Chapter 10
Scalable Interconnection Networks

10.1 Scalable, High Performance Network

At Core of Parallel Computer Architecture

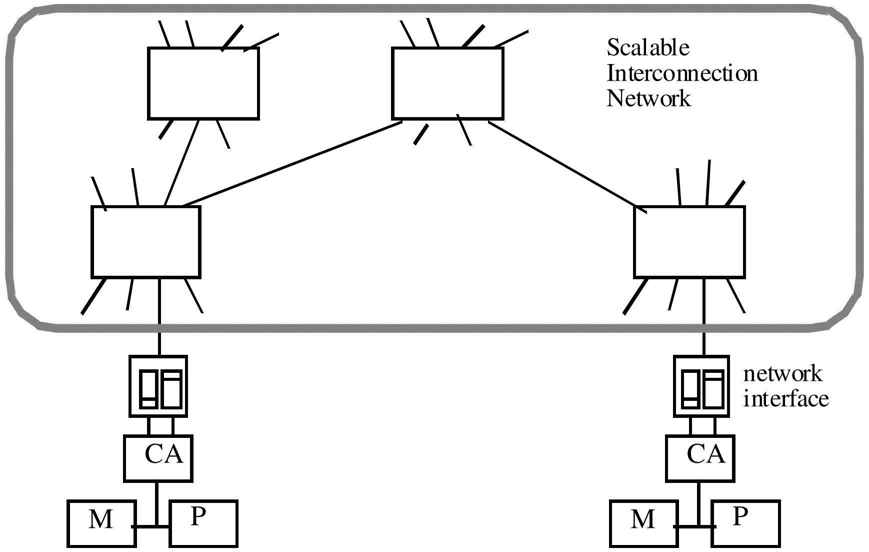
Requirements and trade-offs at many levels

- 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



Adaptado dos slides da editora por Mario Côrtes – IC/Unicamp – 2009s2

Requirements from Above

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
- degree of overlap (slackness)

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

Goals

Latency as small as possible

As many concurrent transfers as possible

- operation bandwidth
- data bandwidth

Cost as low as possible

Outline

Introduction

10.1-2 Basic concepts, definitions, performance perspective

10.3 Organizational structure

10.4 Topologies

10.6 Routing and switch design

Basic Definitions

Network interface

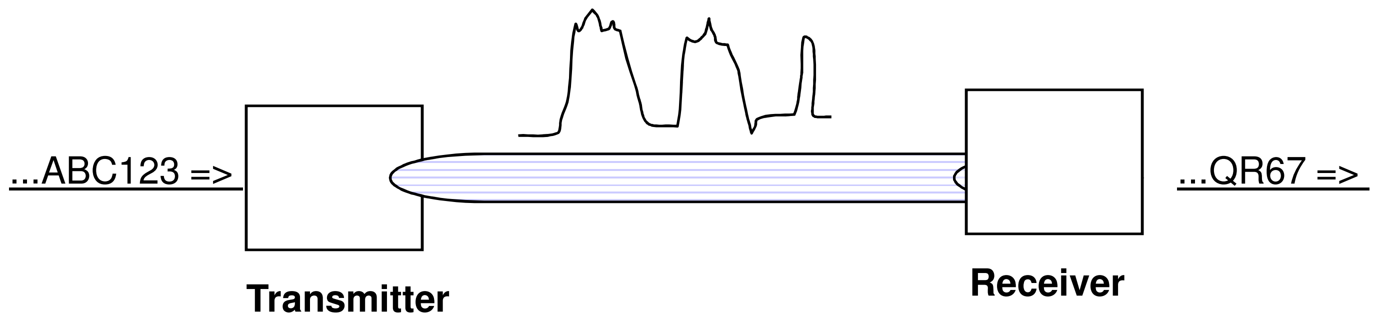
Links

- 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

Adaptado dos slides da editora por Mario Côrtes – IC/Unicamp – 2009s2

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*

Think streets and intersections

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 routes)

- 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?
- (ver definição de flit e exemplo p. 753)

What determines performance

Interplay of all of these aspects of the design

Topological Properties

Routing Distance - number of links on route

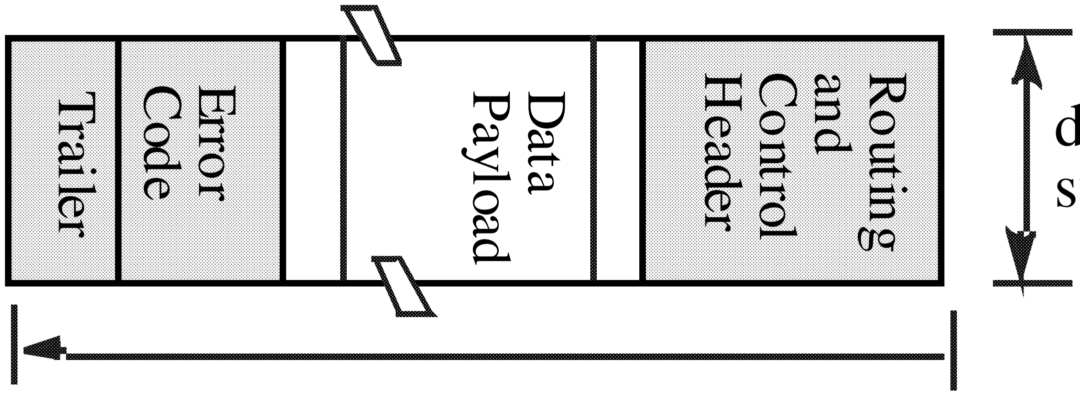
Shortest Path – menor routing distance entre dois nós

Diameter - maximum shortest path entre quaisquer dois nós

Average Distance – média das routing distances de todos pares de nós

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

(Packet switching é mais usado que circuit switching)

Componentes: Header (roteamento e controle), Trailer (ECC), Payload

Two basic mechanisms for abstraction

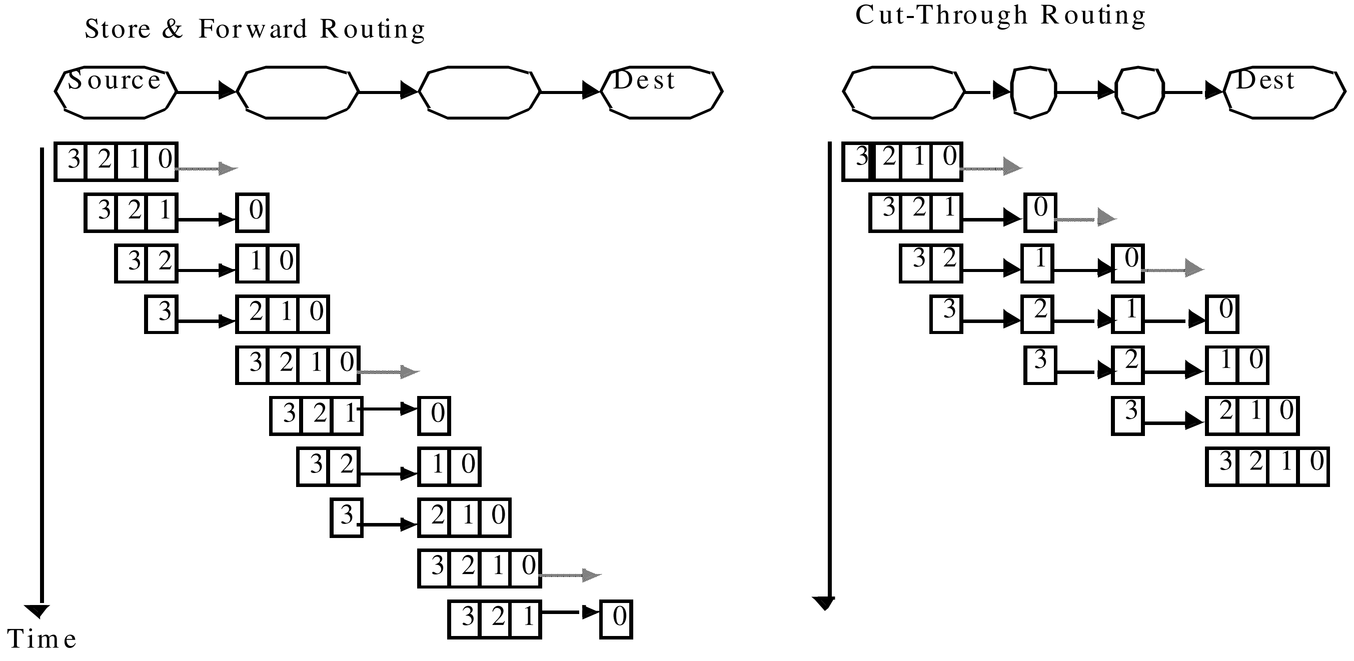
- Encapsulation: carrega informação de protocolo de alto nível dentro do pacote
- Fragmentation: fragmenta as informações de protocolo de alto nível em uma sequência de mensagens

(p. 754)

10.2 Communication Perf: Latency

- $\text{Time}(n)_{s\text{-}d}$ = overhead + routing delay + channel occupancy + contention delay
- Occupancy = $(n + n_e) / b$
  - n= tamanho payload; n_e = tamanho do envelope
  - b= bandwidth
  - visto nos capítulos passados; mas agora, visão do ponto de vista da rede
- Routing delay?
  - depende do Nº de links (routing distance) e do delay de cada switch (Δ)
- Contention?

Store&Forward vs Cut-Through Routing



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

what if message is fragmented? (ver texto; semelh. ct)

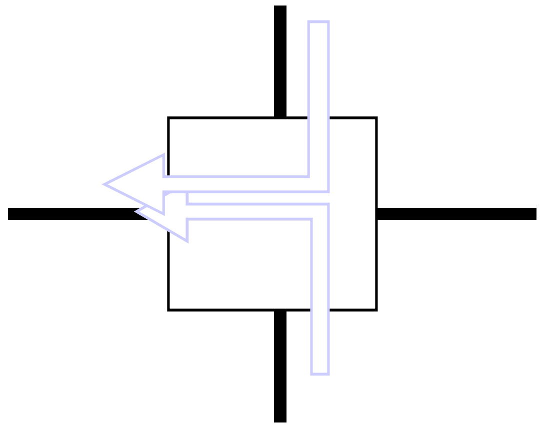
ver texto: packet switching e circuit switching

wormhole vs virtual cut-through

(p. 756-757)

h = routing dist: Δ =delay/hop

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
- (back pressure)
- tree saturation

Closed system - offered load depends on delivered

10.2.2 Bandwidth

What affects local bandwidth?

- packet density b . n/($\mathbf{n + n_e}$)
- routing delay (derated) b . n / ($\mathbf{n + n_e + w\Delta}$)
- contention
  - endpoints
  - within the network

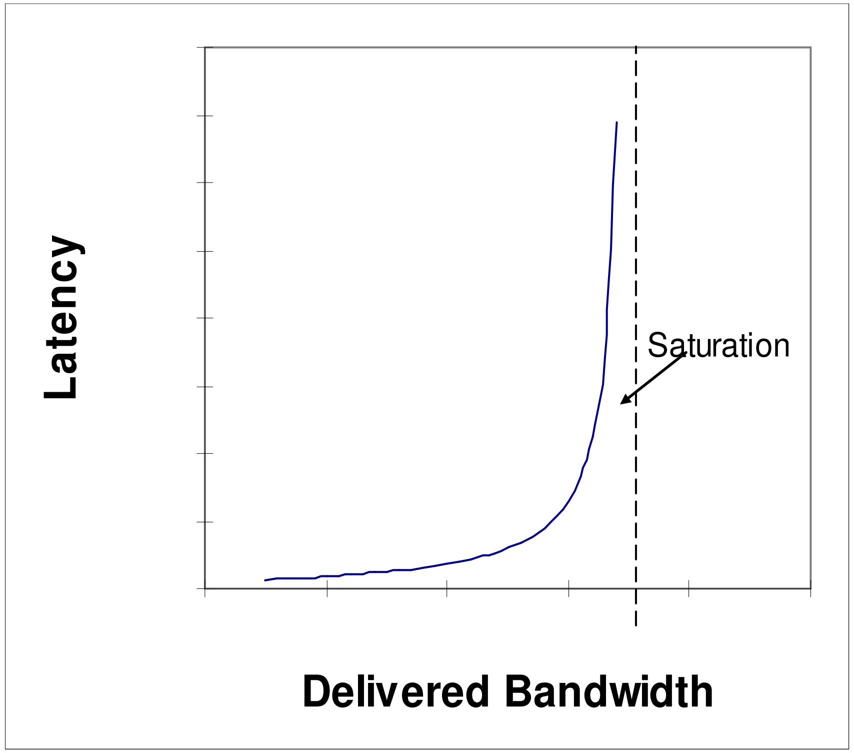
wΔ = oportunidade perdida devido a link bloqueado

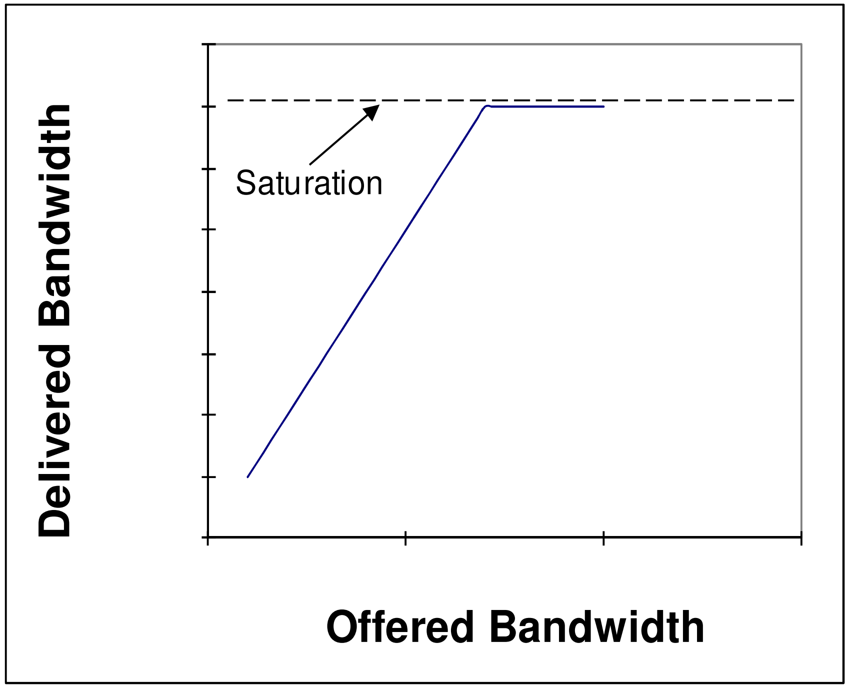
Aggregate bandwidth

- bisection bandwidth
  - sum of bandwidth of smallest set of links that partition the network (duas metades iguais)
- total bandwidth of all the channels: Cb (Nº canais * b / canal)
- suppose N hosts issue packet every M cycles with ave dist h
  - each msg occupies h channels for l = n/w cycles each
  - C/N channels available per node
  - link utilization ρ = MC/N h l < 1 (na realidade << 1)

(p. 761-762)

Saturation





- Delivered bandwidth: valor real “entregue” pela rede
  - saturação: latência cresce exponencialmente
- Offered bandwidth: demanda colocada pelas aplicações
  - ok p/ baixas demandas; atinge a saturação

(p. 762-763)

Outline

10.3 Organizational Structure

Processors

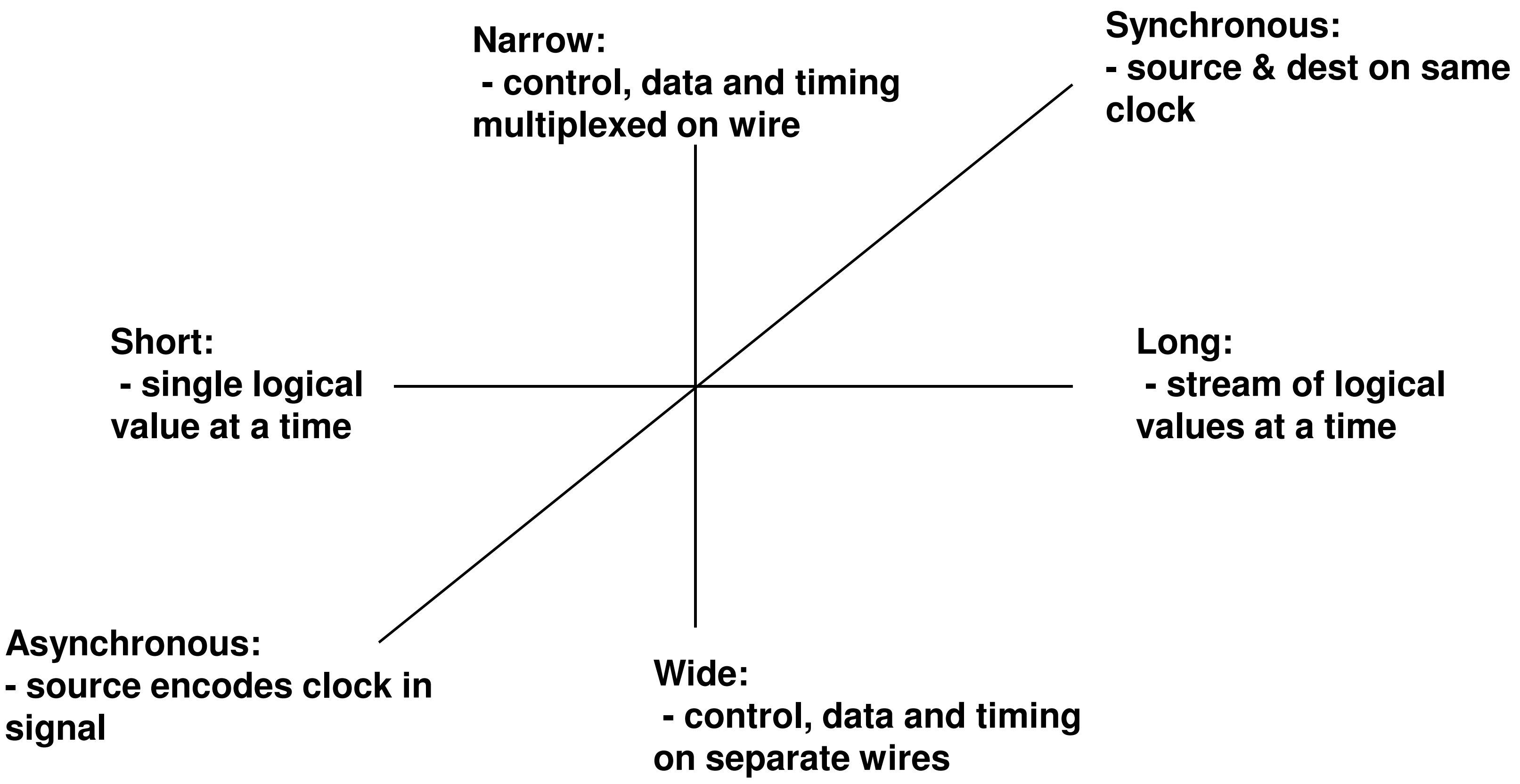
- datapath + control logic + memory interface
- datapath: ALU, register file, pipeline latches …..
- control logic determined by examining register transfers in the datapath
- conexões: via de dados (curtas e rápidas); controle (longas e lentas); escalamento ??

Networks (composta dos componentes)

- links
- switches
- network interfaces

10.3.1 Link Design/Engineering Space

- Cable of one or more wires/fibers with connectors at the ends attached to switches or interfaces
  - características importantes: length, width, clocking



Narrow:
- control, data and timing multiplexed on wire
Synchronous:
- source & dest on same clock
Short:
- single logical value at a time
Long:
- stream of logical values at a time
Asynchronous:
- source encodes clock in signal
Wide:
- control, data and timing on separate wires

(p. 764)

Example: Cray MPPs

T3D: Short, Wide, Synchronous (300 MB/s)

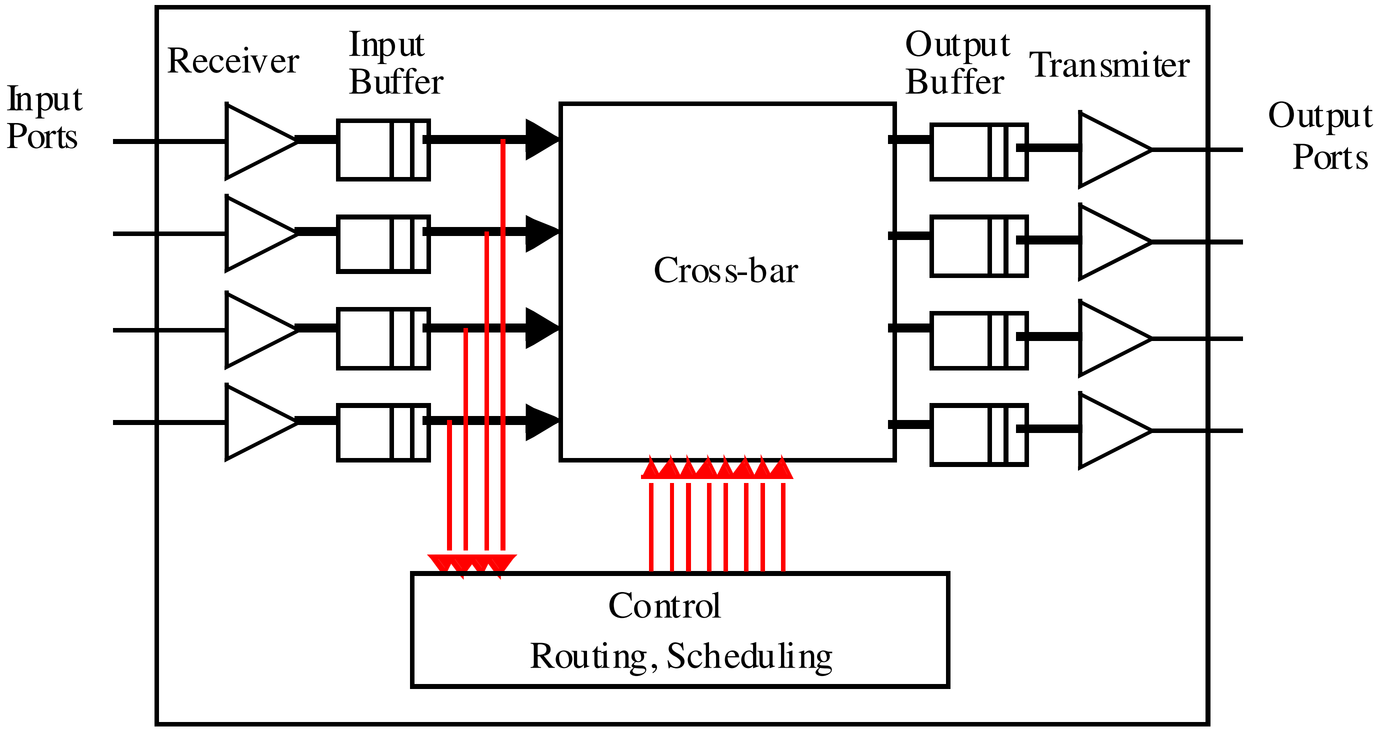
- 24 bits: 16 data, 4 control, 4 reverse direction flow control
- single 150 MHz clock (including processor)
- flit = phit = 16 bits
- two control bits identify flit type (idle and framing)
  - no-info, routing tag, packet, end-of-packet

T3E: long, wide, asynchronous (500 MB/s)

- 14 bits, 375 MHz, LVDS (low voltage differential signal)
- 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) ?

10.3.2 Switches



- usualmente, nº input ports = nº output ports = degree (há exceções)

Switch Components

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

Outline

10.4 Interconnection Topologies

Class networks scaling with N

Logical Properties:

- distance, degree

Physcial properties

- length, width

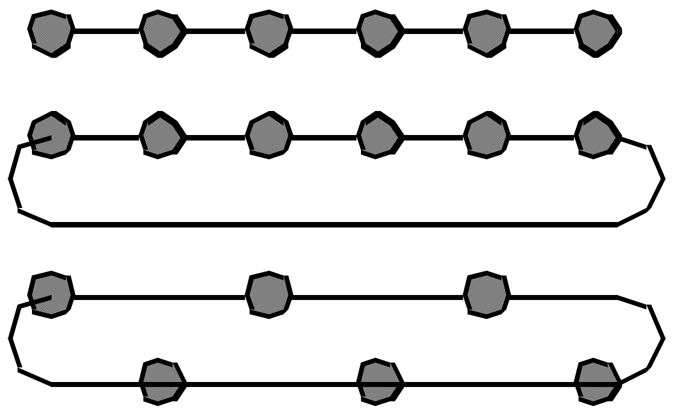
10.4.1 Fully connected network (um switch, grau N, todos x todos)

- diameter = 1
- degree = N
- cost?
  - bus => O(N), but BW is O(1) - actually worse
  - crossbar => O(N^2) for BW O(N)
- fully connected não são escaláveis na prática

VLSI technology determines switch degree (alguns nós são sub-redes fully connected)

(p. 768)

10.4.2 Linear Arrays and Rings



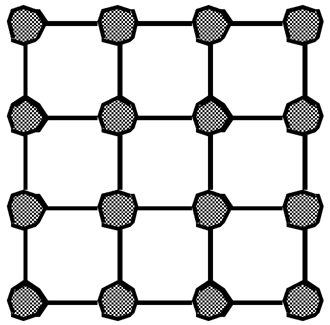
Linear Array

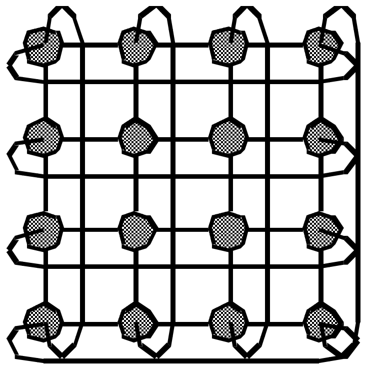
- Diameter?
- Average Distance?
- Bisection bandwidth?
- Route A -> B given by relative address R = B-A

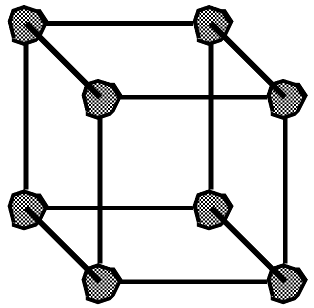
Torus?

Examples: FDDI, SCI, FiberChannel Arbitrated Loop, KSR1

10.4.3 Multidimensional Meshes and Tori


2D Grid




3D Cube

d-dimensional array

- $n = k_{d-1} * ... * k_O$ 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)?

(p. 769)

Properties

Routing

- relative distance: $R = (b_{d-1} - a_{d-1}, \dots, b_0 - a_0)$
- traverse $ri = b_i - a_i$ 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

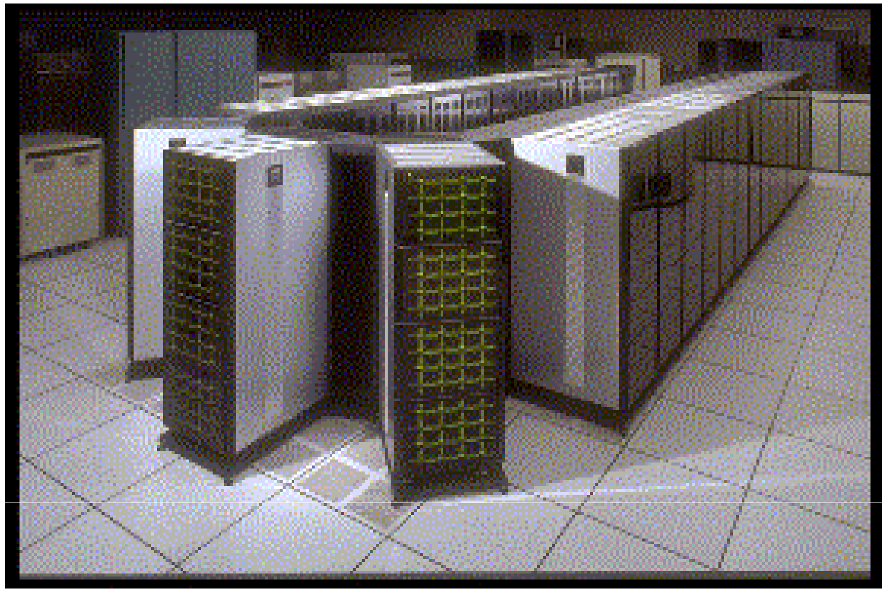
Partitioning?

Physical layout?

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

Short wires

Real World 2D mesh



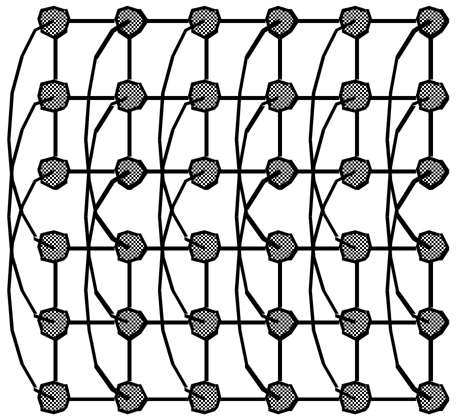
(ver expl 10.1)

1824 node Paragon: 16 x 114 array

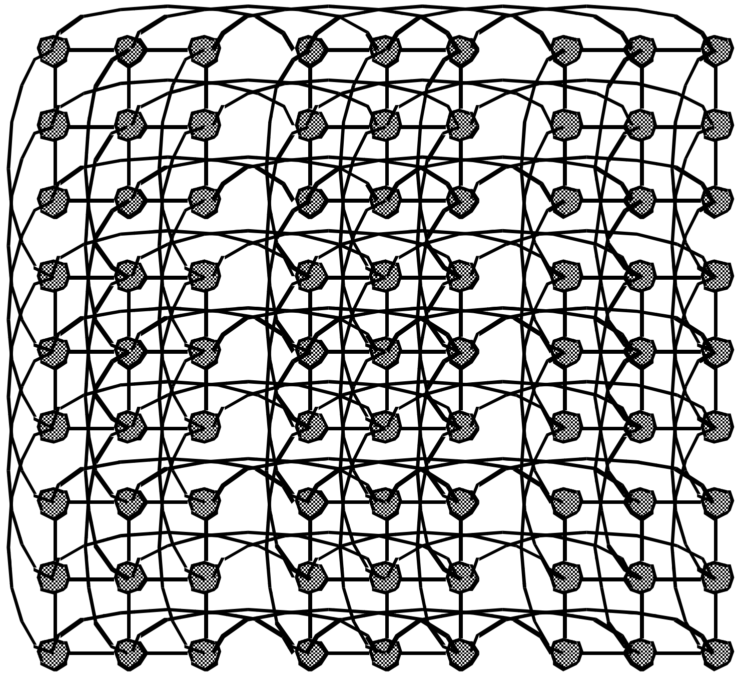
Cada gabinete: 4 (largura) x 16 (altura) = 64 nós

Comunicação entre bastidores (= bissection bandwidth) = 16

Embeddings in two dimensions



6 x 3 x 2



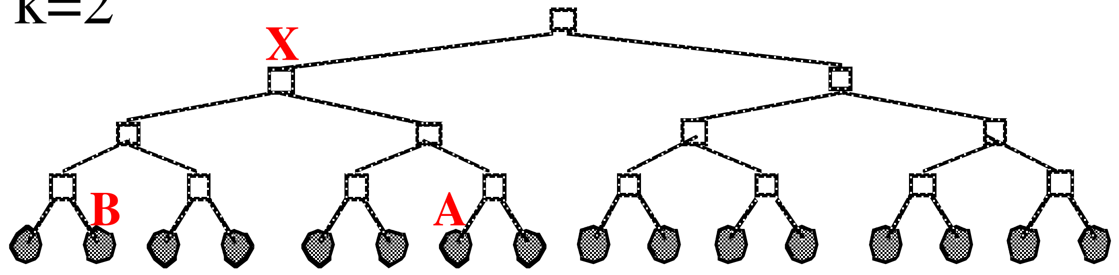
(4 dimension) d=4 k=3;
disposto em 2D

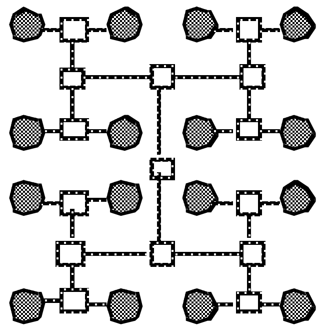
Embed multiple logical dimension in one physical dimension using long wires

(p. 772)

10.4.4 Trees

d=4 k=2





Diameter and avg. distance are logarithmic

- k-ary tree, height d = $\log_k N$
- address specified d-vector of radix k coordinates describing path down from root (0111 ?)

Fixed degree (neste caso =3)

Ver direct x indirect

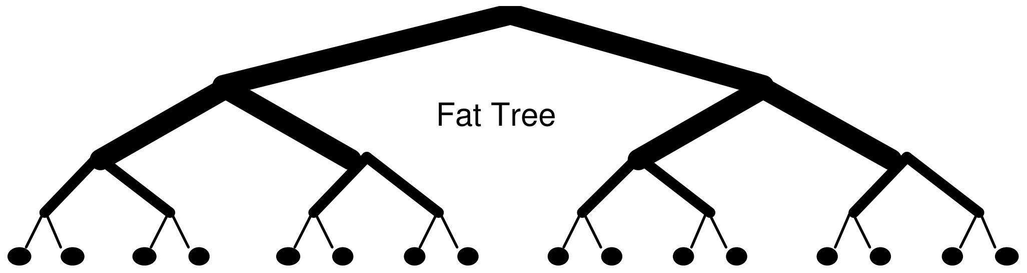
Route up to common ancestor and down (não precisa ir até raiz)

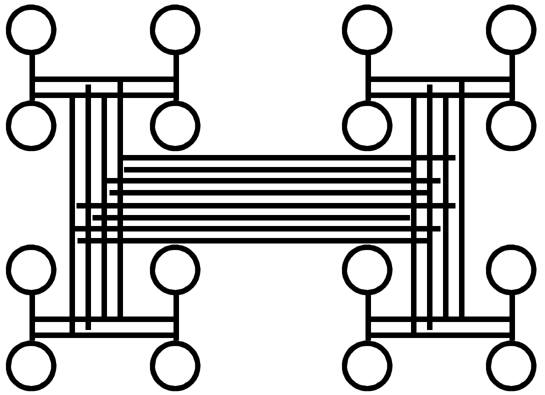
- (A → B) R = B xor A (ex: 0001 xor 0110 =0111)
- let i be position of most significant 1 in R, route up i+1 levels (sobe 2+1)
- down in direction given by low i+1 bits of B (001)

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

Bisection BW?   (analogia sistema circulatório → fat trees)

Fat-Trees



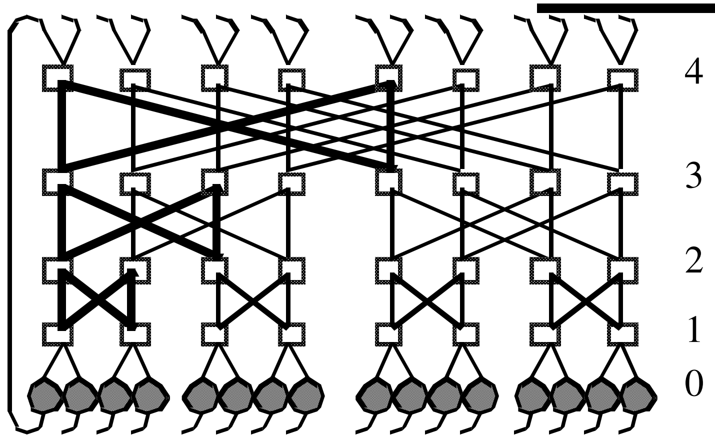


Fatter links (really more of them) as you go up, so bisection BW scales with N

Adaptado dos slides da editora por Mario Côrtes – IC/Unicamp – 2009s2

(p. 774)

10.4.5 Butterflies



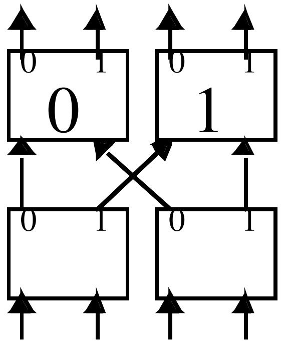
A B C

16 node butterfly

Exmpl:

B→A: A=0010 B=0110

C→B: C=1011



building block

Tree with lots of roots! $N = 2^d$ nós de host; e $d.2^{d-1}$ nós de switch

Indirect: hosts deliver pckts into lvl 0 and receive pckts from lvl d

log N níveis (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 $N^{(d-1)/d}$ (mesh)

(p. 774-775)

k-ary d-cubes vs d-ary k-flies

Tree ou Mesh		Fly
Degree d		
Custo		
N switches	vs	N log N switches
BW		
Diminishing BW per node	vs	constant
Requires locality	vs	little benefit to locality

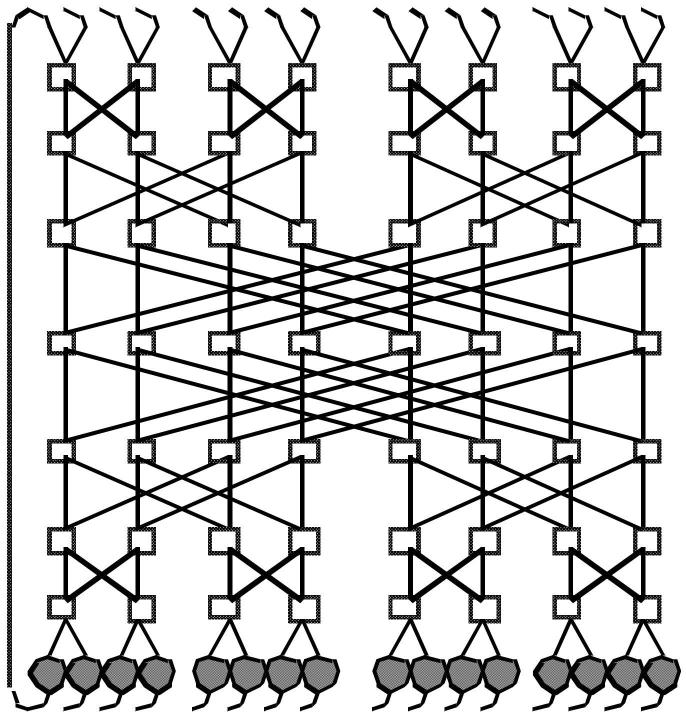
Can you route all permutations?

- conflito em duas mensagens simultâneas que usem algum link comum
- confiabilidade: há apenas uma rota entre 2 links

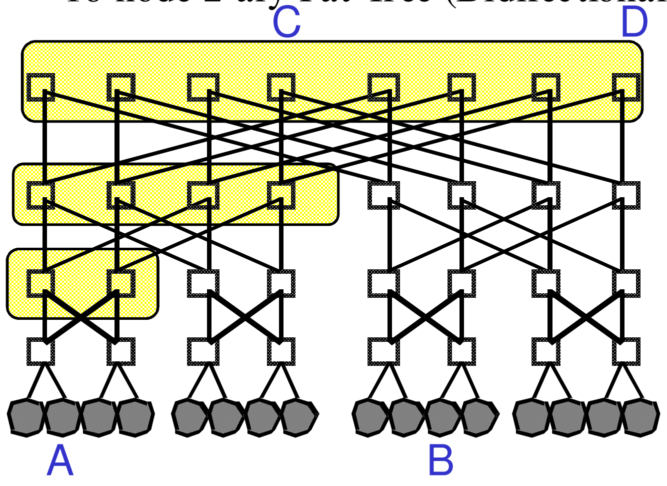
(p. 776)

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?

(expl: de A→B, escolha C ou D como nós intermediários

Forma elegante mas pouco uso (difícil calcular nó intermediário)

(p. 776-777)

Adaptado dos slides da editora por Mario Côrtes – IC/Unicamp – 2009s2

10.4.6 Hypercubes

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

O(logN) hops

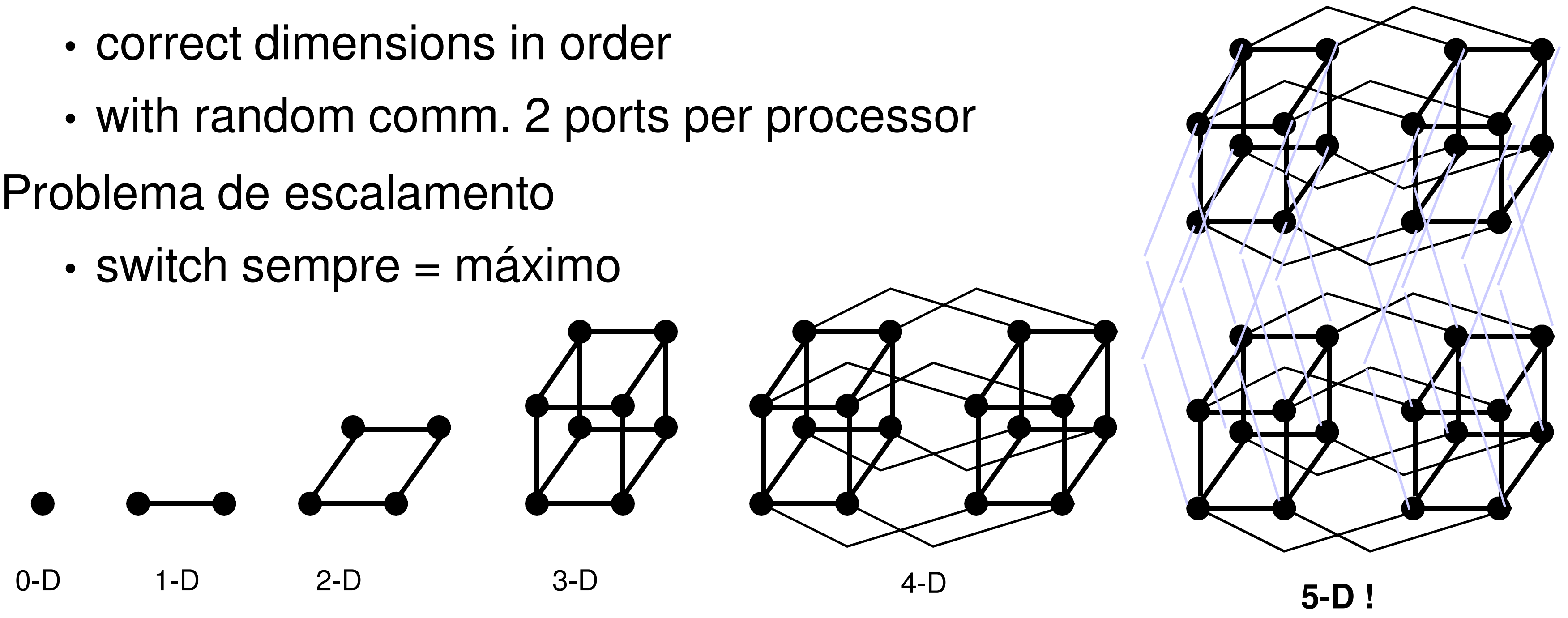
Good bisection BW

Complexity

- out degree is n = logN
- correct dimensions in order
- with random comm. 2 ports per processor

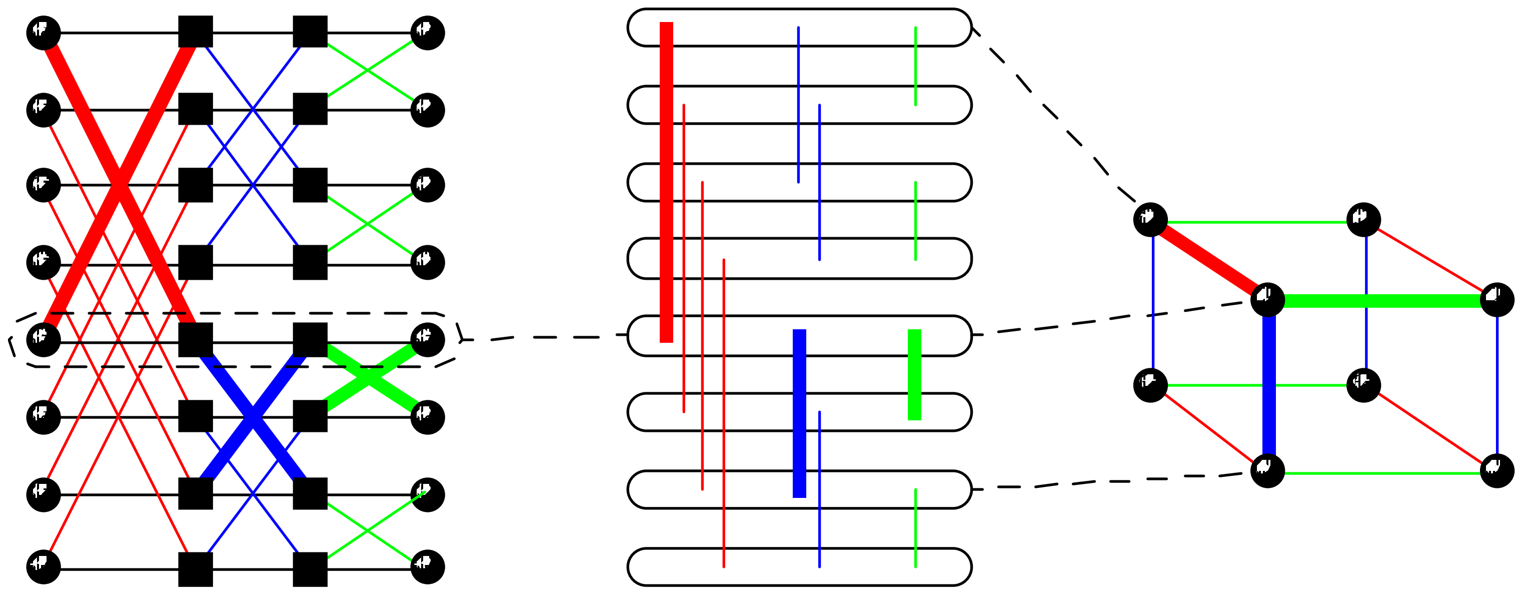
Problema de escalamento

- switch sempre = máximo



(p. 778)

Relationship of Butterflies to Hypercubes



Wiring is isomorphic

Except that Butterfly always takes *log n* steps

Adaptado dos slides da editora por Mario Côrtes – IC/Unicamp – 2009s2

Properties of Some Topologies

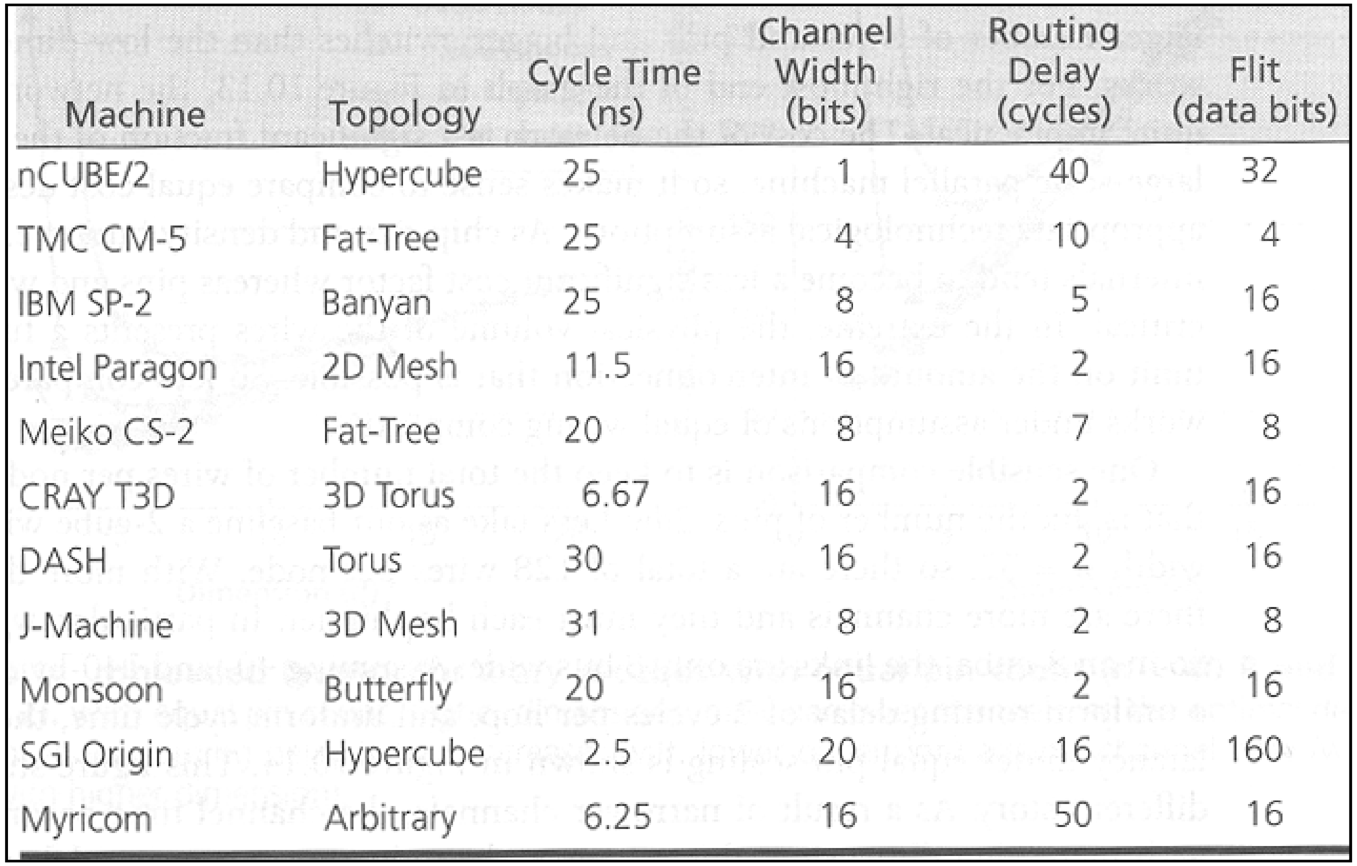
Topology	Degree	Diameter	Ave Dist	Bisection	D (D ave) @ P=1024	
1D Array	2	N-1	N / 3	1	huge	
1D Ring	2	N/2	N/4	2		
2D Mesh	4	2 ($N^{1/2}$ - 1)	2/3 $N^{1/2}$	$N^{1/2}$	63 (21)	
2D Torus	4	$N^{1/2}$	1/2 $N^{1/2}$	2$N^{1/2}$	32 (16)	
k-ary n-cube	2n	nk/2	nk/4	nk/4	15 (7.5)	@n=3
Hypercube	n =log N		n	n/2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

Adaptado dos slides da editora por Mario Côrtes – IC/Unicamp – 2009s2

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

Wide links, smaller routing delay

Tremendous variation

(p. 781)

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

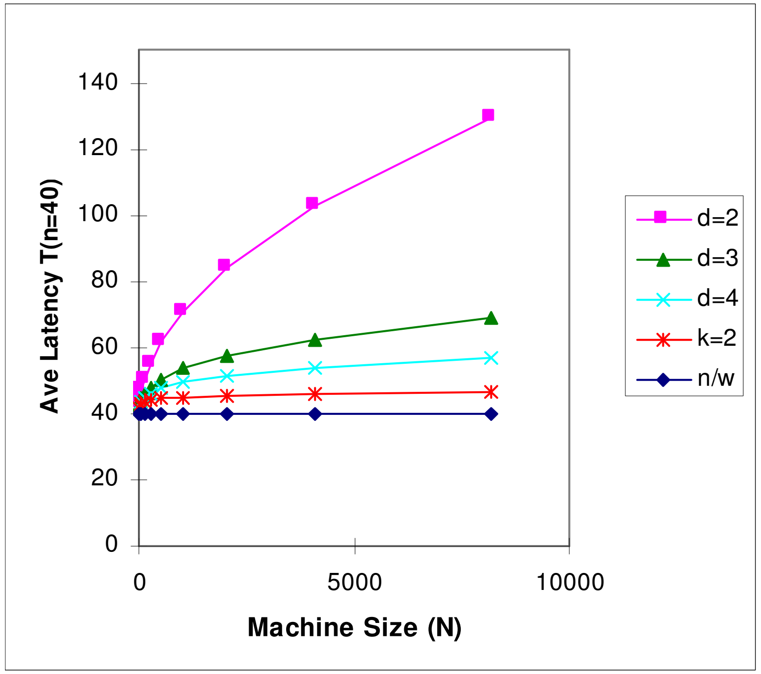
- Harder to build, more wires, longer average length
- Fewer hops, better bisection bandwidth
- Can handle non-local traffic

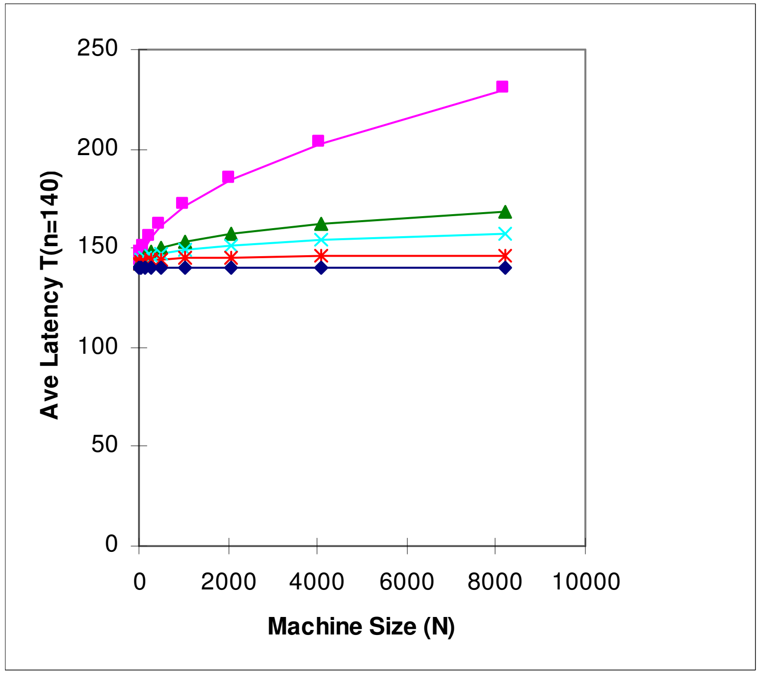
k-ary d-cubes provide a consistent framework for comparison

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

(p. ??? 780?)

Traditional Scaling: Latency(P)





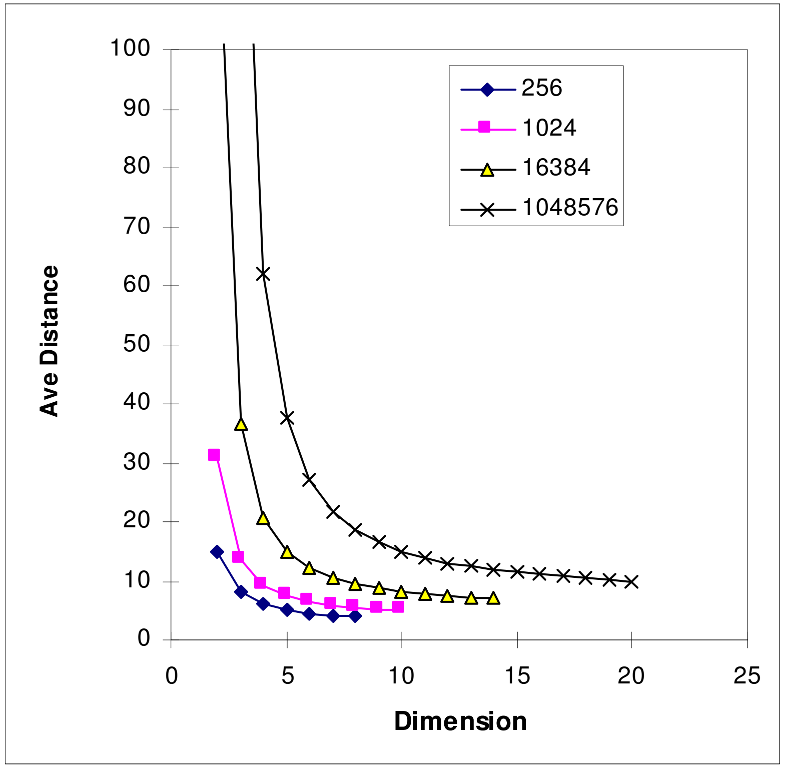
Assumes equal channel width

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

(p. 780)

Adaptado dos slides da editora por Mario Côrtes – IC/Unicamp – 2009s2

Average Distance



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

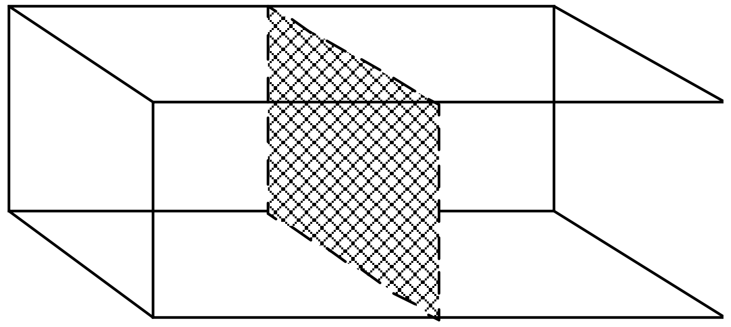
but, equal channel width is not equal cost!

Higher dimension => more channels

(p. ????)

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
- What about many long fat 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)

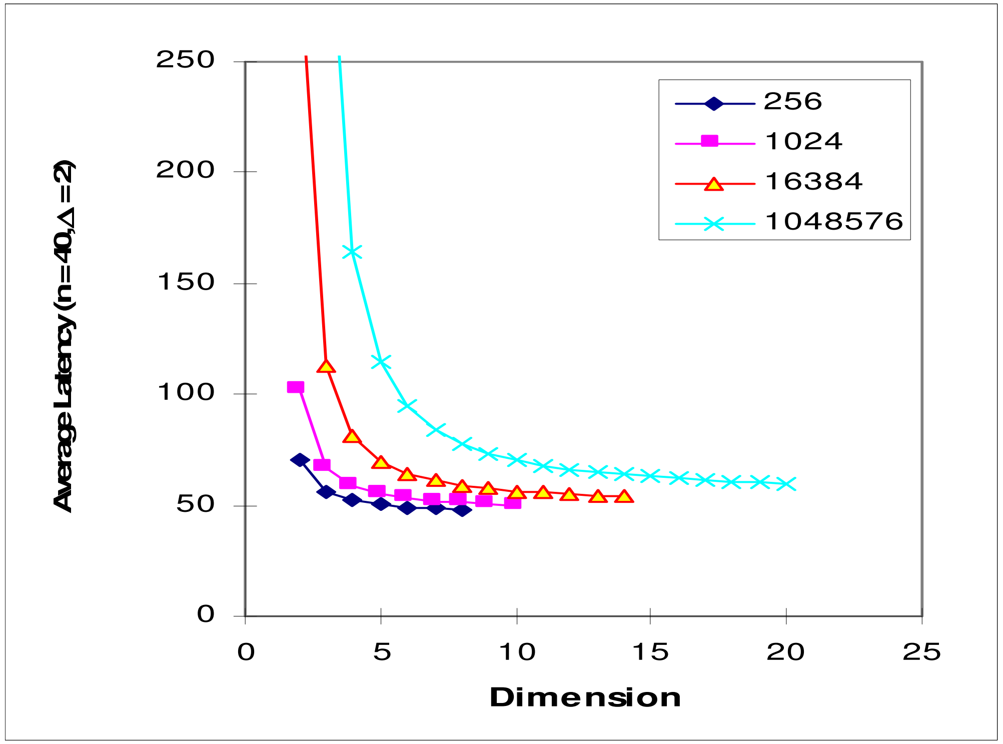
total links = Nd

pins per node = 2wd

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

2Nw/k wires cross the middle

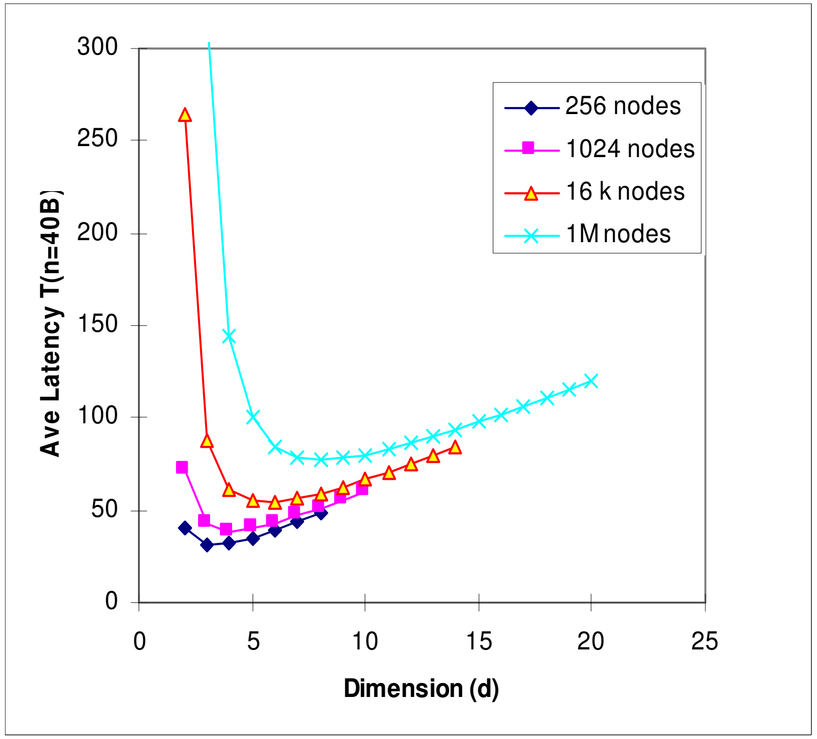
Latency(d) for P with Equal Width

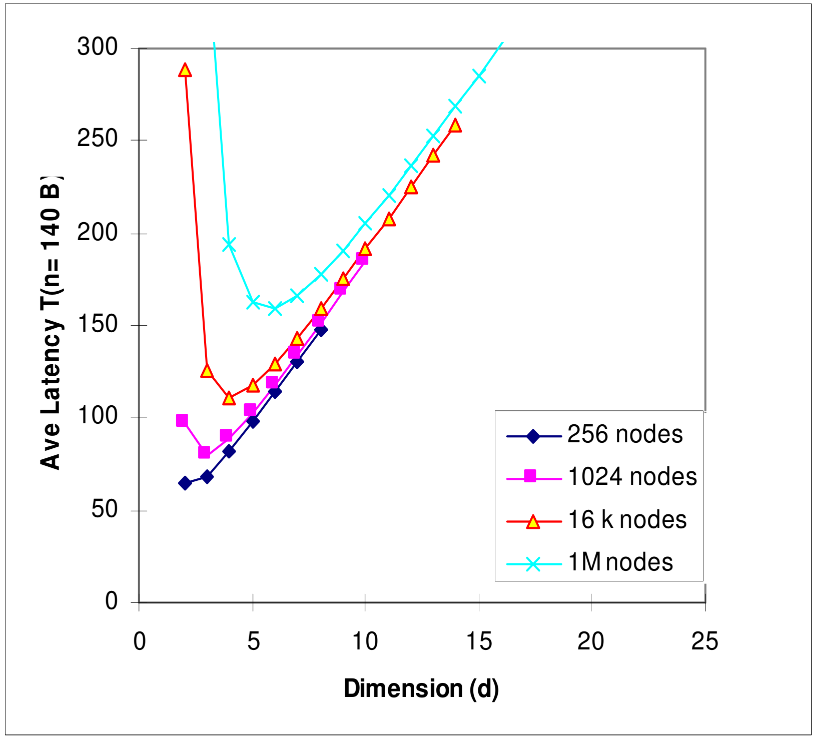


total links(N) = Nd

(p. ?????)

Latency with Equal Pin Count





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

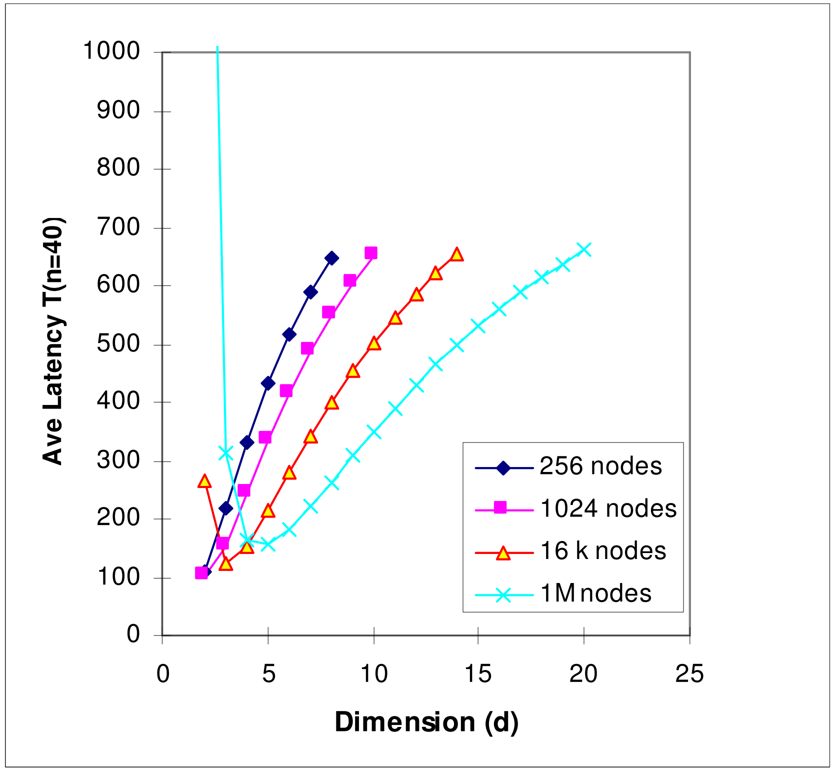
fix 2dw pins => w(d) = 64/d

distance up with d, but channel time down

(p. 782-783)

Adaptado dos slides da editora por Mario Côrtes – IC/Unicamp – 2009s2

Latency with Equal Bisection Width



N-node hypercube has N bisection links

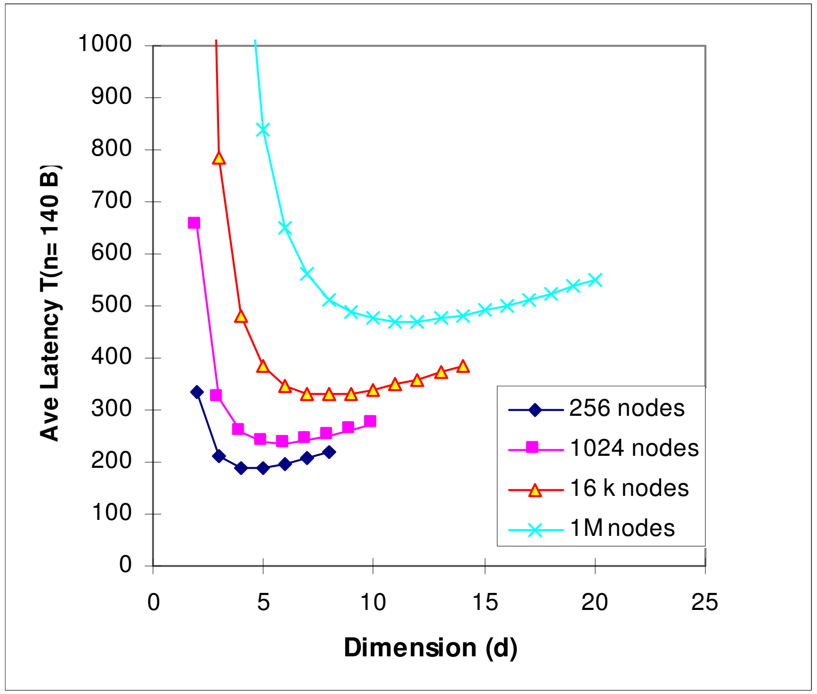
2d torus has 2N $^{1/2}$

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

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

(p. 782-784 Fig 10.15)

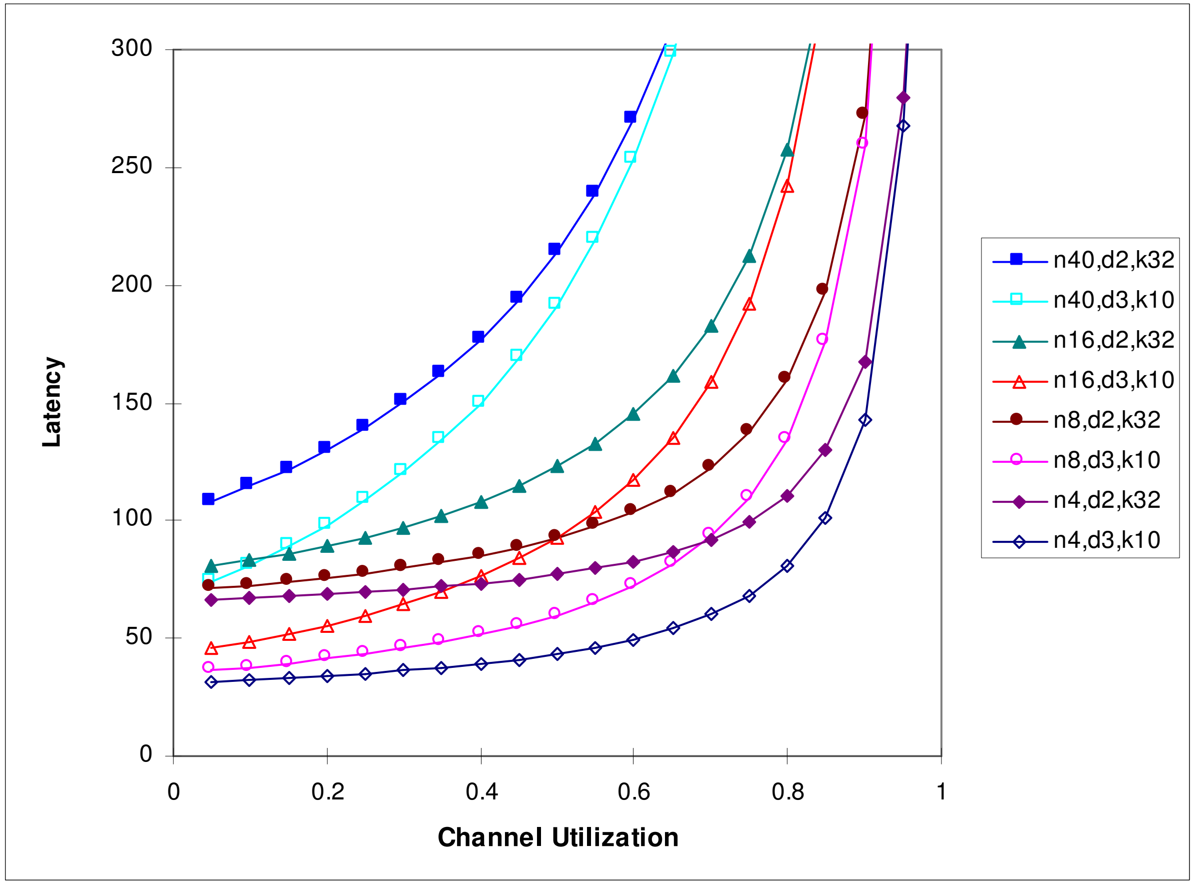
Larger Routing Delay (w/ equal pin)



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

(p. 784 Fig. 10.16)

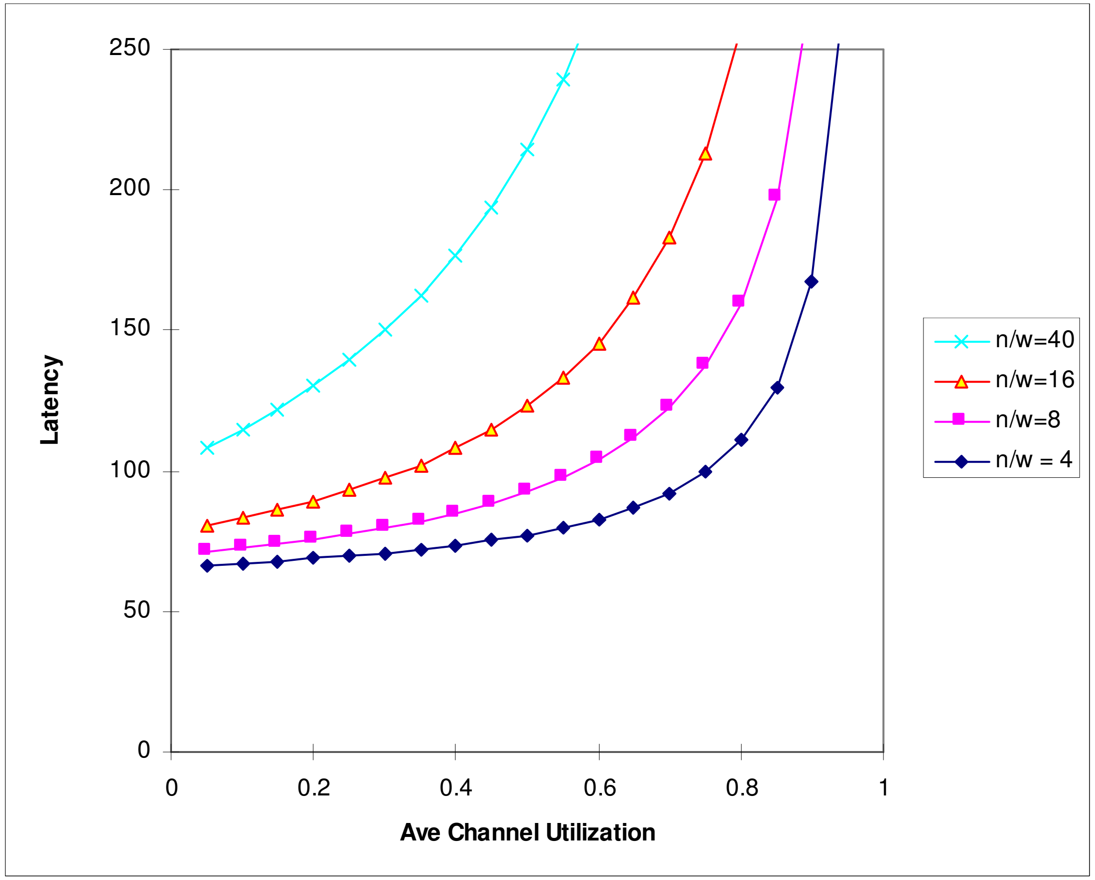
Latency under Contention



Optimal packet size? Channel utilization?

(p. 786 Fig. 10.17)

Saturation

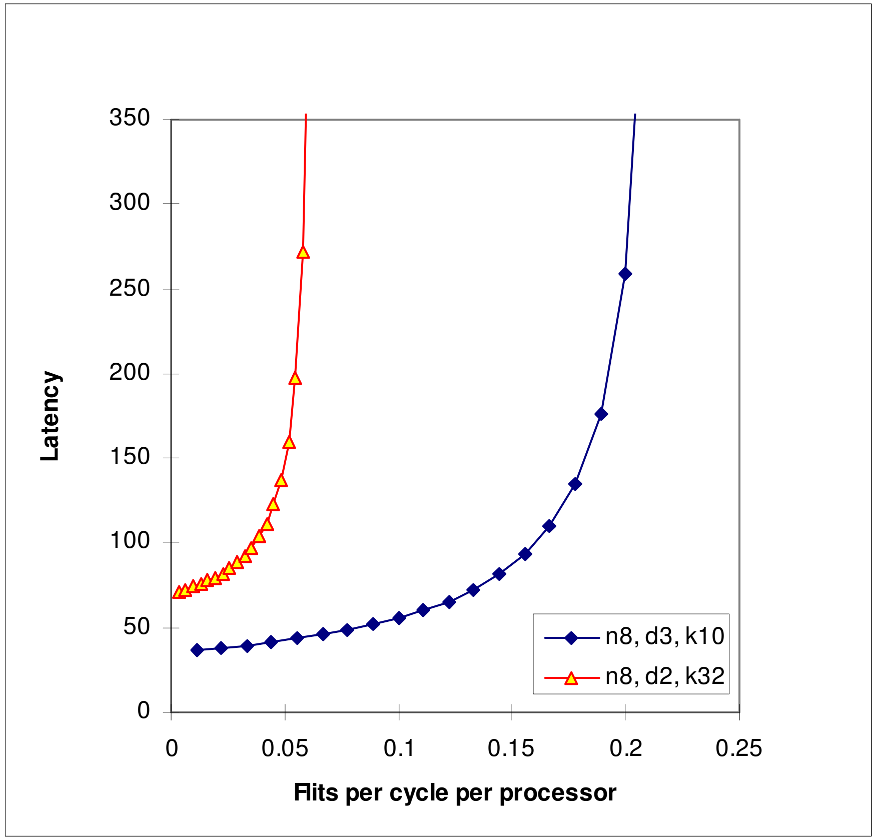


Fatter links shorten queuing delays

Adaptado dos slides da editora por Mario Côrtes – IC/Unicamp – 2009s2

(p. ??)

Phits per cycle



Higher degree network has larger available bandwidth

- cost?

(p. 787-789 Fig 10.18)

Adaptado dos slides da editora por Mario Côrtes – IC/Unicamp – 2009s2

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

Outline

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

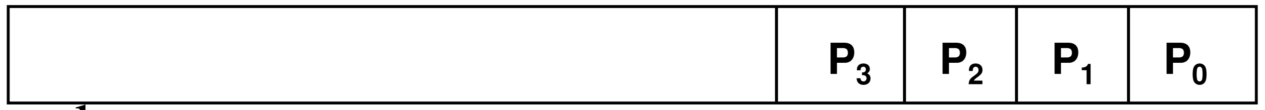
Simple arithmetic in regular topologies

- ex: 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 (primeiro 1 na distância relativa – XOR)

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 (índice para uma tabela em cada switch)
  - output = R[i]
- table also gives index for following hop
  - output, 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
- (pode ser implementada por table driven ou source based)

Minimal

- only selects shortest paths

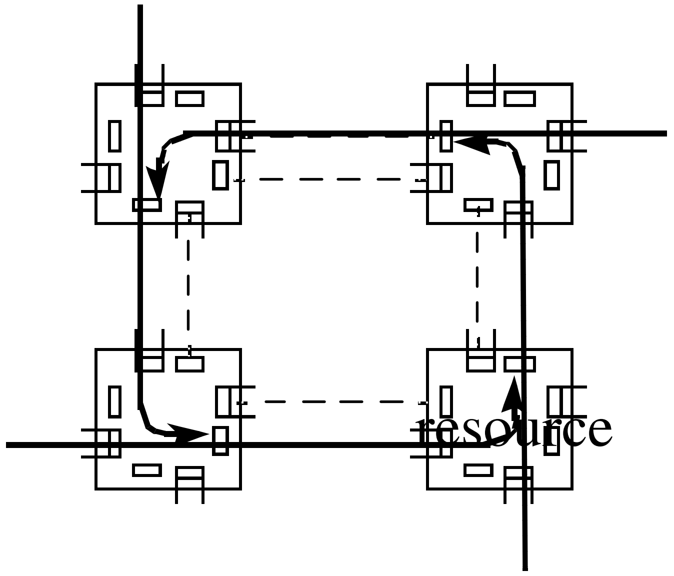
Deadlock free

- 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

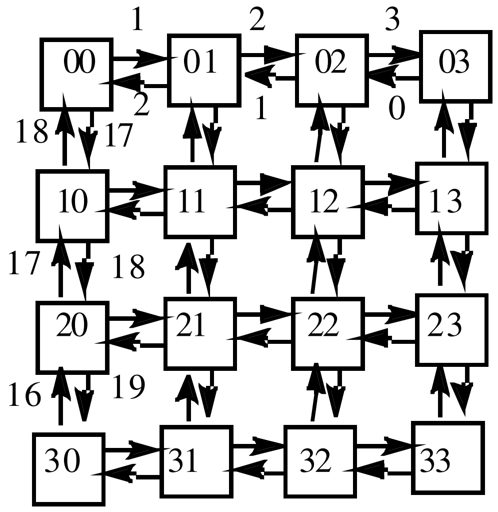
Example: k-ary 2D array

Theorem: x,y routing is deadlock free

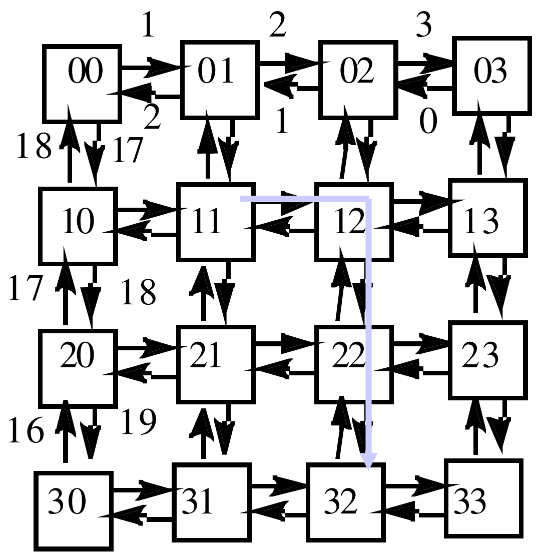
Numbering

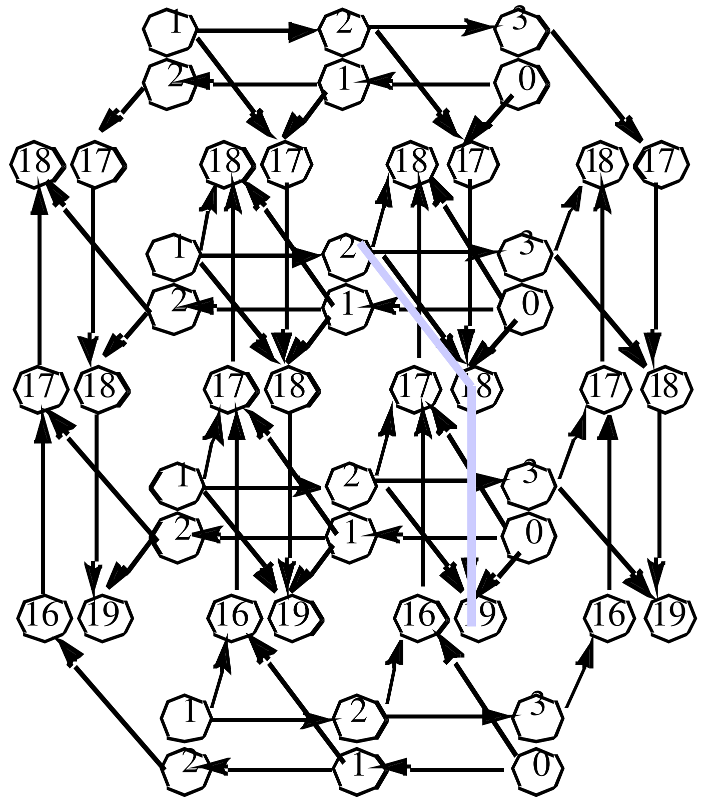
- +x channel (i,y) -> (i+1,y) gets i
- similarly for -x with 0 as most positive edge
- +y channel (x,j) -> (x,j+1) gets N+j
- similary for -y channels

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



Channel Dependence Graph





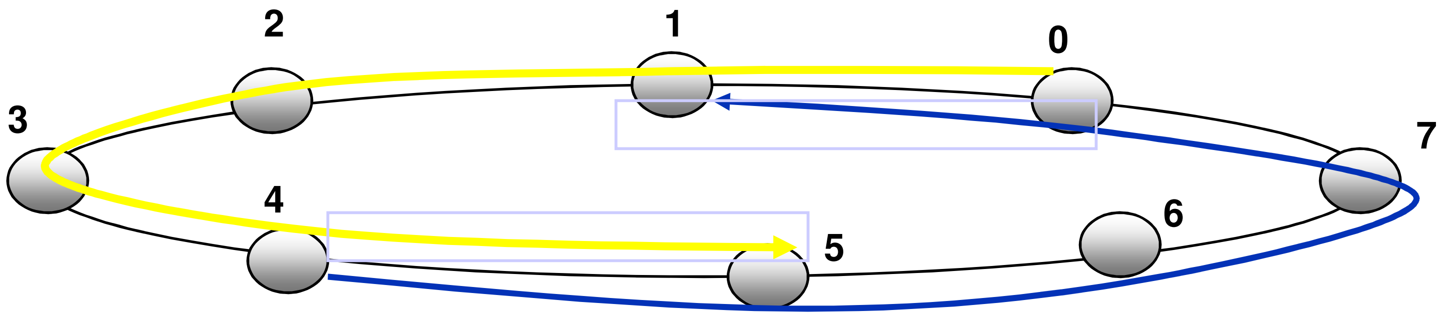
More examples

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?



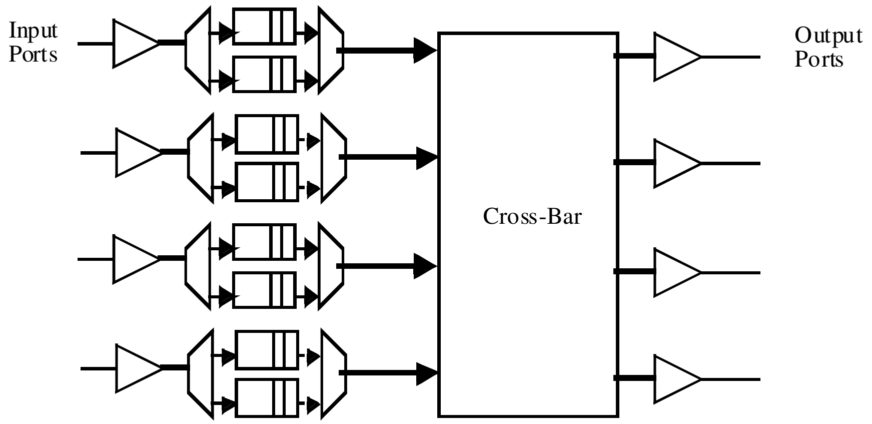
Deadlock free wormhole networks?

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

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

Idea: add channels!

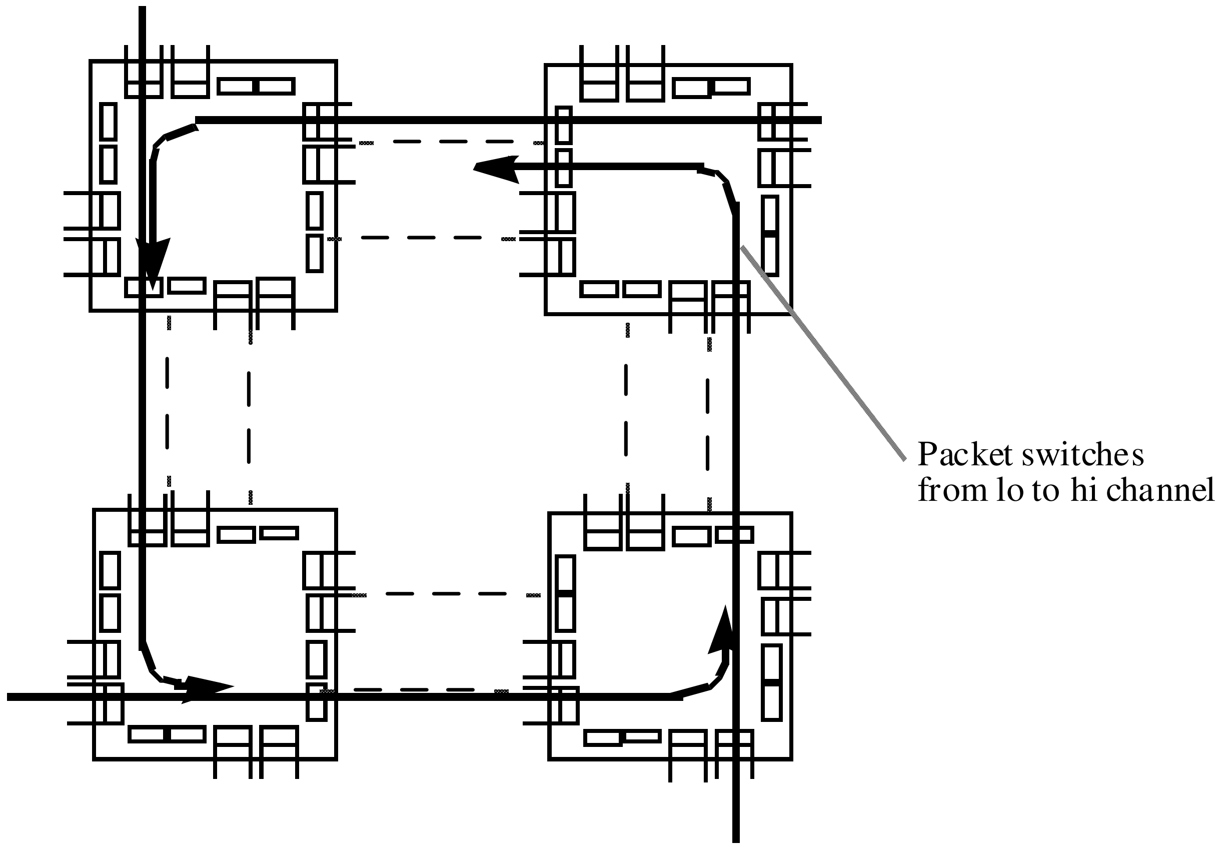
- provide multiple "virtual channels" to break dependence cycle
- good for BW too!



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

This adds nodes the the CDG, remove edges?

Breaking deadlock with virtual channels



Up*-Down* routing

Given any bidirectional network

Construct a spanning tree

Number of the nodes increasing from leaves to roots

UP increase node numbers

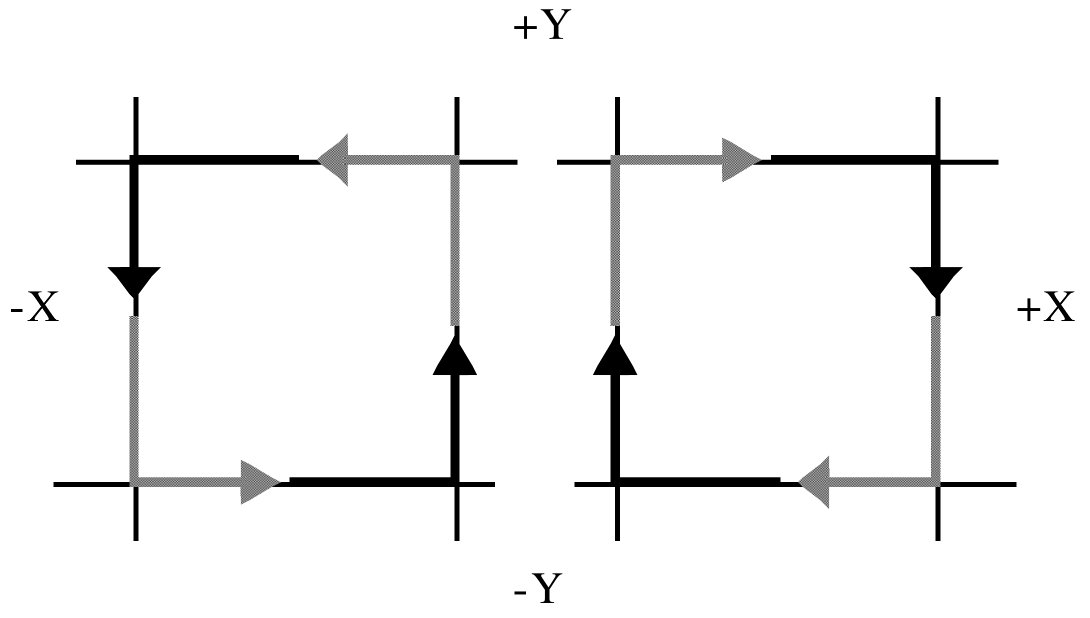
Any Source -> Dest by UP*-DOWN* route

- up edges, single turn, down edges

Performance?

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

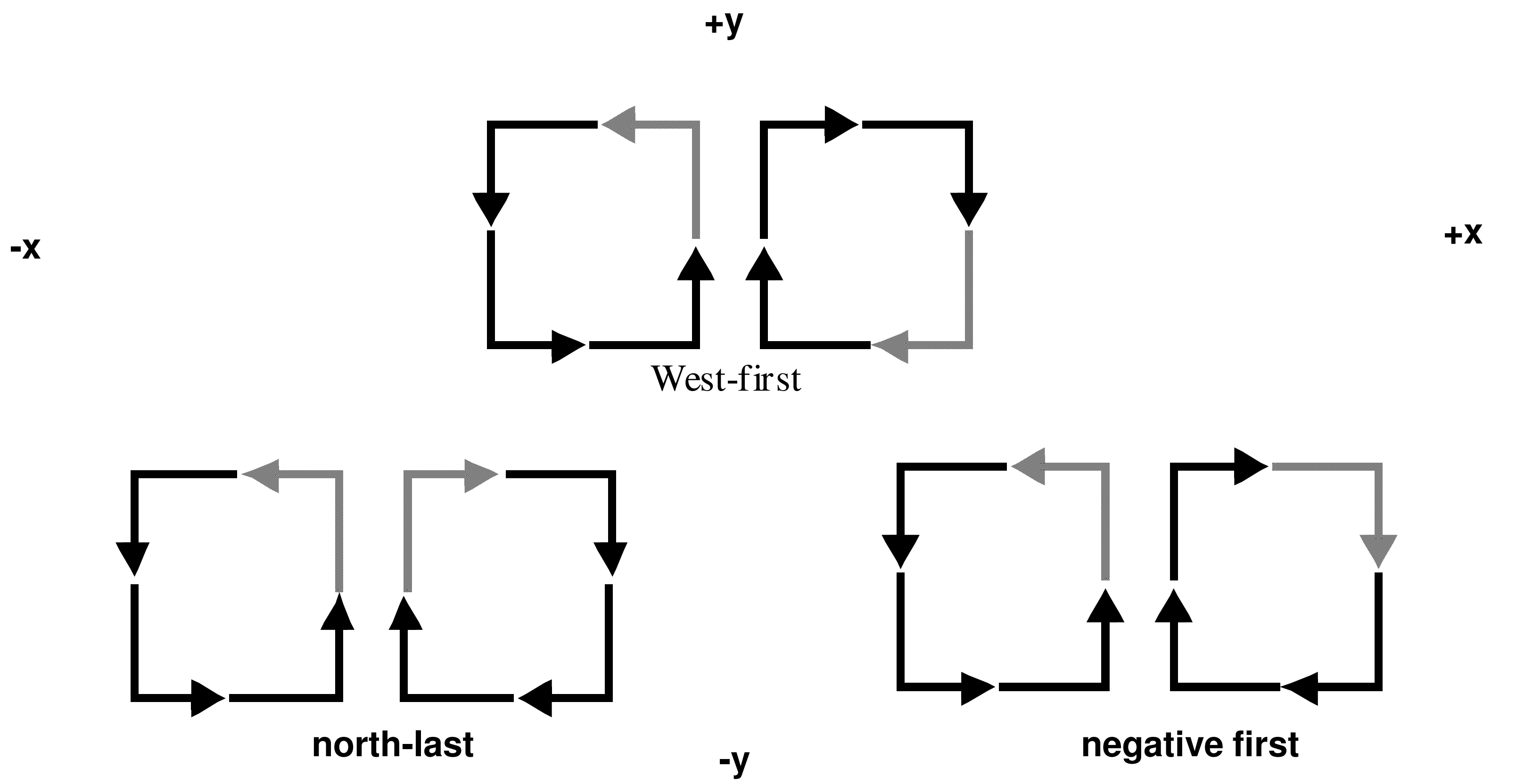
Turn Restrictions in X,Y



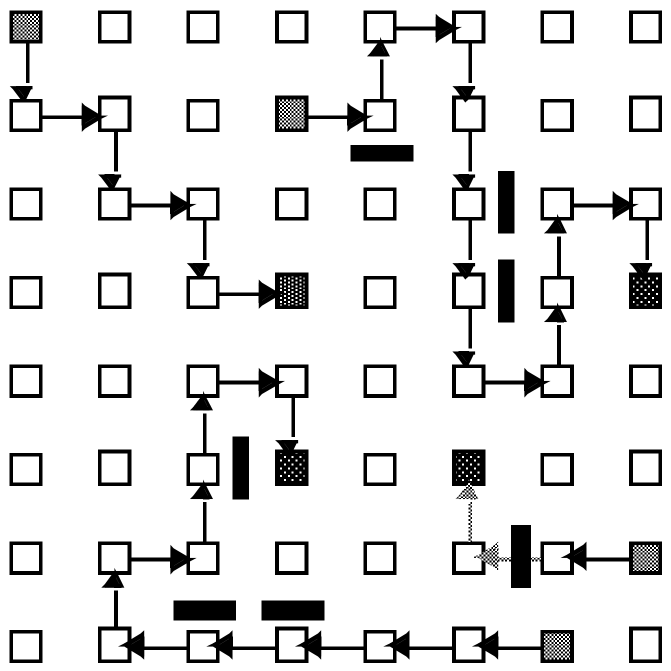
XY routing forbids 4 of 8 turns and leaves no room for adaptive routing

Can you allow more turns and still be deadlock free

Minimal turn restrictions in 2D



Example legal west-first routes



Can route around failures or congestion

Can combine turn restrictions with virtual channels

Adaptive Routing

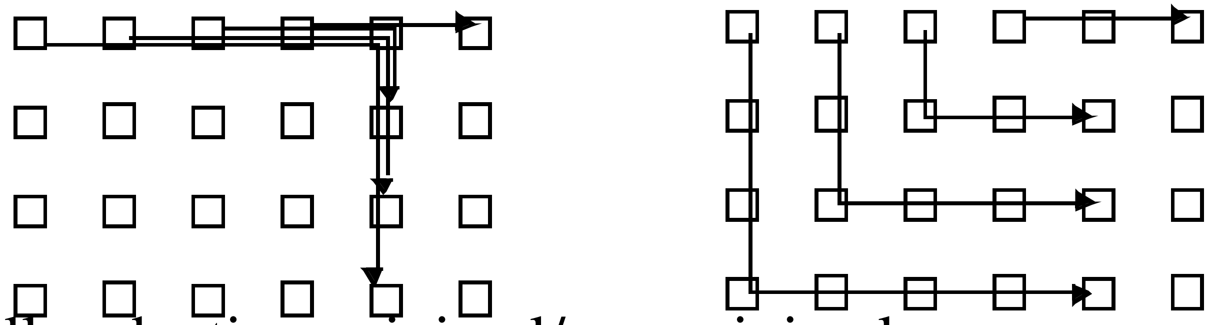
R: C x N x Σ -> C

Essential for fault tolerance

- at least multipath

Can improve utilization of the network

Simple deterministic algorithms easily run into bad permutations

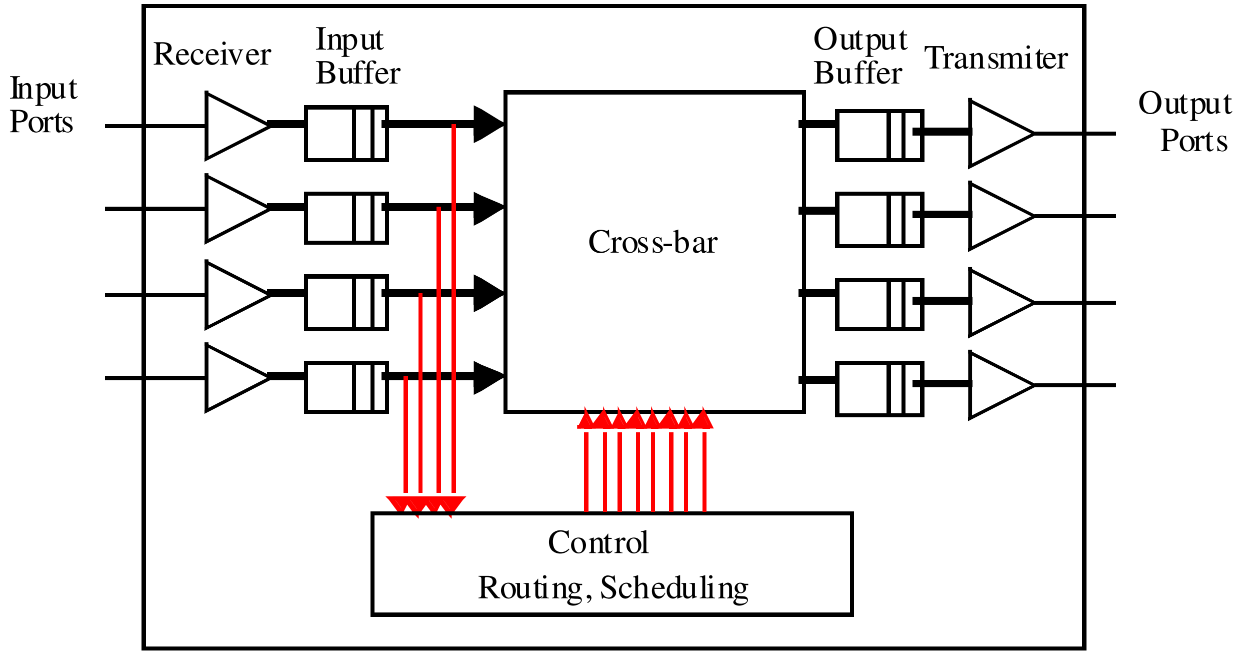


Fully/partially adaptive, minimal/non-minimal

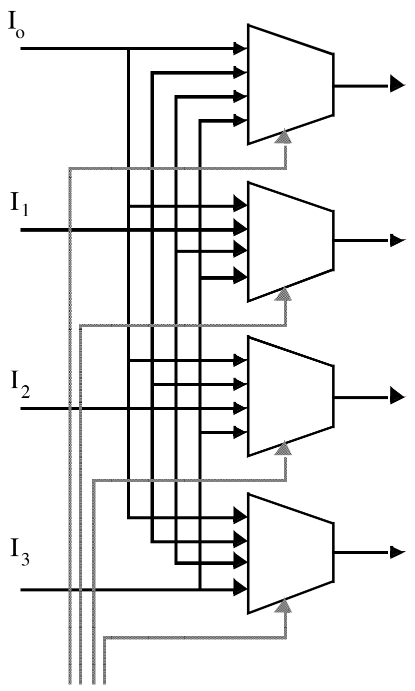
Can introduce complexity or anomolies

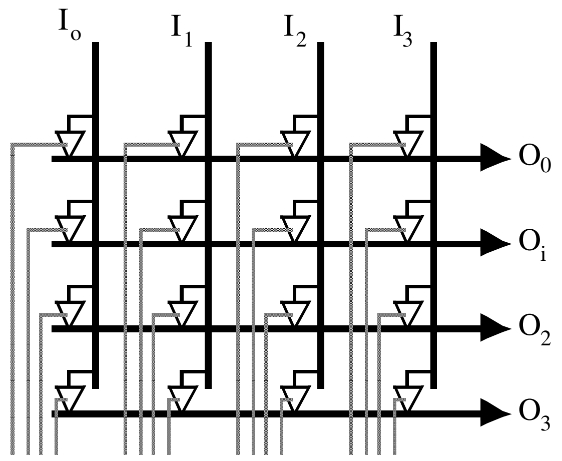
Little adaptation goes a long way!

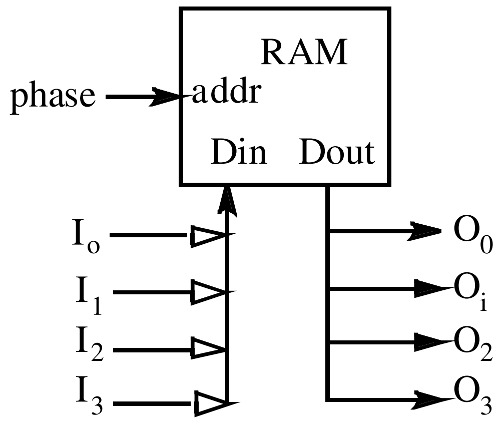
Switch Design



How do you build a crossbar

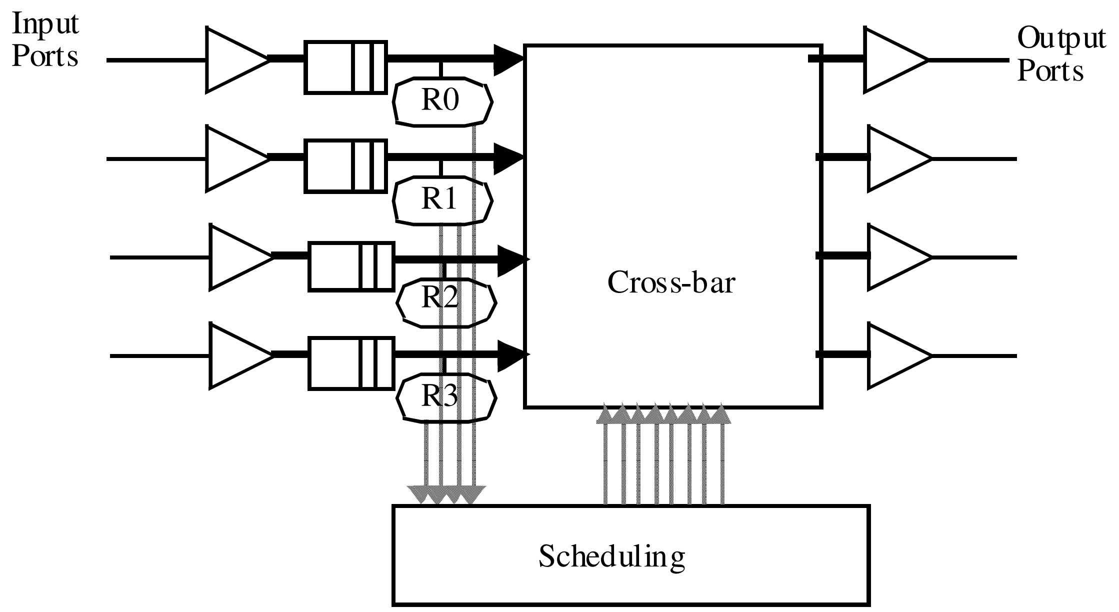






Adaptado dos slides da editora por Mario Côrtes – IC/Unicamp – 2009s2

Input buffered swtich



Independent routing logic per input

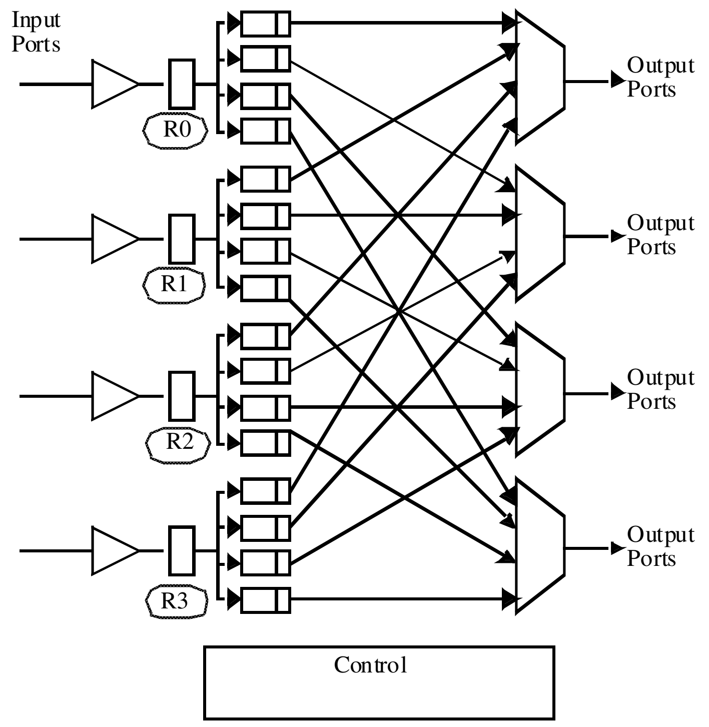
- FSM

Scheduler logic arbitrates each output

- priority, FIFO, random

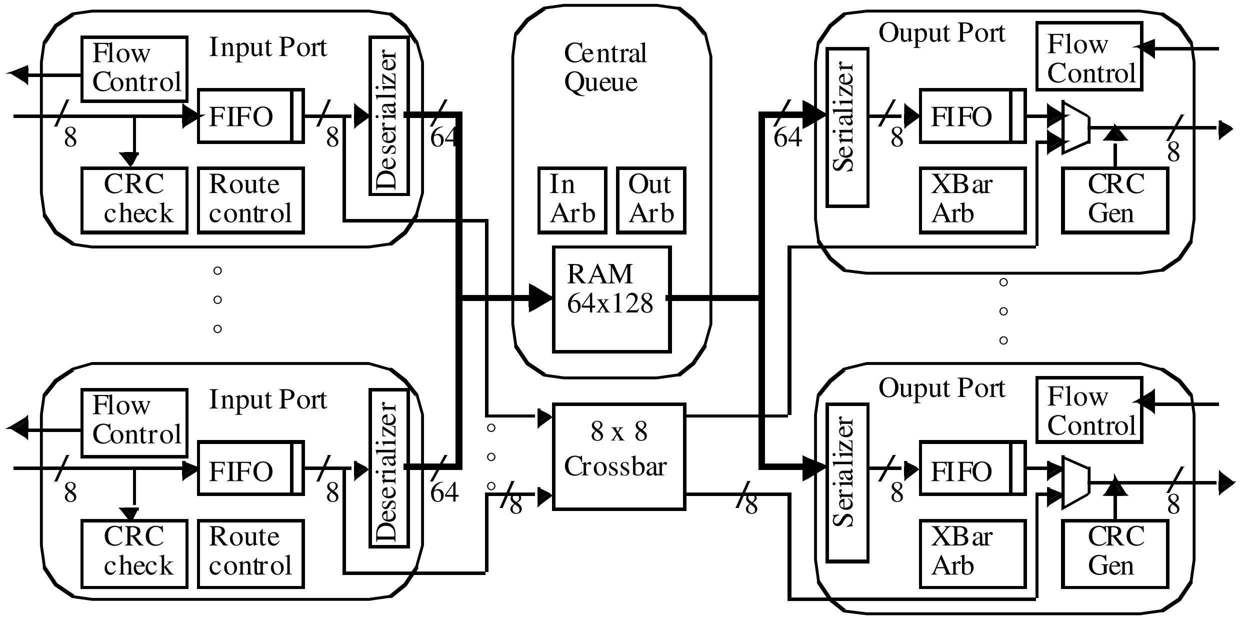
Head-of-line blocking problem

Output Buffered Switch



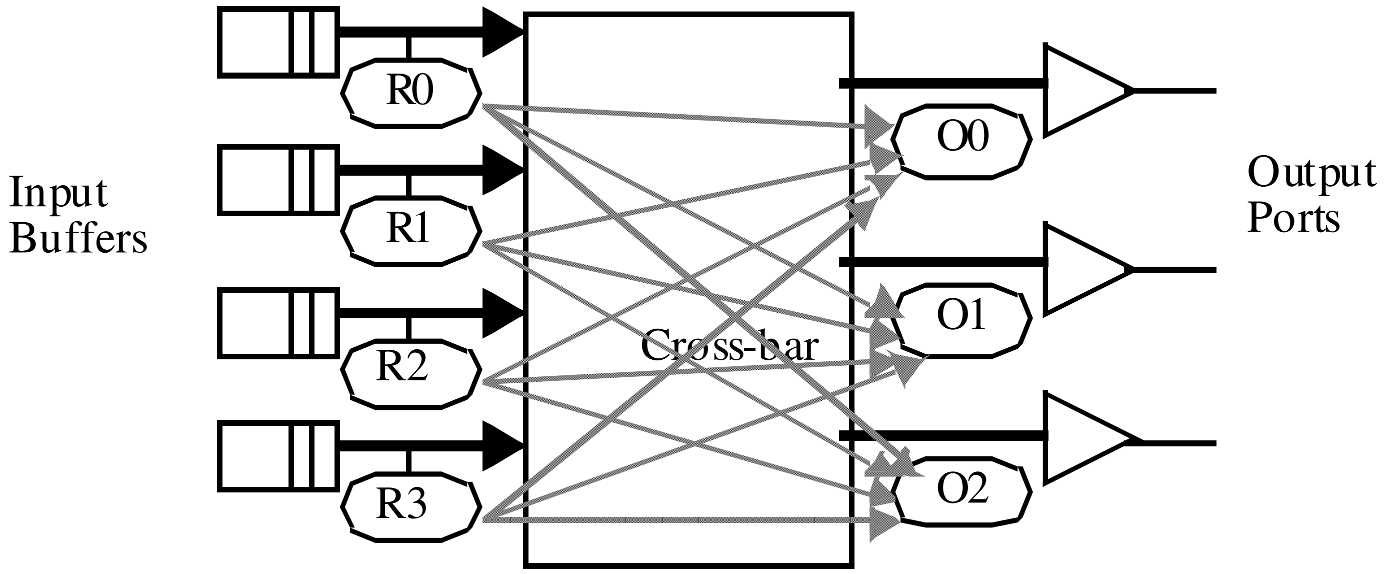
How would you build a shared pool?

Example: IBM SP vulcan switch



Many gigabit ethernet switches use similar design without the cut-through

Output scheduling



n independent arbitration problems?

- static priority, random, round-robin

Simplifications due to routing algorithm?

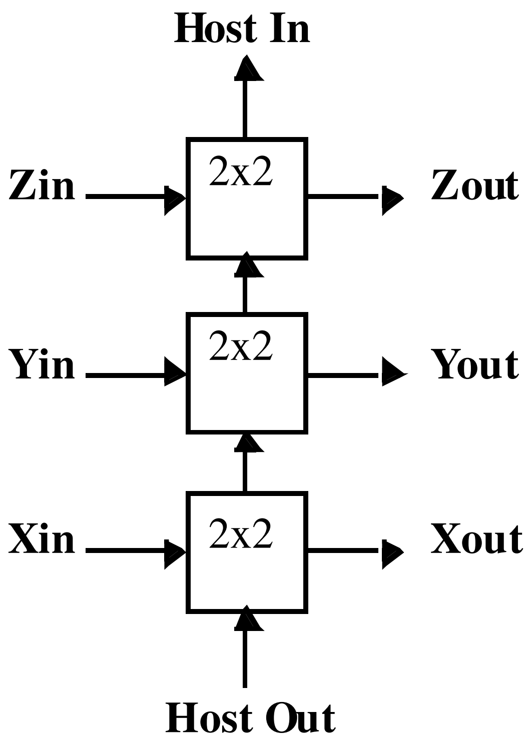
General case is max bipartite matching

Adaptado dos slides da editora por Mario Côrtes – IC/Unicamp – 2009s2

Stacked Dimension Switches

Dimension order on 3D cube?

Cube connected cycles?

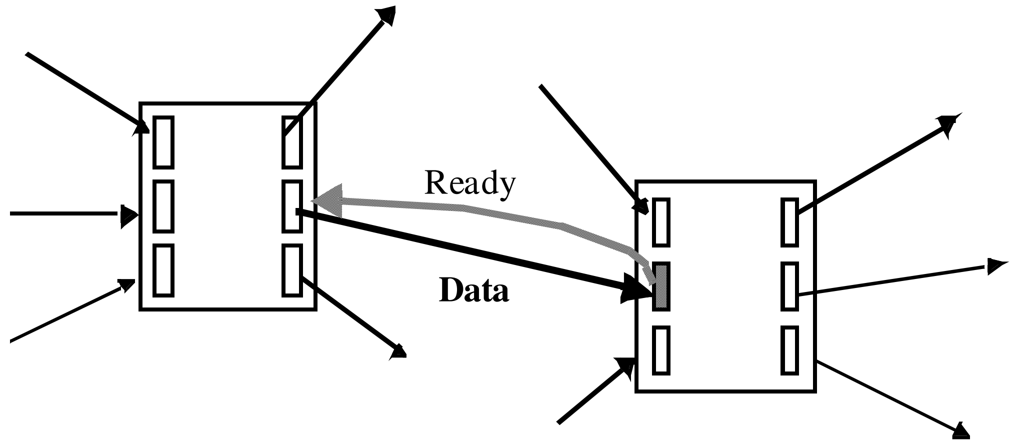


Flow Control

What do you do when push comes to shove?

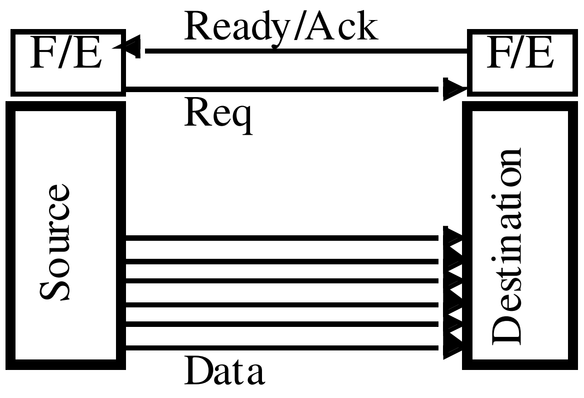
- ethernet: collision detection and retry after delay
- FDDI, token ring: arbitration token
- TCP/WAN: buffer, drop, adjust rate
- any solution must adjust to output rate

Link-level flow control



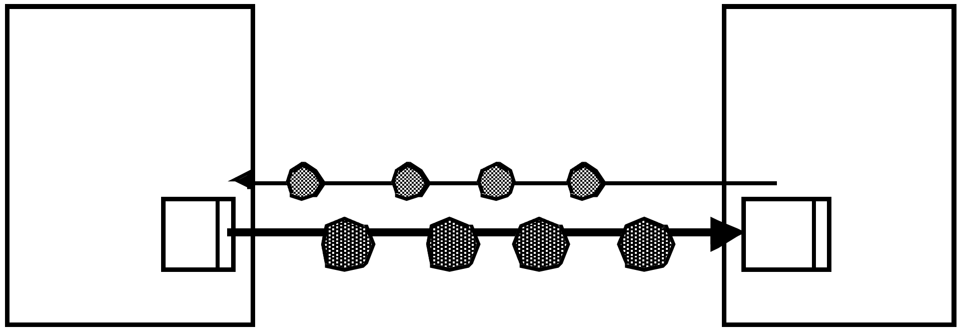
Examples

Short Links

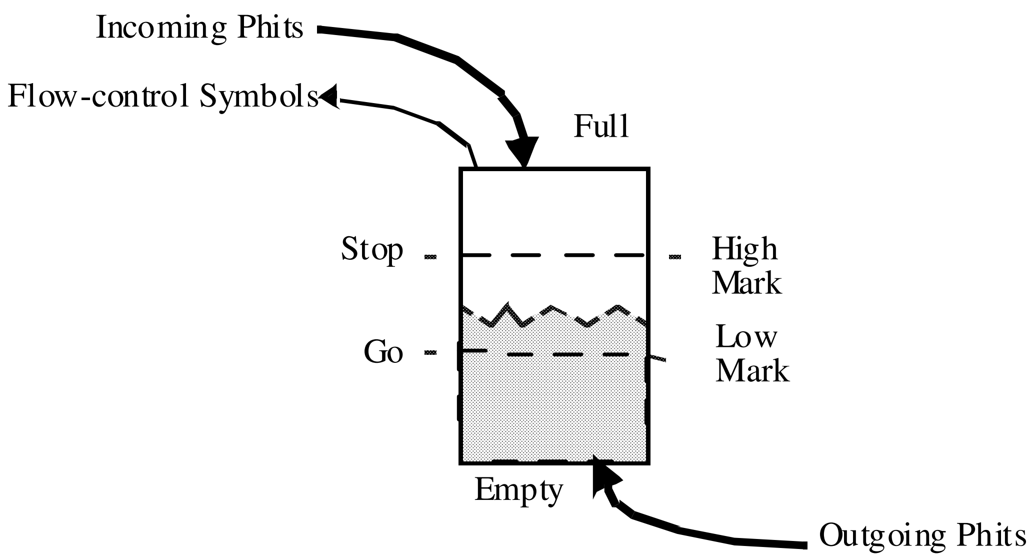


Long links

- several flits on the wire



Smoothing the flow



How much slack do you need to maximize bandwidth?

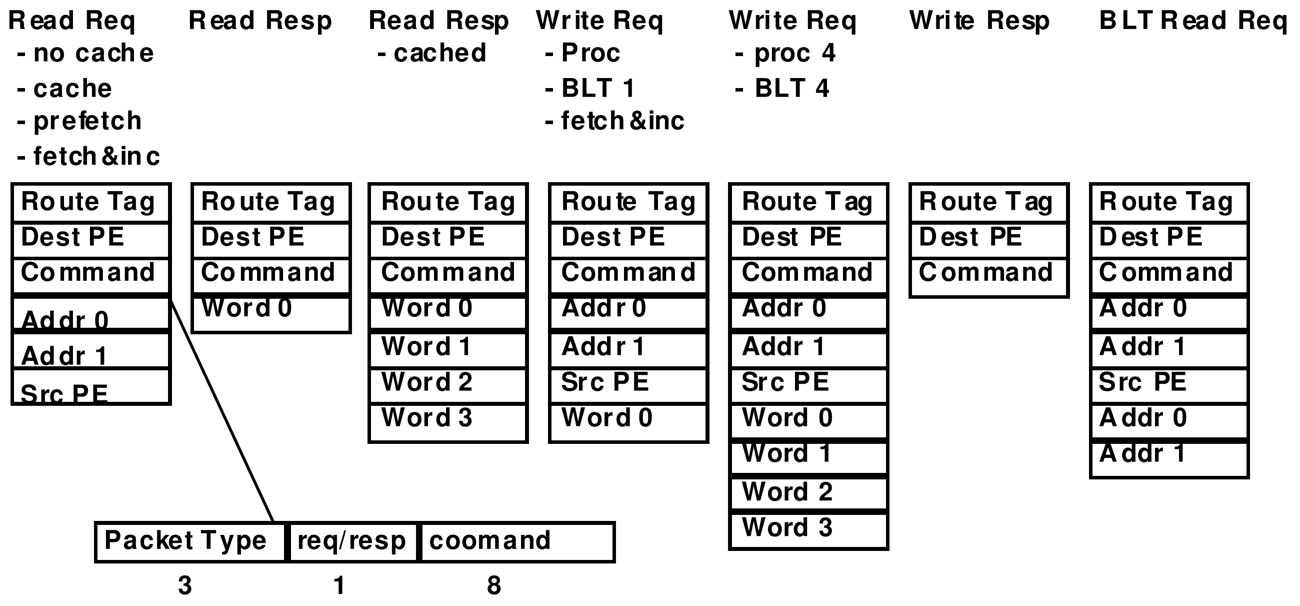
Link vs global flow control

Hot Spots

Global communication operations

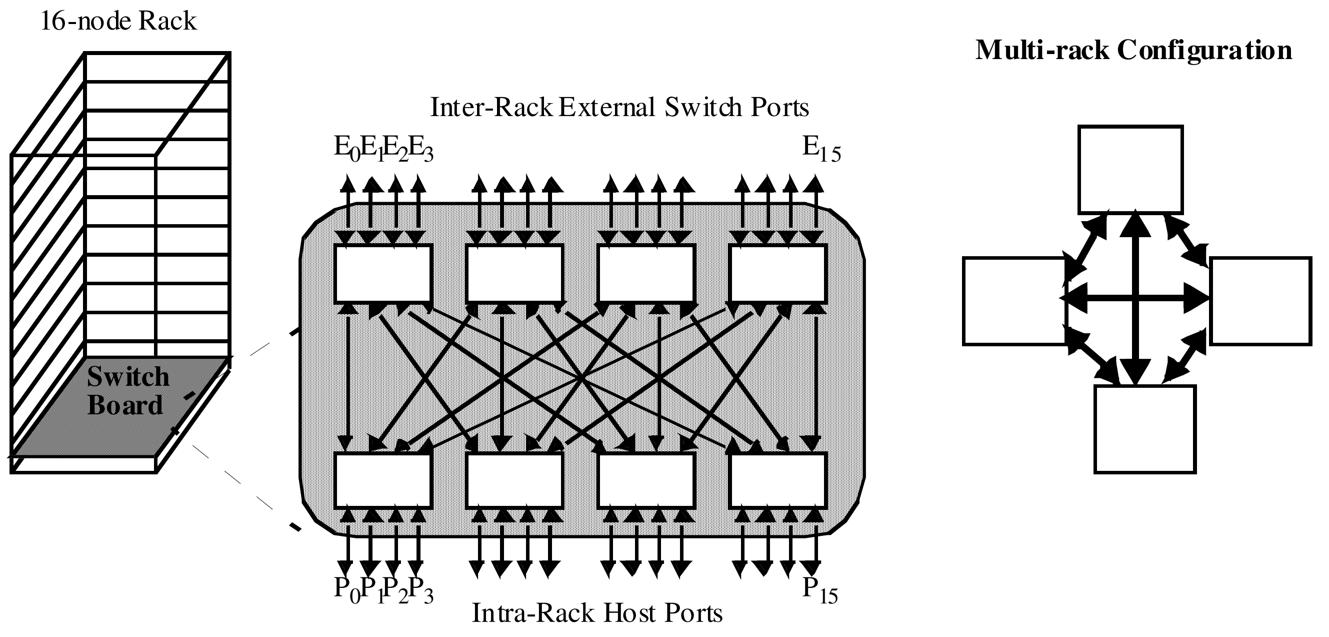
Natural parallel program dependences

Example: T3D



- 3D bidirectional torus, dimension order (NIC selected), virtual cut-through, packet sw.
- 16 bit x 150 MHz, short, wide, synch.
- rotating priority per output
- logically separate request/response
- 3 independent, stacked switches
- 8 16-bit flits on each of 4 VC in each directions

Example: SP



- 8-port switch, 40 MB/s per link, 8-bit phit, 16-bit flit, single 40 MHz clock
- packet sw, cut-through, no virtual channel, source-based routing
- variable packet <= 255 bytes, 31 byte FIFO per input, 7 bytes per output, 16 phit links
- 128 8-byte ‘chunks’ in central queue, LRU per output
- run in shadow mode

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

Medidas para mesh e torus

	Diâmetro	Dist. média	Bisection BW
Linear Array	N-1	2/3 N	1
Torus	N-1; N/2	N/2; N/3	1; 2
D-dim. Array	Σ (k_i - 1)	2/3 Σ k_i	(Π k_i) / k_{iMax}
D-dim. K-ary mesh	d. (k-1)	k.d 2/3	$k^{(d-1)}$
D-dim. K-ary torus (k-ary d-cube)	d. (k-1) ; d. k/2	d. k/2; d. k/3	$k^{(d-1)}$; 2 . $k^{(d-1)}$
Obs: Σ e Π são para i de 1 a d			

Valores separados por ponto-e-vírgula:

- **Primeiro: conexão uni-direcional**
- **Segundo: conexão bi-direcional**

Adaptado dos slides da editora por Mario Côrtes – IC/Unicamp – 2009s2