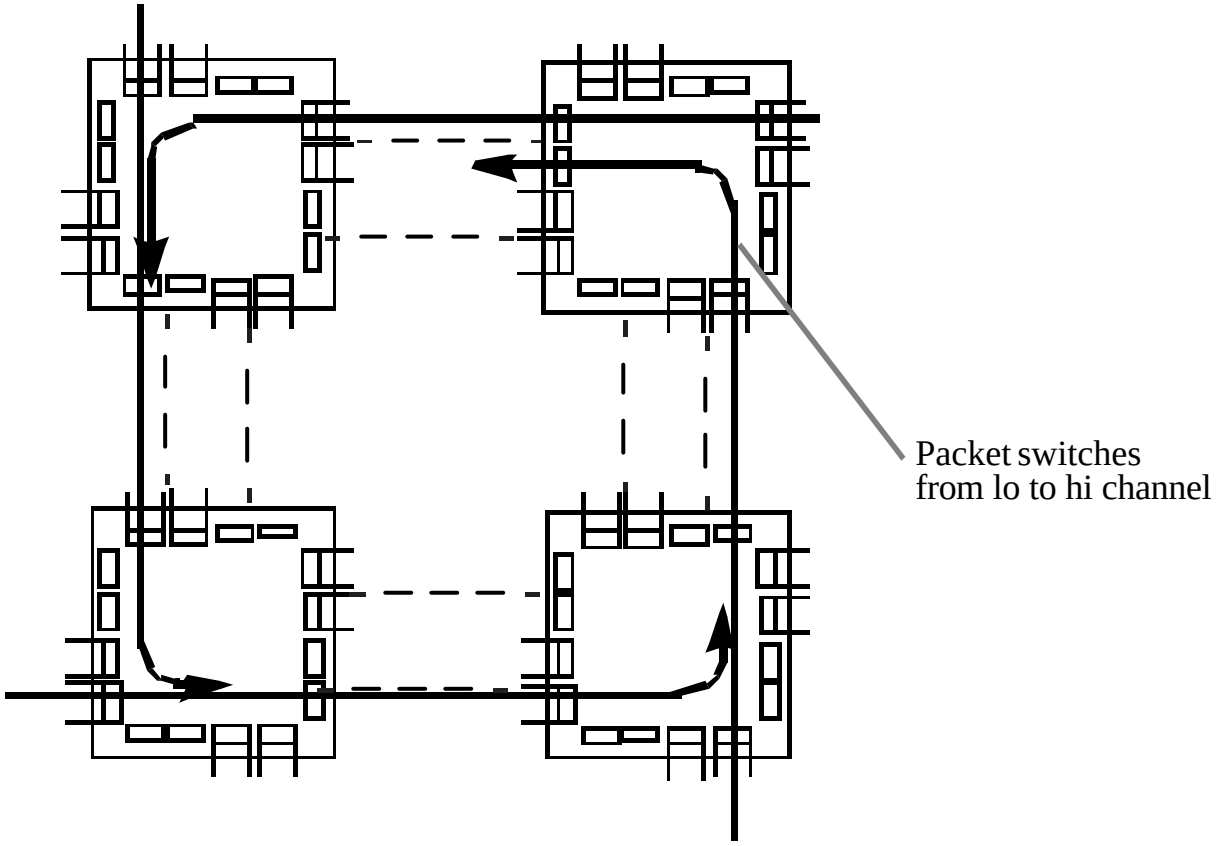
Deadlock free wormhole networks?

- Basic dimension order routing techniques don't work for k-ary d-cubes
  - only for k-ary d-arrays (bi-directional)
- Idea: add channels!
  - provide multiple "virtual channels" to break the dependence cycle
  - good for BW too!



  - Do not need to add links, or xbar, only buffer resources
- This adds nodes the the CDG, remove edges?

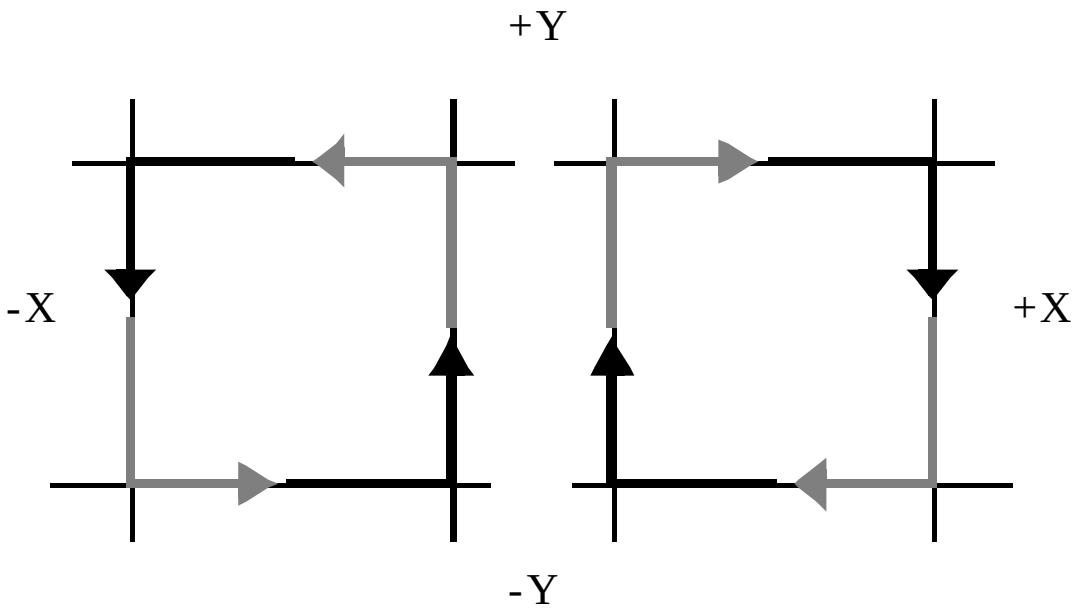
Breaking deadlock with virtual channels



Up*-Down* routing

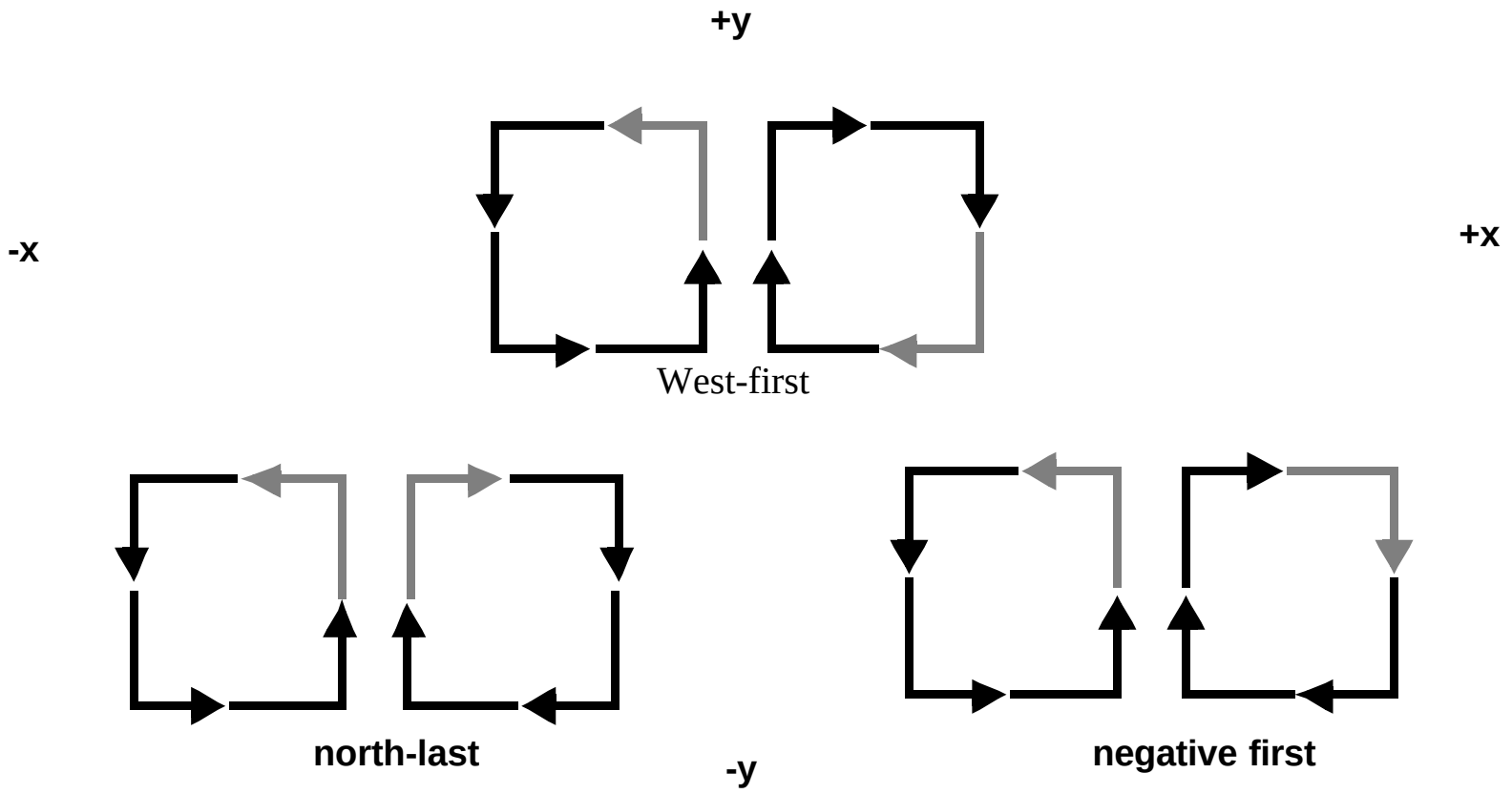
- Given any bidirectional network
- Construct a spanning tree
- Number of the nodes increasing from leaves to roots
- UP increase node numbers
- Any Source -> Dest by UP*-DOWN* route
  - up edges, single turn, down edges

Turn Restrictions in ΔX, ΔY

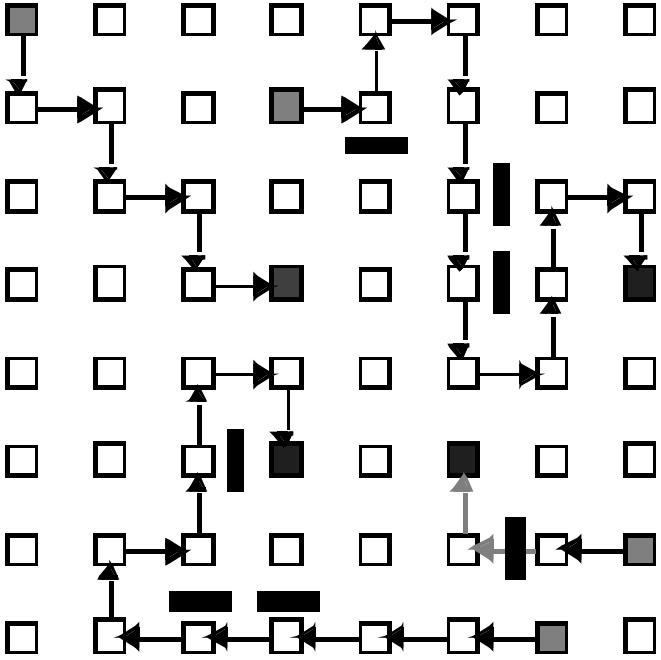


- XY routing forbids 4 of 8 turns and leaves no room for adaptive routing
- Can you allow more turns and still be deadlock free

Minimal turn restrictions in 2D



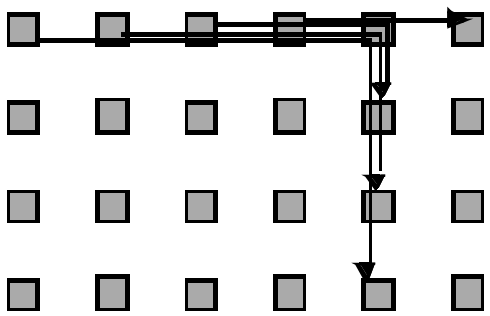
Example legal west-first routes

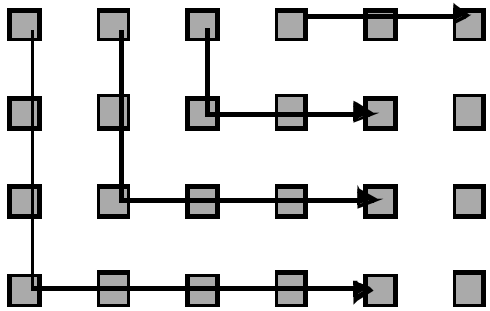


- Can route around failures or congestion
- Can combine turn restrictions with virtual channels

Adaptive Routing

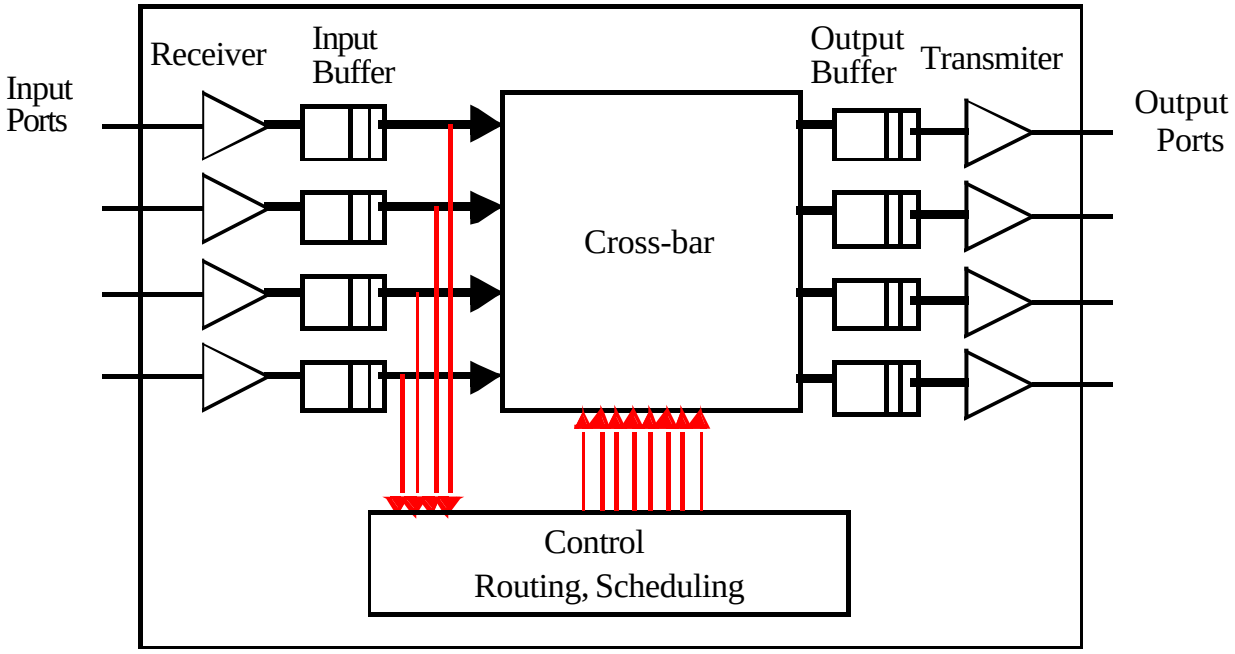
- R: C x N x Σ -> C
- Essential for fault tolerance
  - at least multipath
- Can improve utilization of the network
- Simple deterministic algorithms easily run into bad permutations



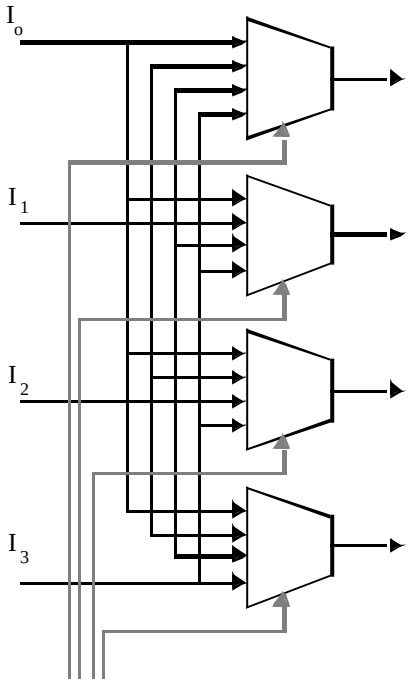


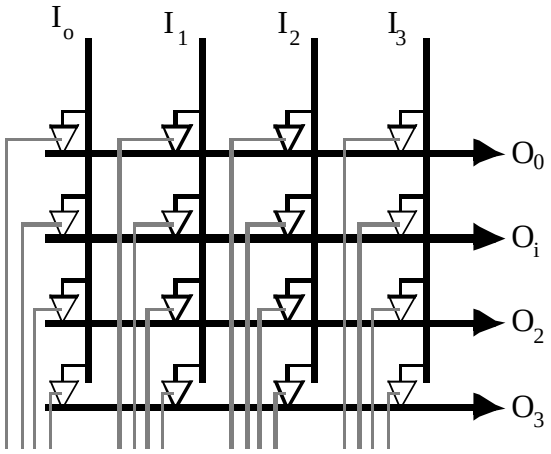
- fully/partially adaptive, minimal/non-minimal
- can introduce complexity or anomolies
- little adaptation goes a long way!

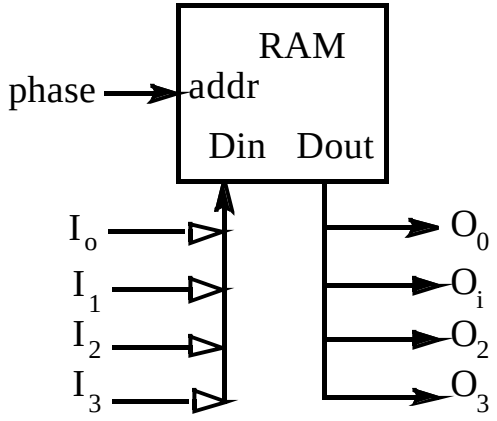
Switch Design



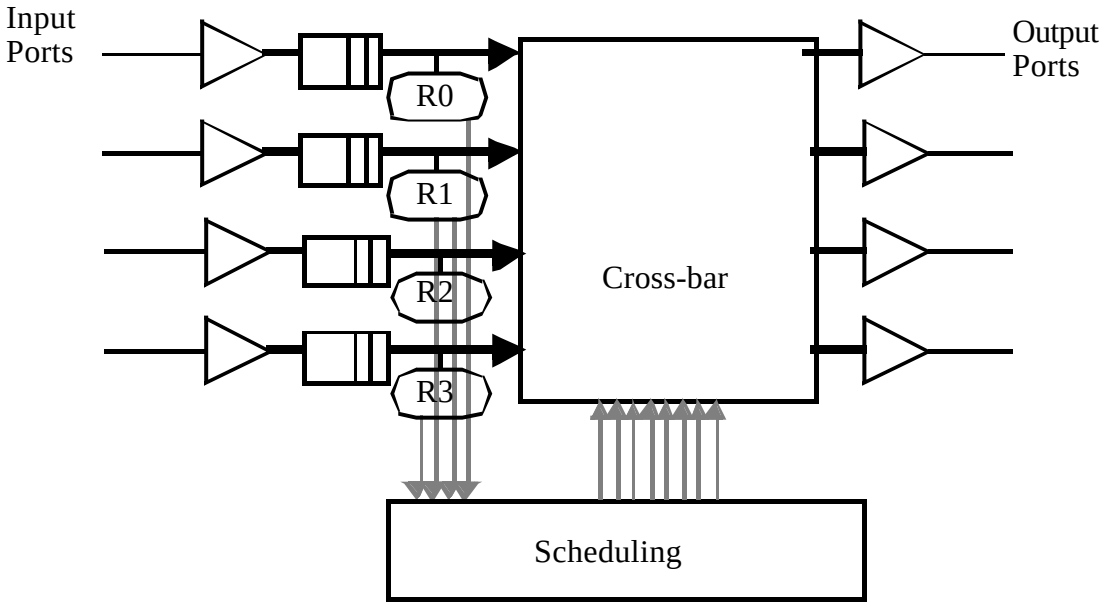
How do you build a crossbar





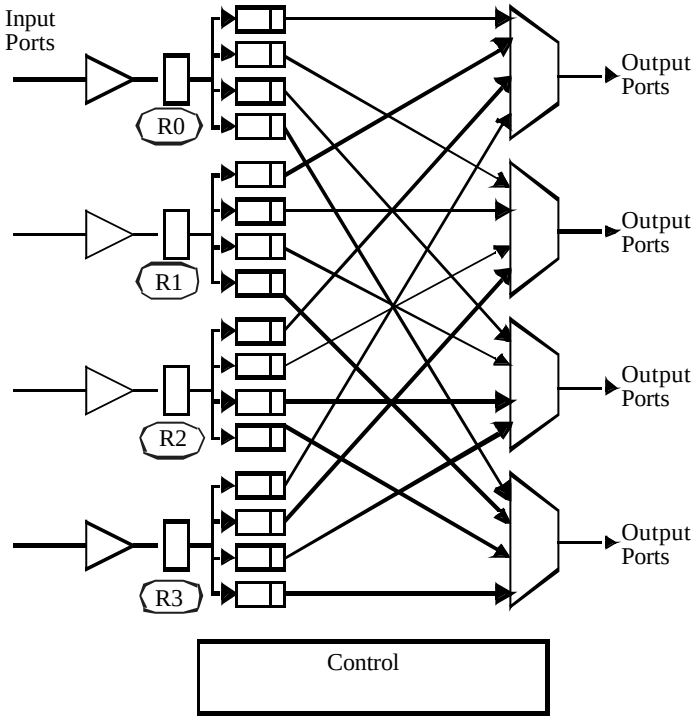


Input buffered switch



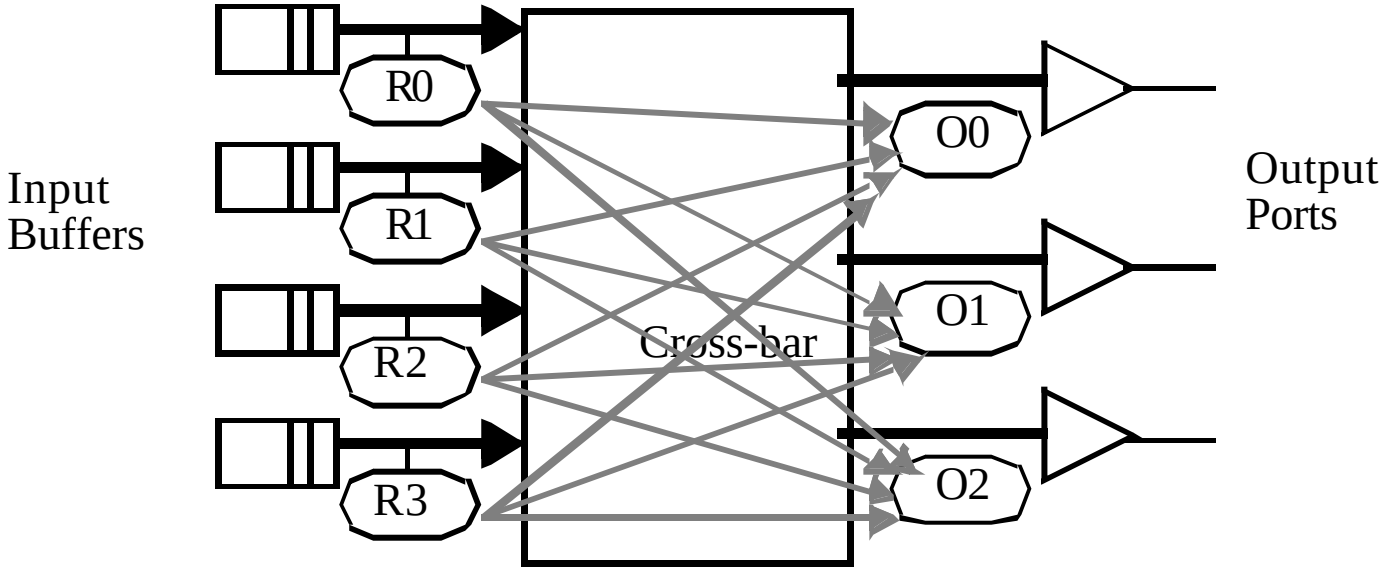
- Independent routing logic per input
  - FSM
- Scheduler logic arbitrates each output
  - priority, FIFO, random
- Head-of-line blocking problem -> output buffering

Output Buffered Switch



- Added cost of multiplexers/wires -> shared pool?

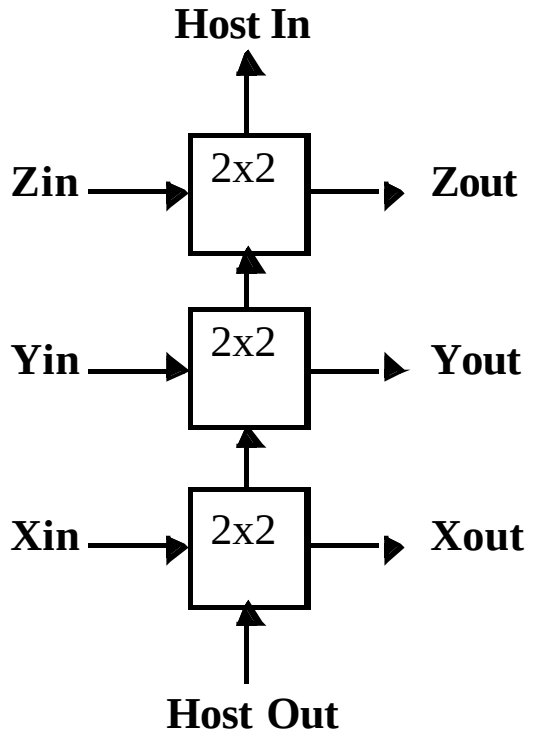
Output scheduling



- n independent arbitration problems?
  - static priority, random, round-robin
- simplifications due to routing algorithm?
- general case is max bipartite matching

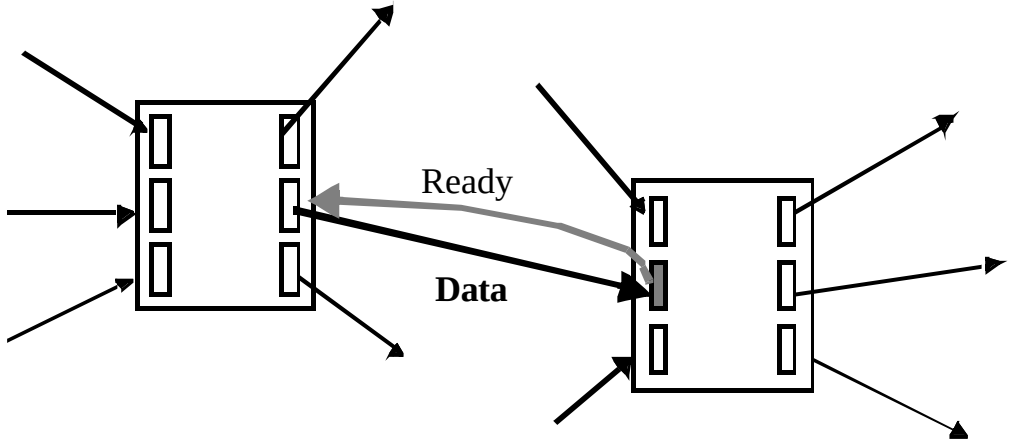
Stacked Dimension Switches

- Dimension order on 3D cube



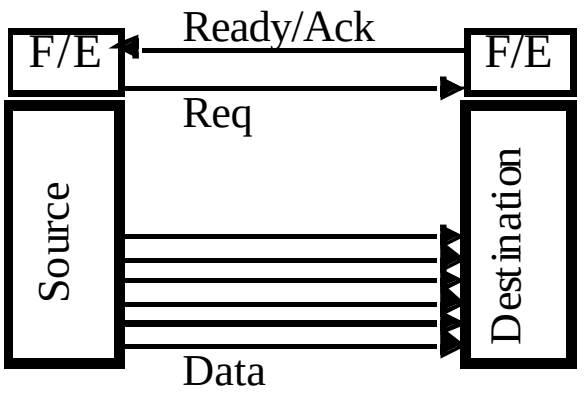
Flow Control

- Comparison with LAN/WAN
  - Must be delivered more reliably, large concurrent flow, small timescale
  - 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

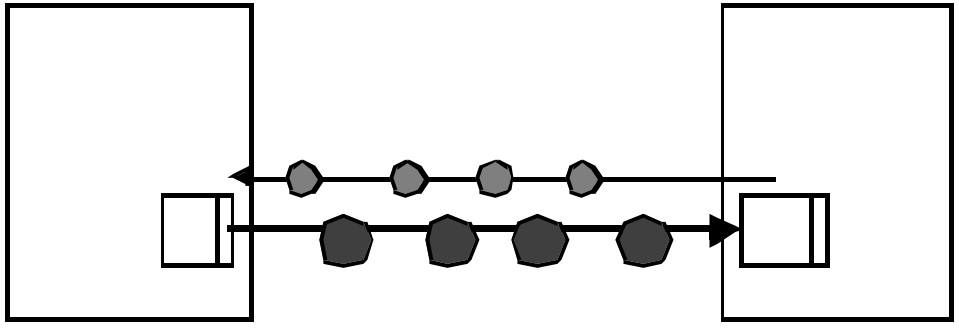


Examples

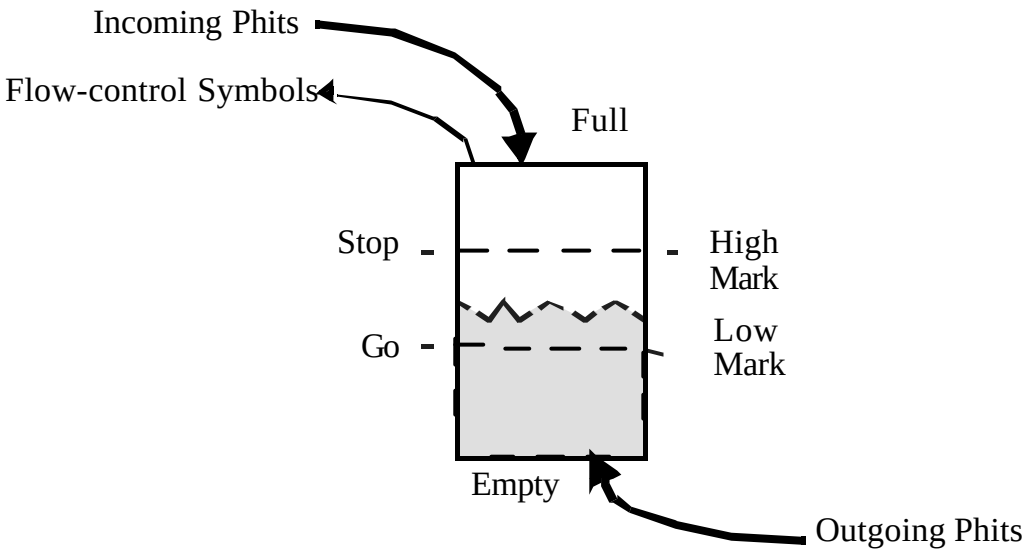
- Short Links



- Long links
  - several flits on the wire



Smoothing the flow



- How much slack do you need to maximize bandwidth?

End-to-End flow control

- Hot Spots
- Global communication operations
- Natural parallel program dependences

Routing/Switching/Flow Control 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