# Communication Performance in Network-on-Chips

Axel Jantsch
Royal Institute of Technology, Stockholm

November 24, 2004

# Overview

Introduction

Communication Performance

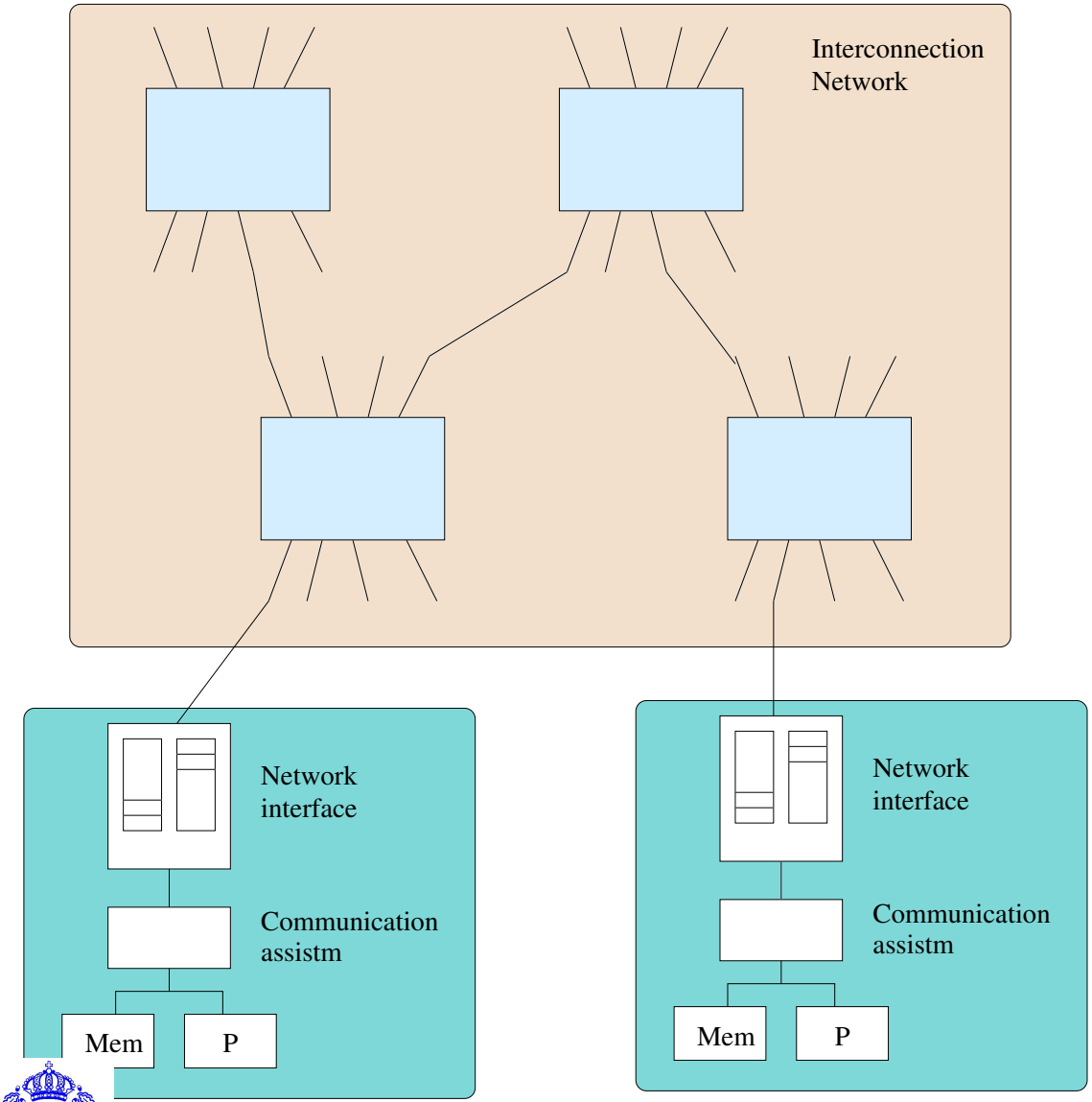
Organizational Structure

Interconnection Topologies

Trade-offs in Network Topology

Routing

# Introduction




- **Topology**: How switches and nodes are connected
- **Routing algorithm**: determines the route from source to destination
- **Switching strategy**: how a message traverses the route
- **Flow control**: Schedules the traversal of the message over time

# Basic Definitions

**Message** is the basic communication entity.

**Flit** is the basic flow control unit. A message consists of 1 or many flits.

**Phit** is the basic unit of the physical layer.

**Direct network** is a network where each switch connects to a node.

**Indirect network** is a network with switches not connected to any node.

**Hop** is the basic communication action from node to switch or from switch to switch.

**Diameter** is the length of the maximum shortest path between any two nodes measured in hops.

**Routing distance** between two nodes is the number of hops on a route.

**Average distance** is the average of the routing distance over all pairs of nodes.

# Basic Switching Techniques

**Circuit Switching** A real or virtual circuit establishes a direct connection between source and destination.

**Packet Switching** Each packet of a message is routed independently. The destination address has to be provided with each packet.

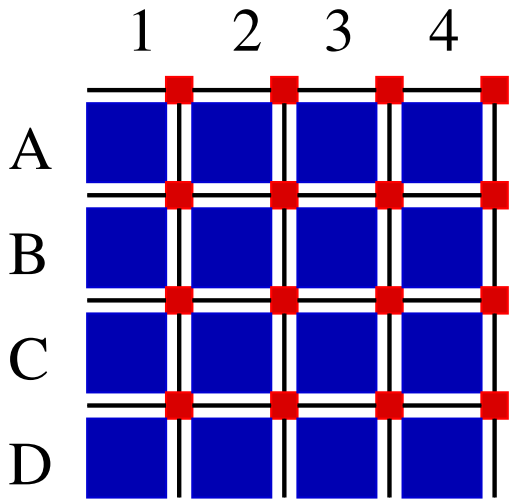
**Store and Forward Packet Switching** The entire packet is stored and then forwarded at each switch.

**Cut Through Packet Switching** The flits of a packet are pipelined through the network. The packet is not completely buffered in each switch.

**Virtual Cut Through Packet Switching** The entire packet is stored in a switch only when the header flit is blocked due to congestion.

**Wormhole Switching** is cut through switching and all flits are blocked on the spot when the header flit is blocked.

# Latency




$$\text{Time}(n) = \text{Admission} + \text{ChannelOccupancy} + \text{RoutingDelay} + \text{ContentionDelay}$$

**Admission** is the time it takes to emit the message into the network.

**ChannelOccupancy** is the time a channel is occupied.

**RoutingDelay** is the delay for the route.

**ContentionDelay** is the delay of a message due to contention.

# Channel Occupancy

$$\textbf{ChannelOccupancy} = \frac{n + n_E}{b}$$

$n$ ... message size in bits
$n_E$ ... envelop size in bits
$b$ ... raw bandwidth of the channel

# Routing Delay

Store and Forward: $T_{sf}(n,h) = h(\frac{n}{b} + \Delta)$

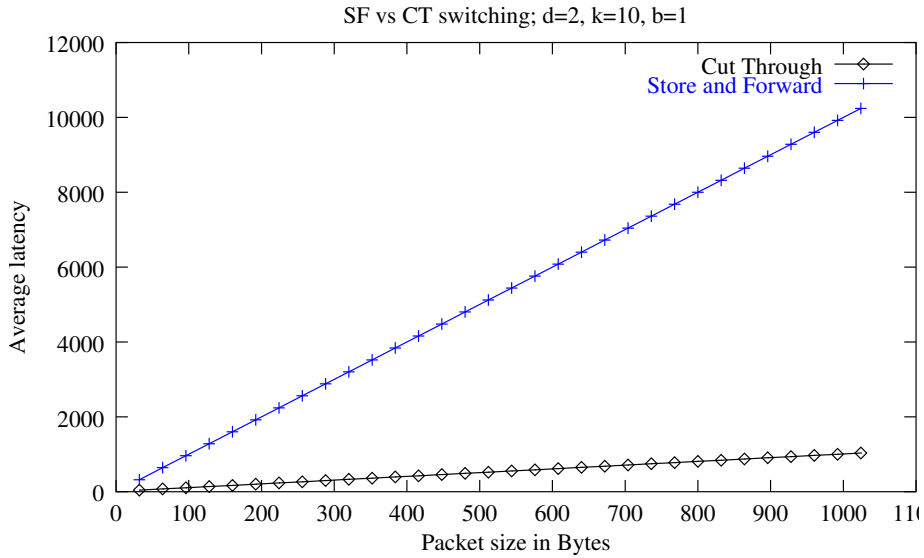
Circuit Switching: $T_{cs}(n,h) = \frac{n}{b} + h\Delta$

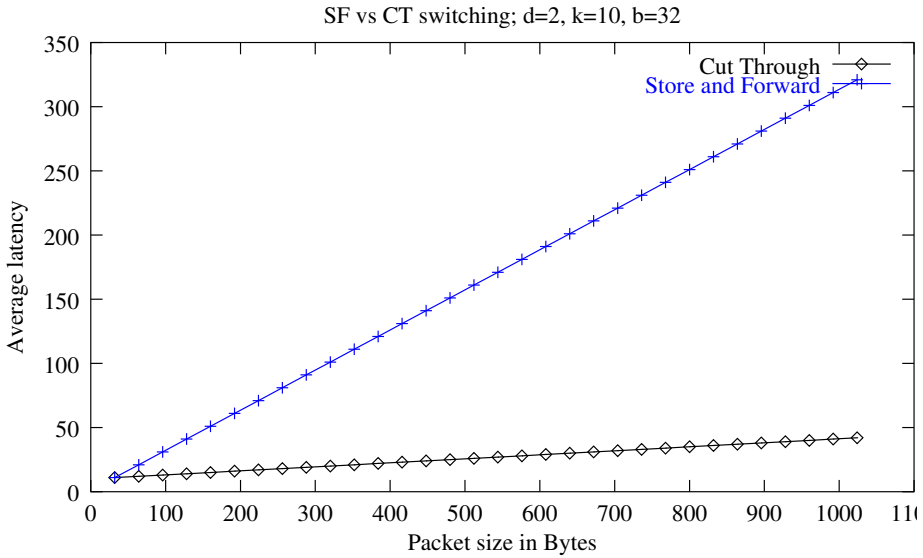
Cut Through: $T_{ct}(n,h) = \frac{n}{b} + h\Delta$

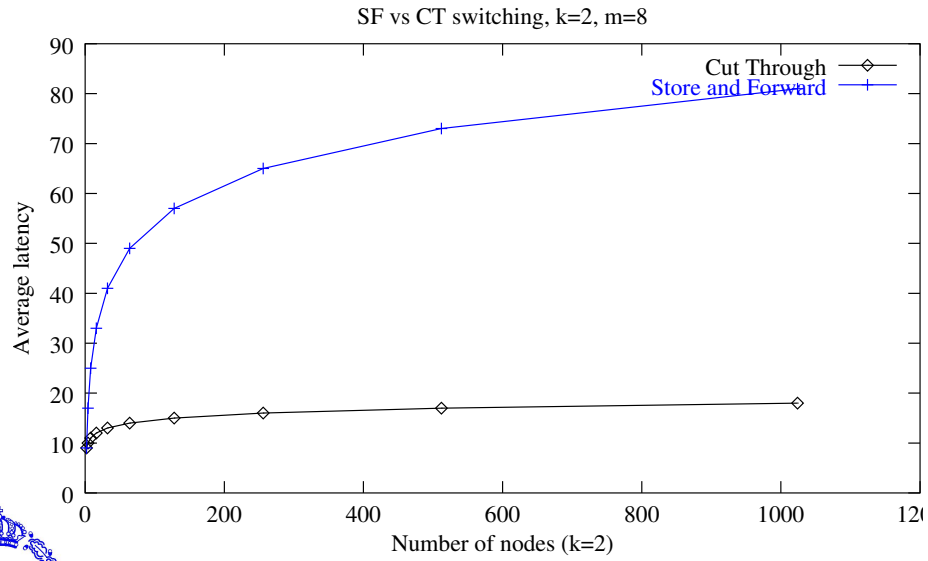
Store and Forward with fragmented packets: $T_{sf}(n,h,n_p) = \frac{n-n_p}{b} + h(\frac{n_p}{b} + \Delta)$

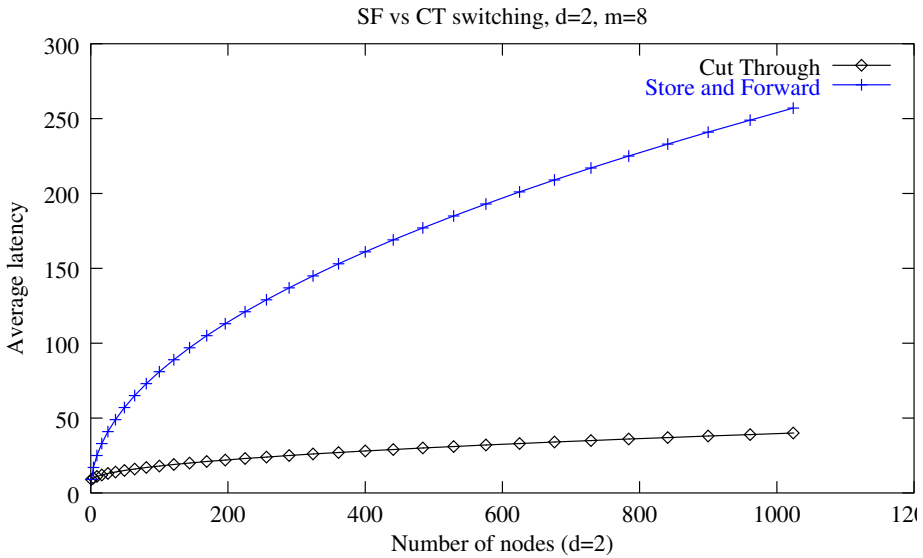
$n$ ... message size in bits
$n_p$ ... size of message fragments in bits
$h$ ... number of hops
$b$ ... raw bandwidth of the channel
$\Delta$ ... switching delay per hop

# Routing Delay: Store and Forward vs Cut Through



# Local and Global Bandwidth

**Local bandwidth** $= b\left(\frac{n}{n+n_E+w\Delta}\right)$

**Total bandwidth** $= Cb[\text{bits/second}] = Cw[\text{bits/cycle}] = C[\text{phits/cycle}]$

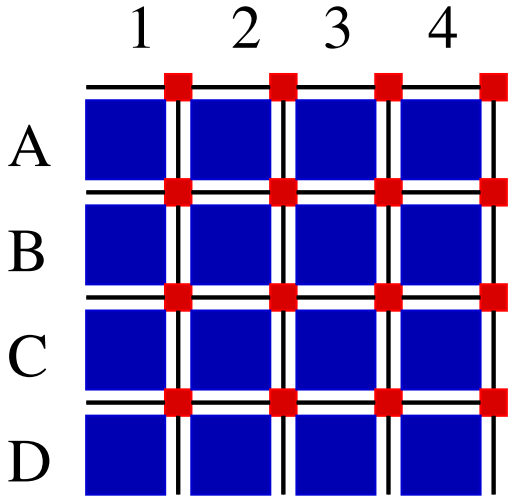
**Bisection bandwidth** ... minimum bandwidth to cut the net into two equal parts.

$b$ ... raw bandwidth of a link;
$n$ ... message size;
$n_E$ ... size of message envelope;
$w$ ... link bandwidth per cycle;

$\Delta$ ... switching time for each switch in cycles;
$w\Delta$ ... bandwidth lost during switching;
$C$ ... total number of channels;

For a $k \times k$ mesh with bidirectional channels:

Total bandwidth $= (4k^2 - 4k)b$
Bisection bandwidth $= 2kb$



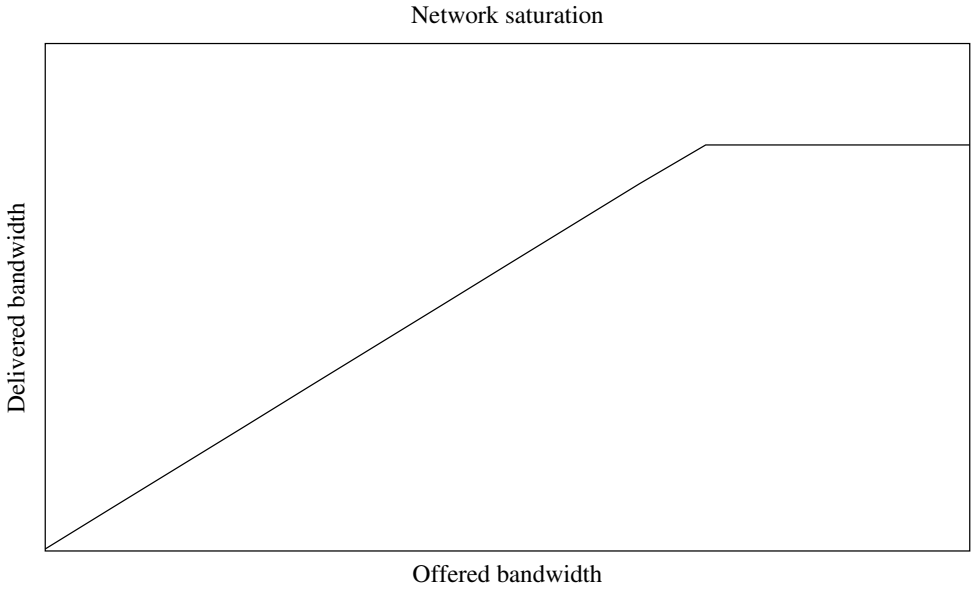
# Link and Network Utilization

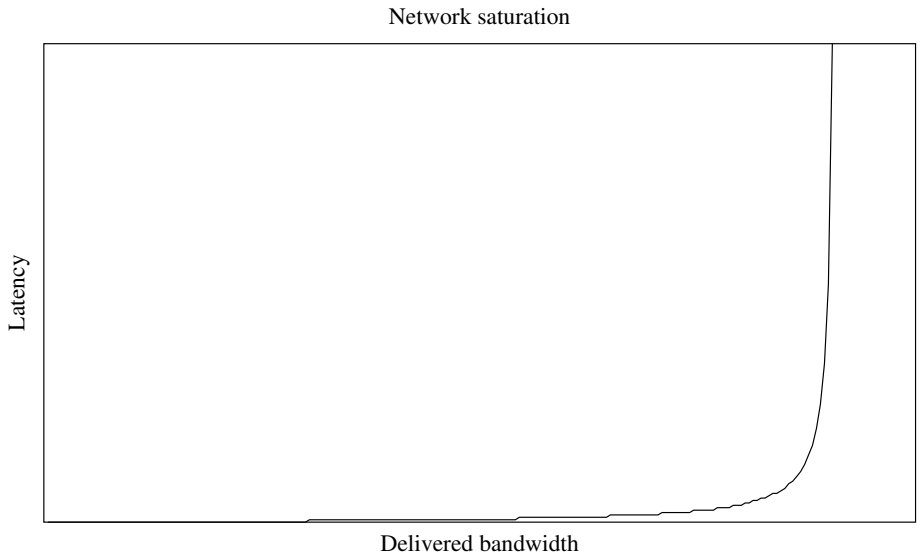
**total load on the network:** $L = \frac{Nhl}{M}$[phits/cycle]

**load per channel:** $\rho = \frac{Nhl}{MC}$[phits/cycle] $\leq 1$

$M$ ... each host issues a packet every $M$ cycles
$C$ ... number of channels
$N$ ... number of nodes
$h$ ... average routing distance
$l = n/w$ ... number of cycles a message occupies a channel
$n$ ... average message size
$w$ ... bitwidth per channel

# Network Saturation




Typical saturation points are between 40% and 70%.
The saturation point depends on

- Traffic pattern
- Stochastic variations in traffic
- Routing algorithm

# Organizational Structure

- Link
- Switch
- Network Interface

# Link

**Short link** At any time there is only one data word on the link.

**Long link** Several data words can travel on the link simultaneously.

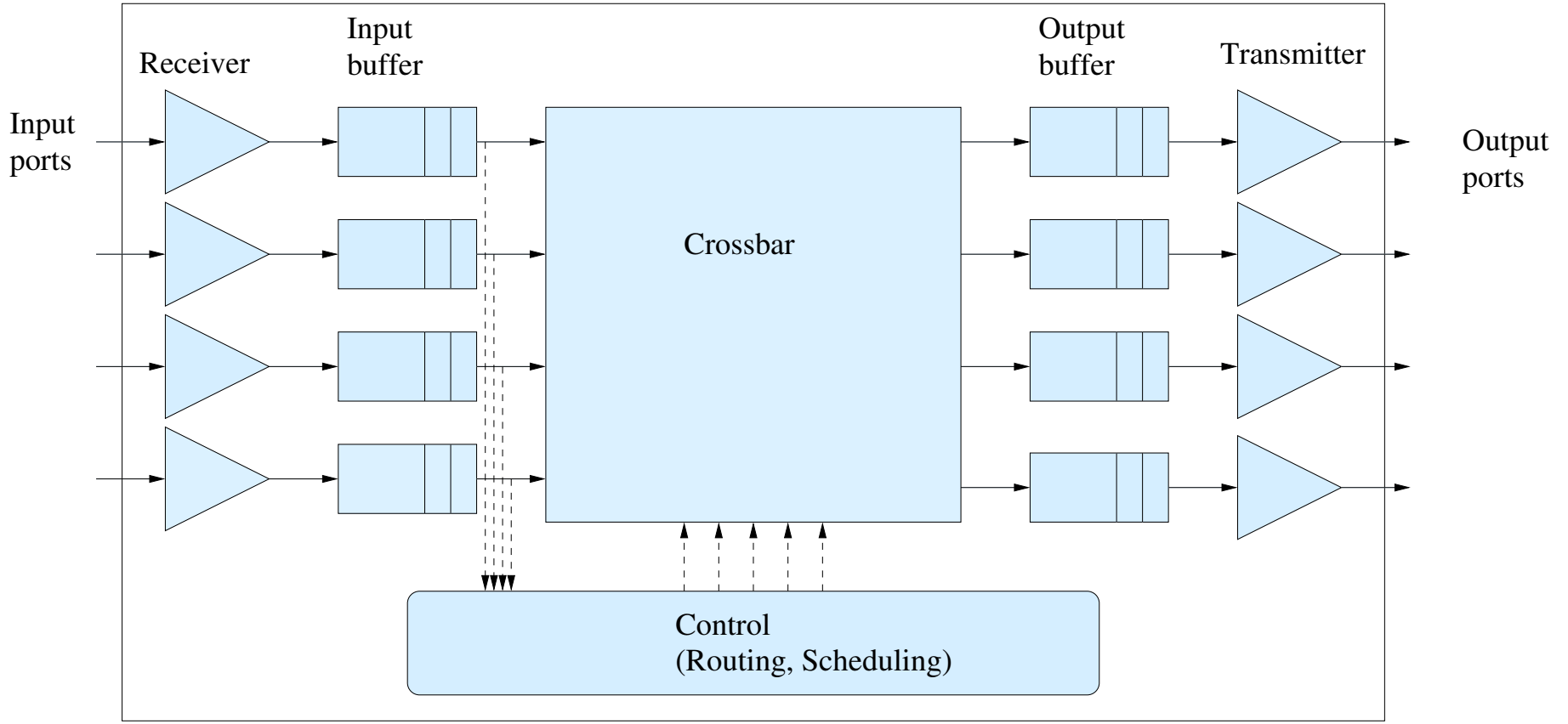
**Narrow link** Data and control information is multiplexed on the same wires.

**Wide link** Data and control information is transmitted in parallel and simultaneously.

**Synchronous clocking** Both source and destination operate on the same clock.

**Asynchronous clocking** The clock is encoded in the transmitted data to allow the receiver to sample at the right time instance.

# Switch


Input
ports
Receiver
Input
buffer
Crossbar
Output
buffer
Transmitter
Output
ports
Control
(Routing, Scheduling)

# Switch Design Issues

**Degree:** number of inputs and outputs;

**Buffering**

- Input buffers
- Output buffers
- Shared buffers

**Routing**

- Source routing
- Deterministic routing
- Adaptive routing

**Output scheduling**

**Deadlock handling**

**Control flow**

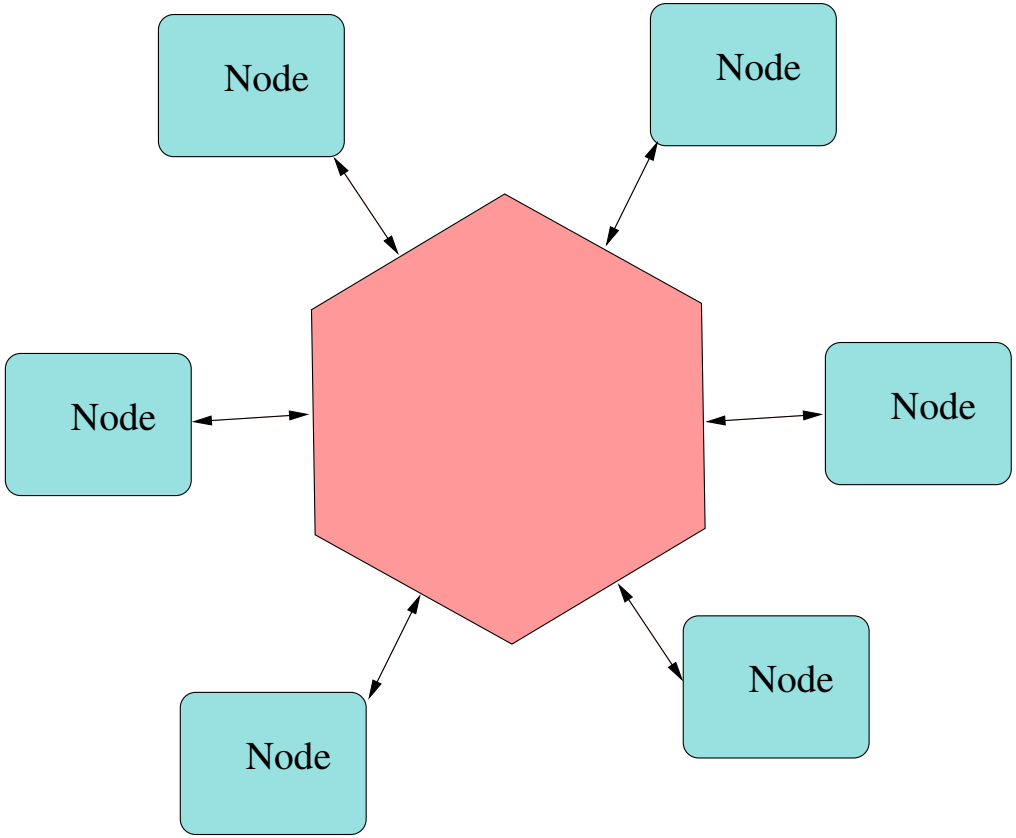
# Network Interface

- Admission protocol
- Reception obligations
- Buffering
- Assembling and disassembling of messages
- Routing
- Higher level services and protocols

# Interconnection Topologies

- Fully connected networks
- Linear arrays and rings
- Multidimensional meshes and tori
- Trees
- Butterflies

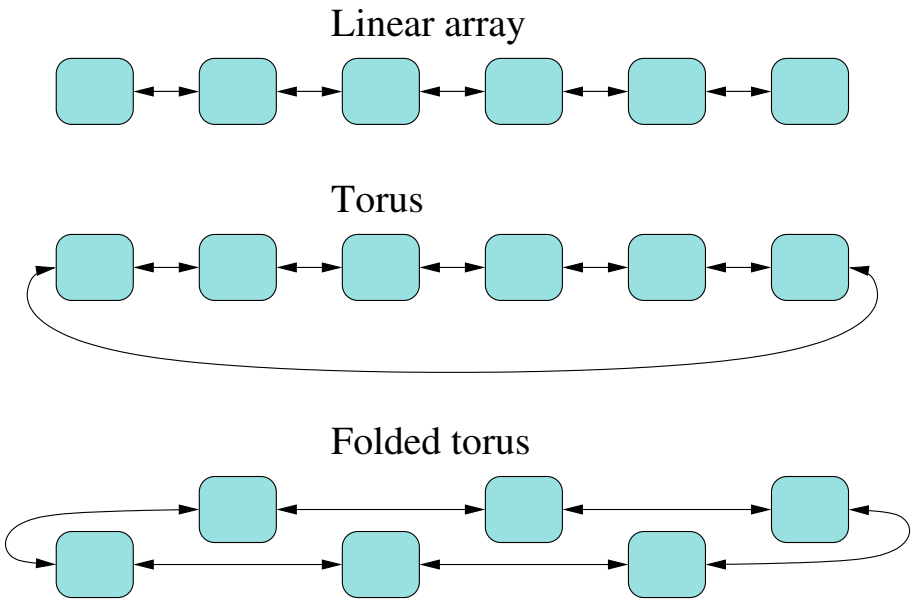
# Fully Connected Networks




| | | | |
|---|---|---|---|
| Bus: | switch degree | = | $N$ |
| | diameter | = | $1$ |
| | distance | = | $1$ |
| | network cost | = | $O(N)$ |
| | total bandwidth | = | $b$ |
| | bisection bandwidth | = | $b$ |

| | | | |
|---|---|---|---|
| Crossbar: | switch degree | = | $N$ |
| | diameter | = | $1$ |
| | distance | = | $1$ |
| | network cost | = | $O(N^2)$ |
| | total bandwidth | = | $Nb$ |
| | bisection bandwidth | = | $Nb$ |

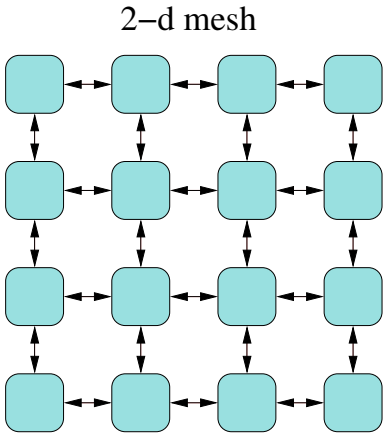
# Linear Arrays and Rings

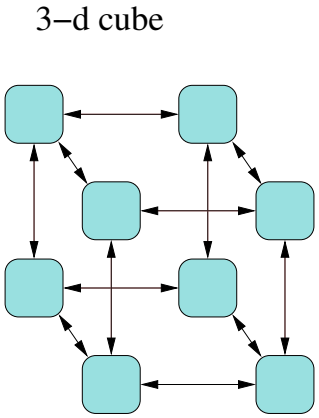


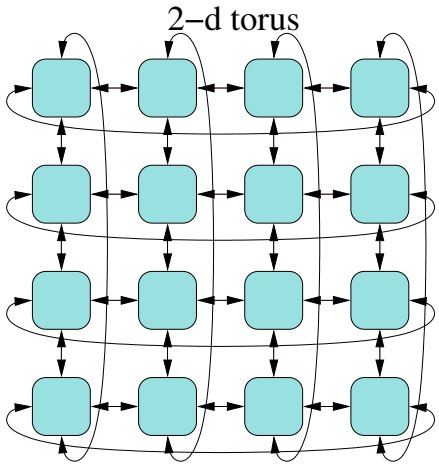

| | | | |
|---|---|---|---|
| Linear array: | switch degree | $=$ | $2$ |
| | diameter | $=$ | $N-1$ |
| | distance | $\sim$ | $2/3N$ |
| | network cost | $=$ | $O(N)$ |
| | total bandwidth | $=$ | $2(N-1)b$ |
| | bisection bandwidth | $=$ | $2b$ |

| | | | |
|---|---|---|---|
| Torus: | switch degree | $=$ | $2$ |
| | diameter | $=$ | $N/2$ |
| | distance | $\sim$ | $1/3N$ |
| | network cost | $=$ | $O(N)$ |
| | total bandwidth | $=$ | $2Nb$ |
| | bisection bandwidth | $=$ | $4b$ |

# Multidimensional Meshes and Tori



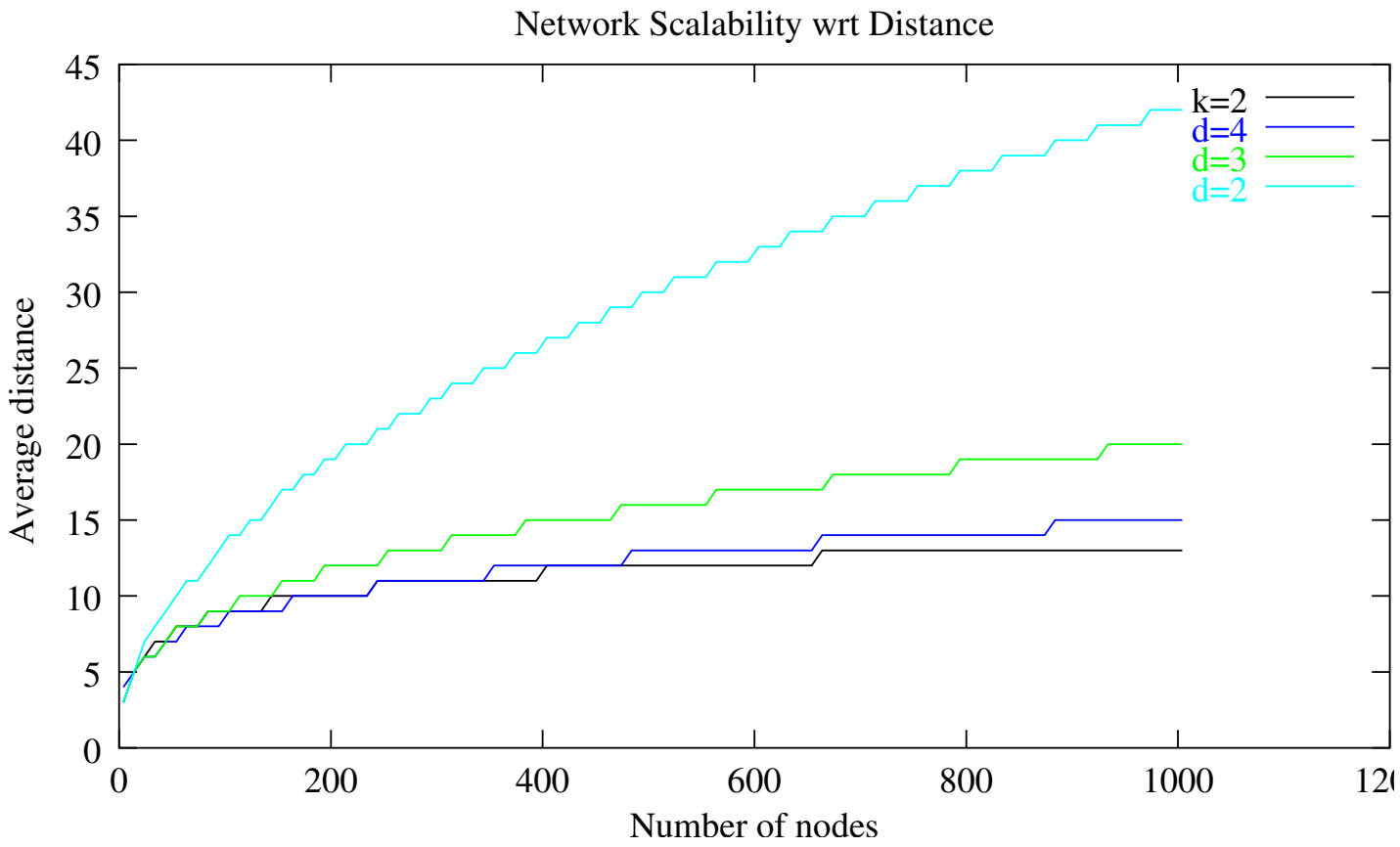



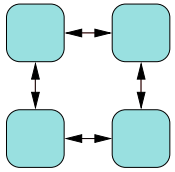

**$k$-ary $d$-cubes** are $d$-dimensional tori with unidirectional links and $k$ nodes in each dimension:

| | | |
|---|---|---|
| number of nodes $N$ | $=$ | $k^d$ |
| switch degree | $=$ | $d$ |
| diameter | $=$ | $d(k-1)$ |
| distance | $\sim$ | $d\frac{1}{2}(k-1)$ |
| network cost | $=$ | $O(N)$ |
| total bandwidth | $=$ | $2Nb$ |
| bisection bandwidth | $=$ | $2k^{(d-1)}b$ |

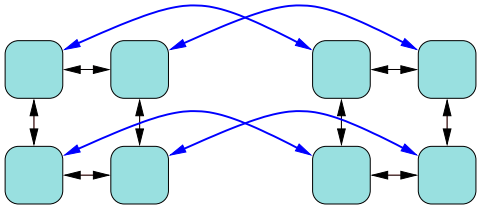
# Routing Distance in $k$-ary $n$-Cubes



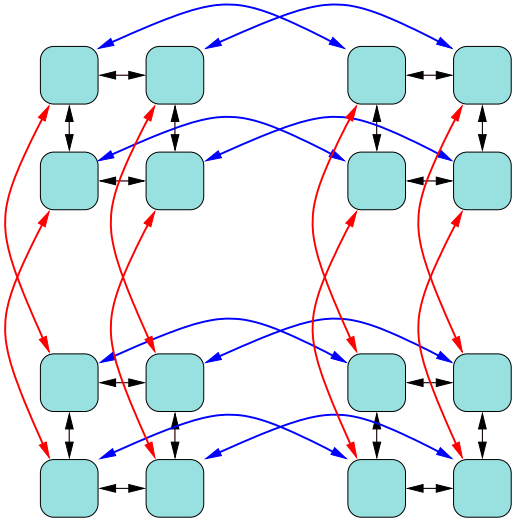
# Projecting High Dimensional Cubes



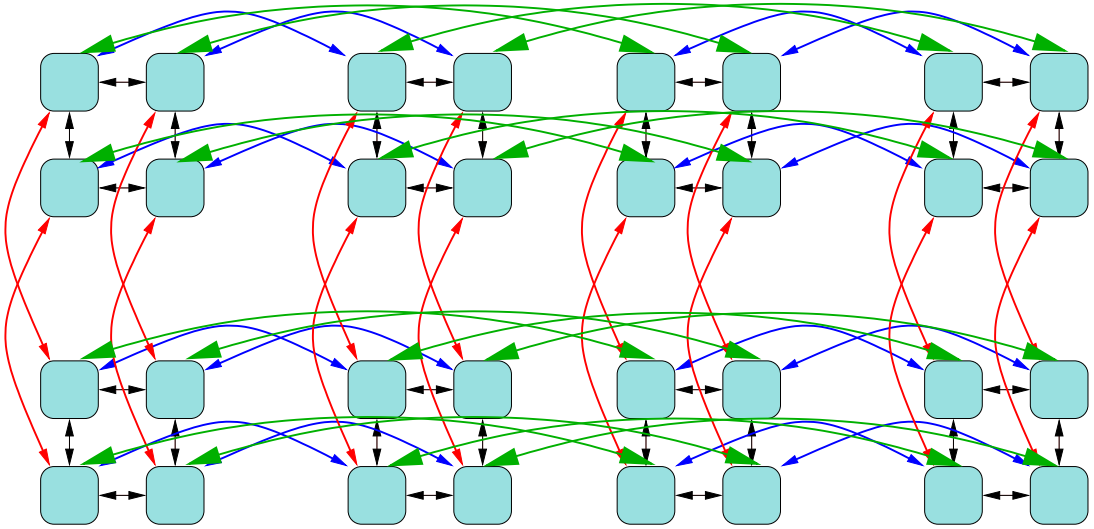
2–ary 2–cube



2–ary 3–cube

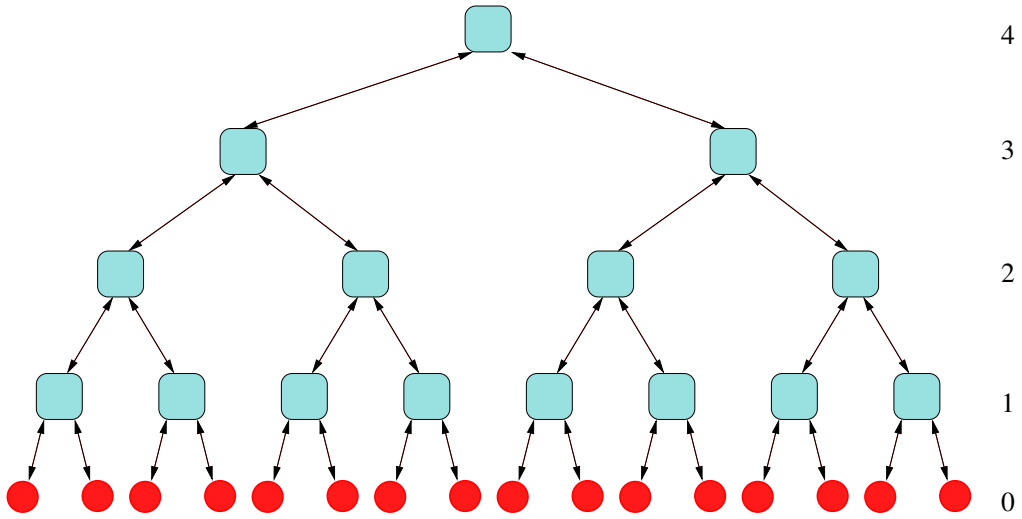


2–ary 4–cube



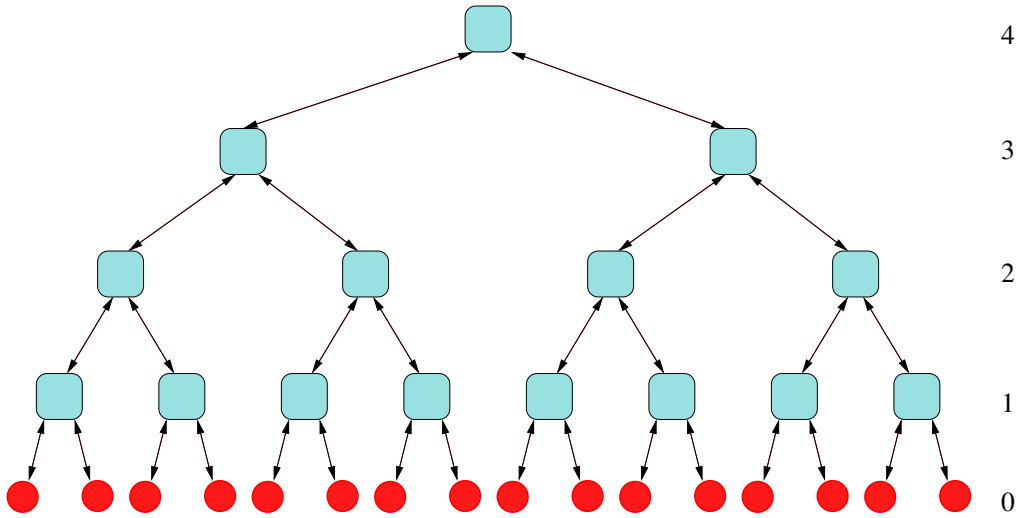
2–ary 5–cube

# Binary Trees



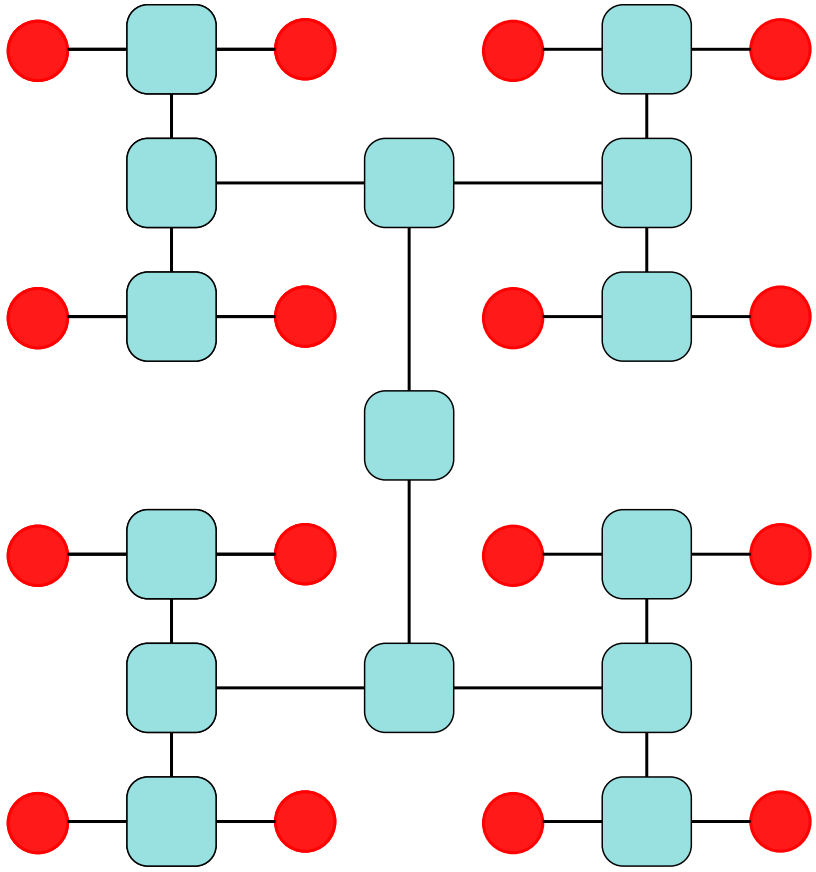

| | | |
|---|---|---|
| number of nodes $N$ | $=$ | $2^d$ |
| number of switches | $=$ | $2^d - 1$ |
| switch degree | $=$ | $3$ |
| diameter | $=$ | $2d$ |
| distance | $\sim$ | $d + 2$ |
| network cost | $=$ | $O(N)$ |
| total bandwidth | $=$ | $2 \cdot 2(N-1)b$ |
| bisection bandwidth | $=$ | $2b$ |

# $k$-ary Trees




| | | |
|---|---|---|
| number of nodes $N$ | $=$ | $k^d$ |
| number of switches | $\sim$ | $k^d$ |
| switch degree | $=$ | $k+1$ |
| diameter | $=$ | $2d$ |
| distance | $\sim$ | $d+2$ |
| network cost | $=$ | $O(N)$ |
| total bandwidth | $=$ | $2 \cdot 2(N-1)b$ |
| bisection bandwidth | $=$ | $kb$ |

# Binary Tree Projection



- Efficient and regular 2-layout;
- Longest wires in resource width:

$$lW = 2^{\lfloor \frac{d-1}{2} \rfloor - 1}$$

| $d$ | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|
| $N$ | 4 | 8 | 16 | 32 | 64 | 128 | 256 | 512 | 1024 |
| $lW$ | 0 | 1 | 1 | 2 | 2 | 4 | 4 | 8 | 8 |

# $k$-ary $n$-Cubes versus $k$-ary Trees

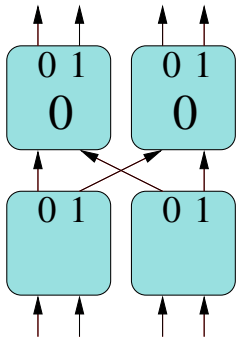
$k$-ary $n$-cubes:

| | | |
|---|---|---|
| number of nodes $N$ | $=$ | $k^d$ |
| switch degree | $=$ | $d+2$ |
| diameter | $=$ | $d(k-1)$ |
| distance | $\sim$ | $d\frac{1}{2}(k-1)$ |
| network cost | $=$ | $O(N)$ |
| total bandwidth | $=$ | $2Nb$ |
| bisection bandwidth | $=$ | $2k^{(d-1)}b$ |

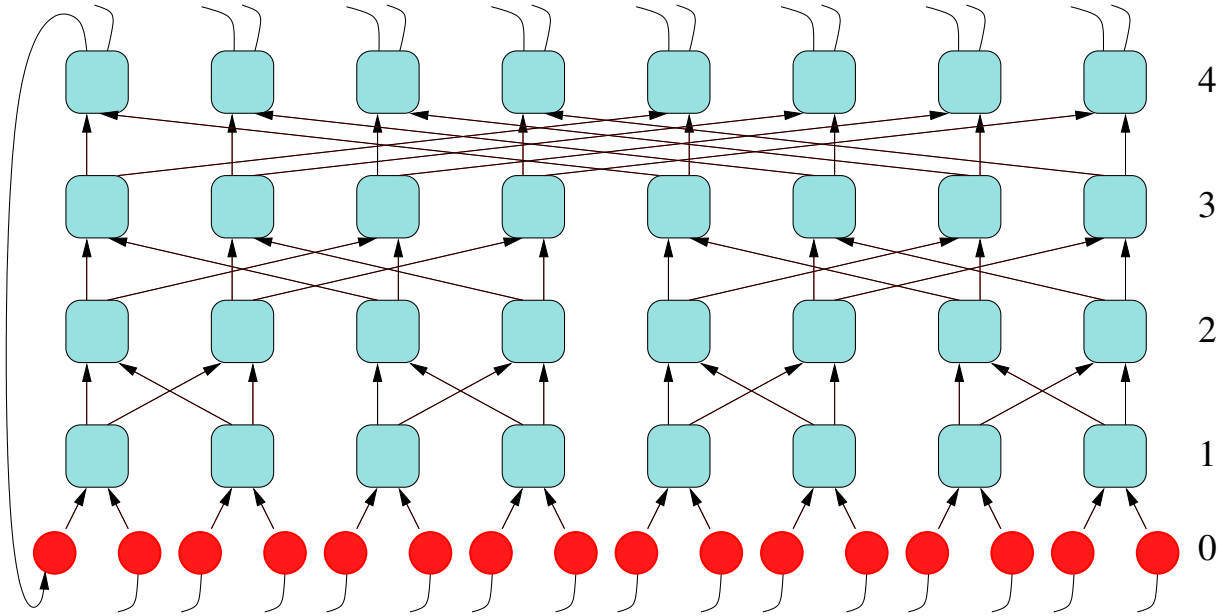
$k$-ary trees:

| | | |
|---|---|---|
| number of nodes $N$ | $=$ | $k^d$ |
| number of switches | $\sim$ | $k^d$ |
| switch degree | $=$ | $k+1$ |
| diameter | $=$ | $2d$ |
| distance | $\sim$ | $d+2$ |
| network cost | $=$ | $O(N)$ |
| total bandwidth | $=$ | $2 \cdot 2(N-1)b$ |
| bisection bandwidth | $=$ | $kb$ |

# Butterflies

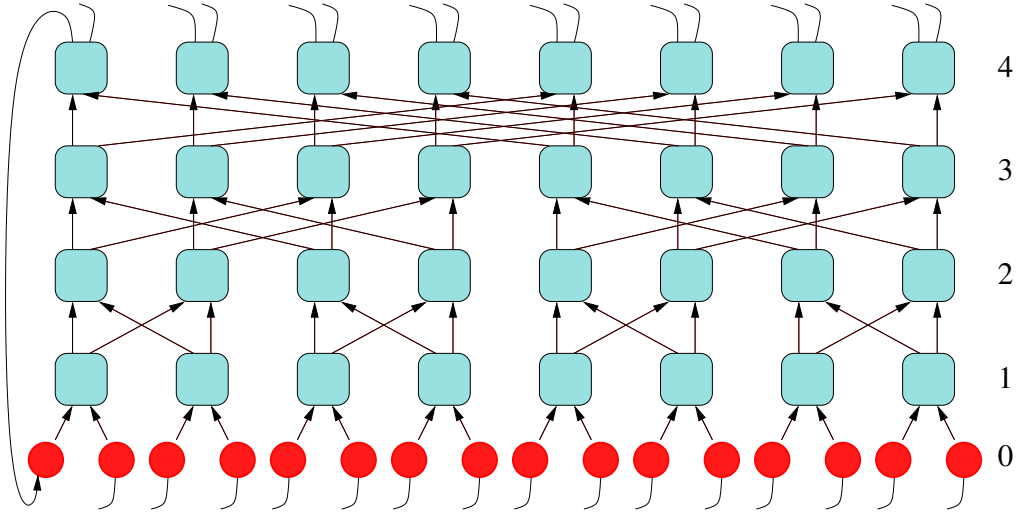



Butterfly building block




16 node butterfly

# Butterfly Characteristics




| | | |
|---|---|---|
| number of nodes $N$ | $=$ | $2^d$ |
| number of switches | $=$ | $2^{d-1}d$ |
| switch degree | $=$ | $2$ |
| diameter | $=$ | $d+1$ |
| distance | $=$ | $d+1$ |
| network cost | $=$ | $O(Nd)$ |
| total bandwidth | $=$ | $2^d db$ |
| bisection bandwidth | $=$ | $\frac{N}{2}b$ |

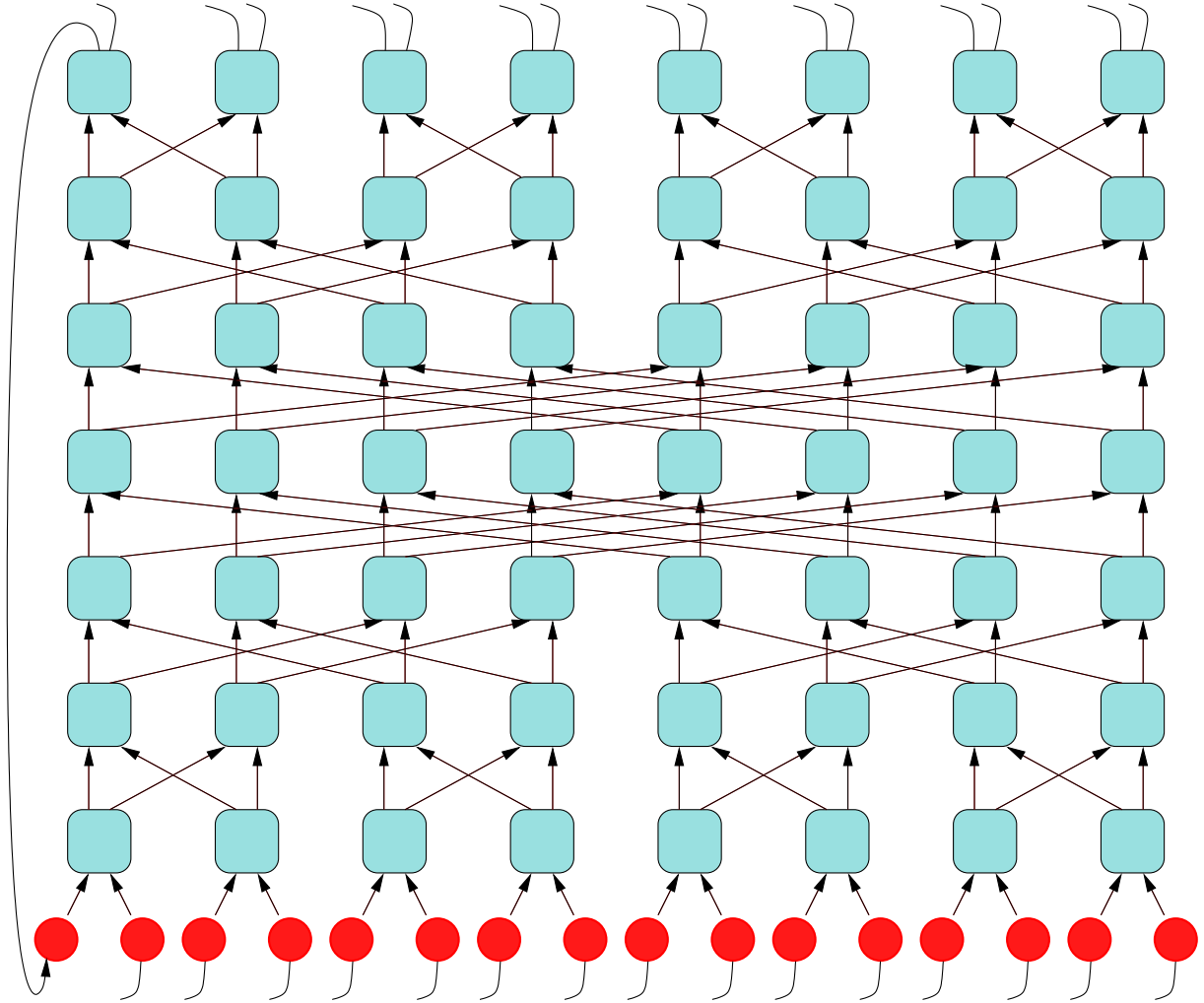
# $k$-ary $n$-Cubes versus $k$-ary Trees vs Butterflies

| | $k$-ary $n$-cubes | binary tree | butterfly |
|---|---|---|---|
| cost per node | $O(N)$ | $O(N)$ | $O(N \log N)$ |
| distance | $\frac{1}{2}\sqrt[d]{N} \log N$ | $2 \log N$ | $\log N$ |
| links per node | $2$ | $2$ | $\log N$ |
| bisection | $2N^{\frac{d-1}{d}}$ | $1$ | $\frac{1}{2}N$ |
| frequency limit of random traffic | $1/(\sqrt[d]{\frac{N}{2}})$ | $1/N$ | $1/2$ |

# Problems with Butterflies

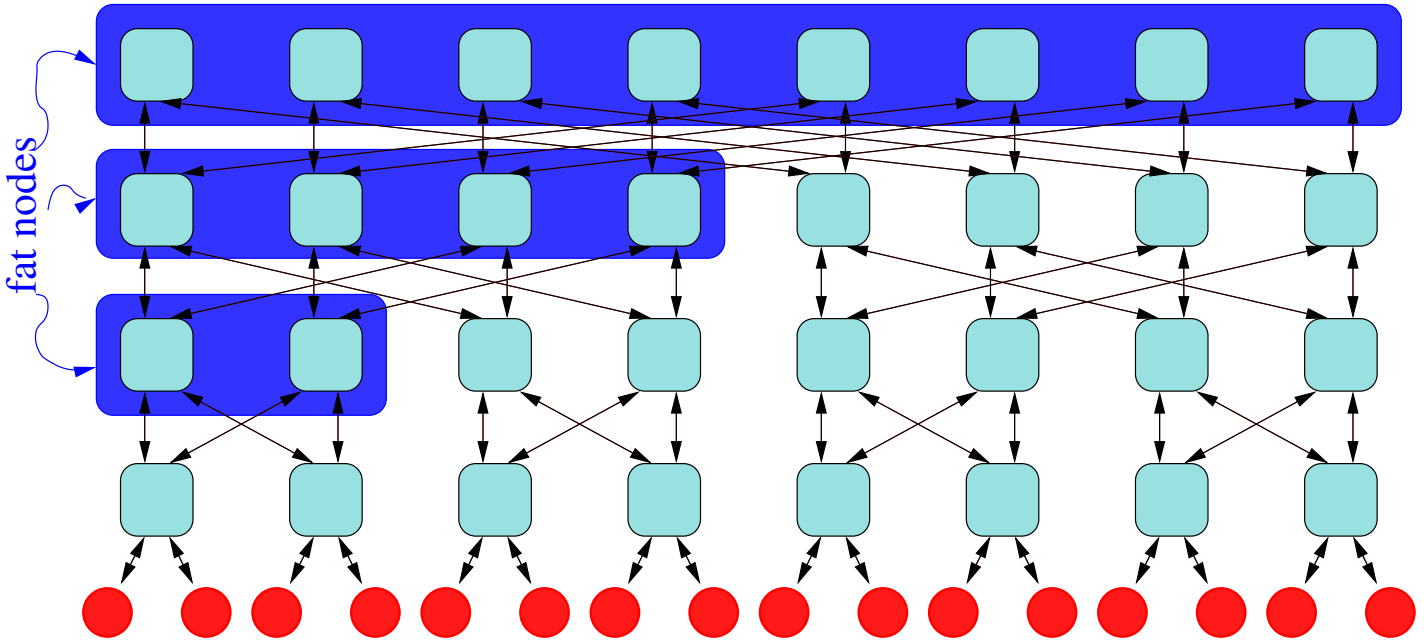
- Cost of the network
  - ⋆ $O(N \log N)$
  - ⋆ 2-d layout is more difficult than for binary trees
  - ⋆ Number of long wires grows faster than for trees.
- For each source-destination pair there is only one route.
- Each route blocks many other routes.

# Benes Networks



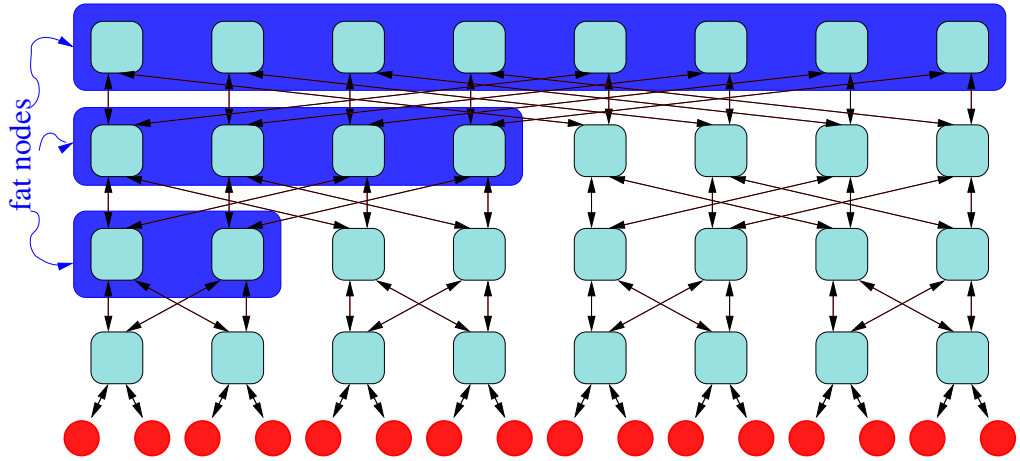
- Many routes;
- Costly to compute non-blocking routes;
- High probability for non-blocking route by randomly selecting an intermediate node [Leighton, 1992];

# Fat Trees




16–node 2–ary fat–tree

# $k$-ary $n$-dimensional Fat Tree Characteristics




16–node 2–ary fat–tree

| | | |
|---|---|---|
| number of nodes $N$ | $=$ | $k^d$ |
| number of switches | $=$ | $k^{d-1}d$ |
| switch degree | $=$ | $2k$ |
| diameter | $=$ | $2d$ |
| distance | $\sim$ | $d$ |
| network cost | $=$ | $O(Nd)$ |
| total bandwidth | $=$ | $2k^d db$ |
| bisection bandwidth | $=$ | $2k^{d-1}b$ |

# $k$-ary $n$-Cubes versus $k$-ary $d$-dimensional Fat Trees

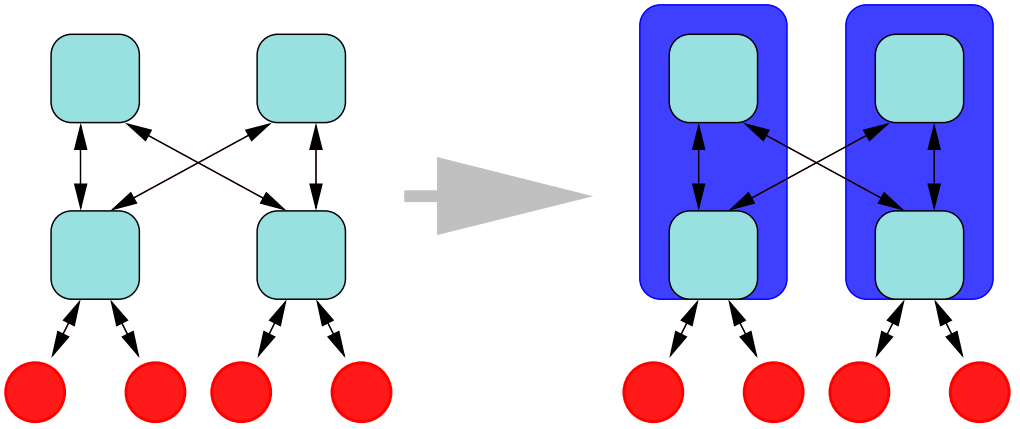
$k$-ary $n$-cubes:

| | | |
|---|---|---|
| number of nodes $N$ | $=$ | $k^d$ |
| switch degree | $=$ | $d$ |
| diameter | $=$ | $d(k-1)$ |
| distance | $\sim$ | $d\frac{1}{2}(k-1)$ |
| network cost | $=$ | $O(N)$ |
| total bandwidth | $=$ | $2Nb$ |
| bisection bandwidth | $=$ | $2k^{(d-1)}b$ |

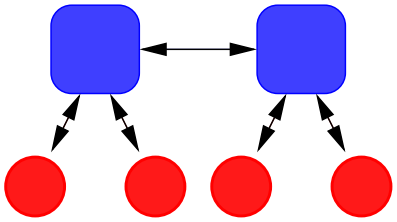
$k$-ary $n$-dimensional fat trees:

| | | |
|---|---|---|
| number of nodes $N$ | $=$ | $k^d$ |
| number of switches | $=$ | $k^{d-1}d$ |
| switch degree | $=$ | $2k$ |
| diameter | $=$ | $2d$ |
| distance | $\sim$ | $d$ |
| network cost | $=$ | $O(Nd)$ |
| total bandwidth | $=$ | $2k^d db$ |
| bisection bandwidth | $=$ | $2k^{d-1}b$ |

# Relation between Fat Tree and Hypercube

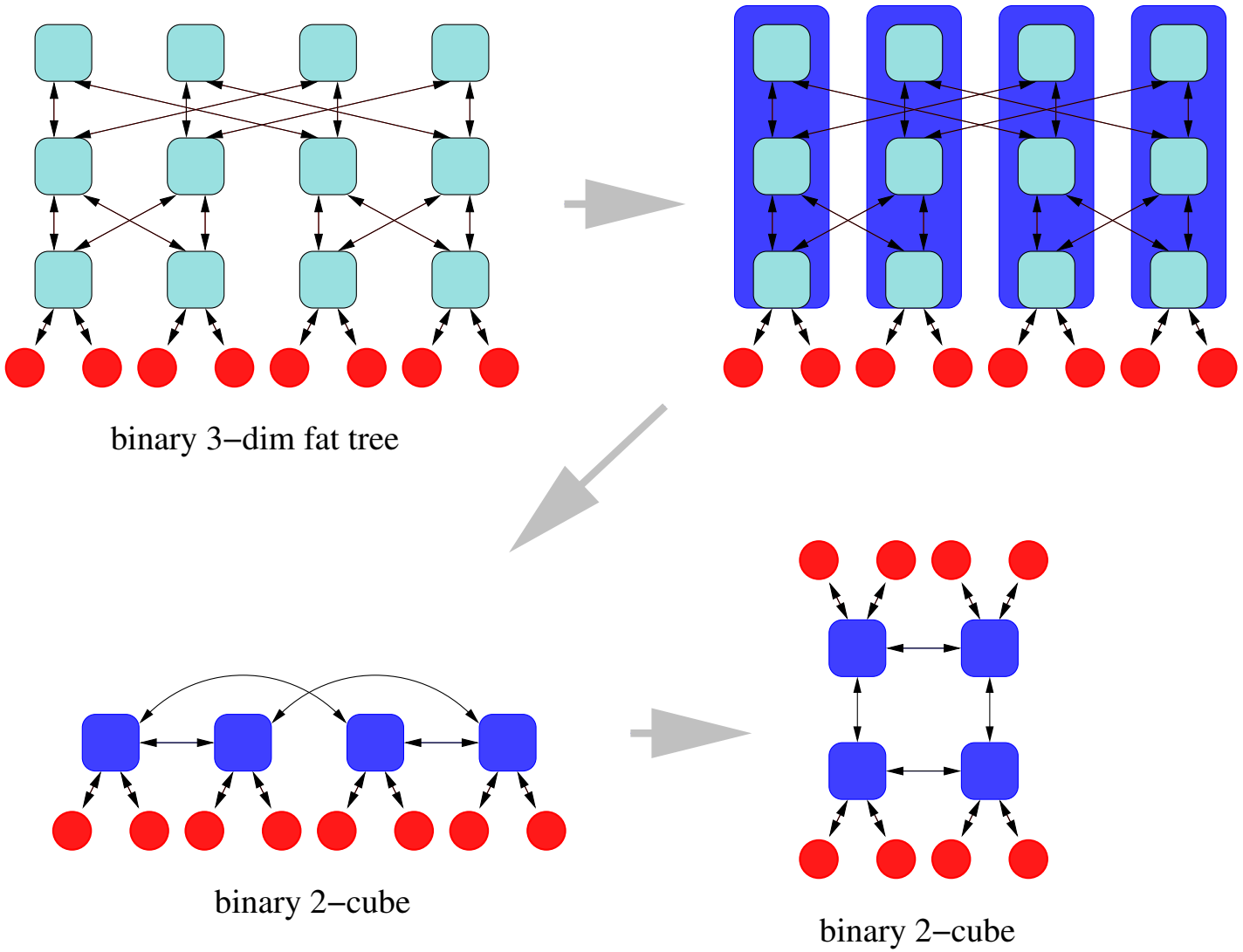


binary 2–dim fat tree

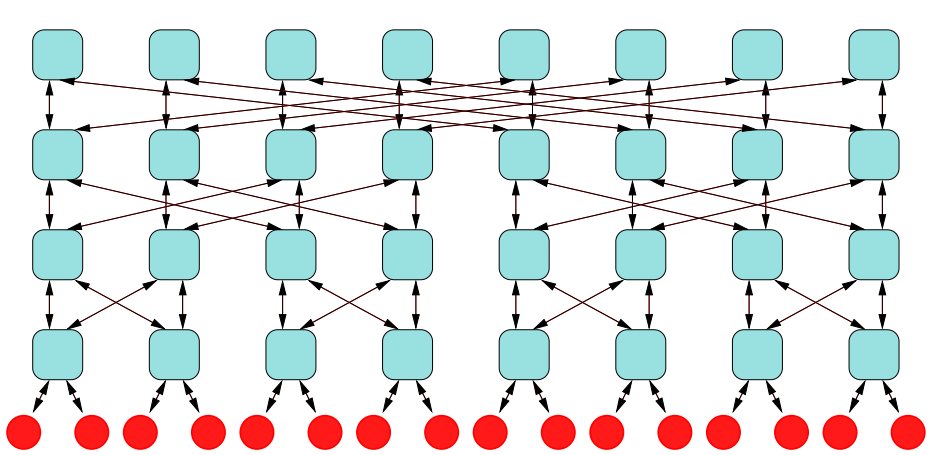


binary 1–cube

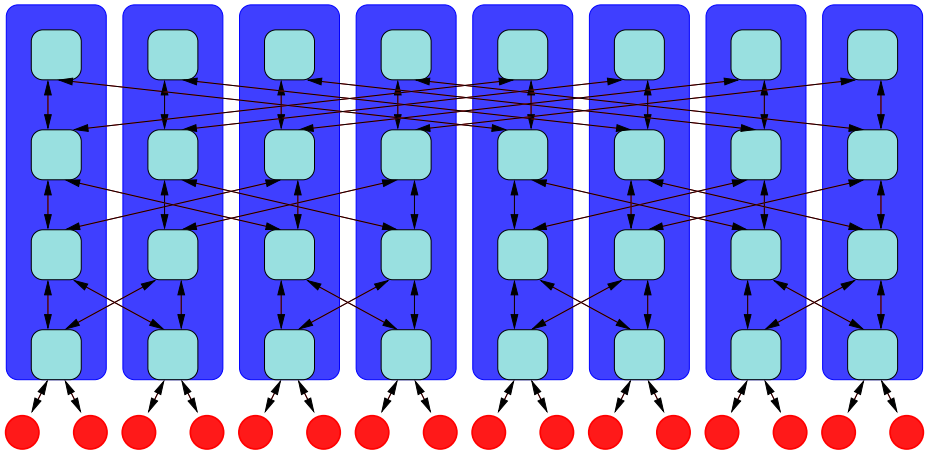
# Relation between Fat Tree and Hypercube - cont'd

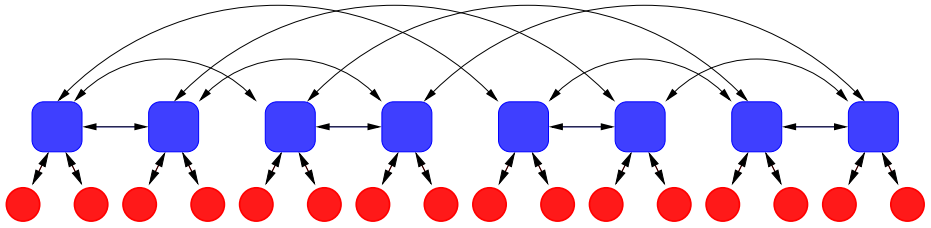


# Relation between Fat Tree and Hypercube - cont'd

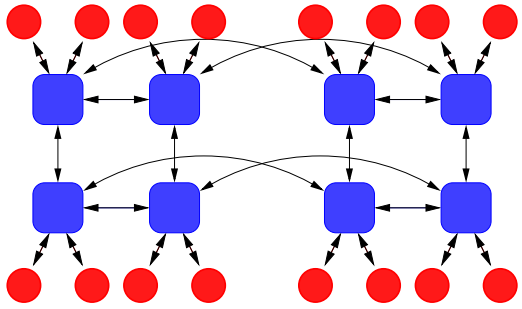


binary 4–dim fat tree





binary 3–cube



binary 3–cube

# Topologies of Parallel Computers

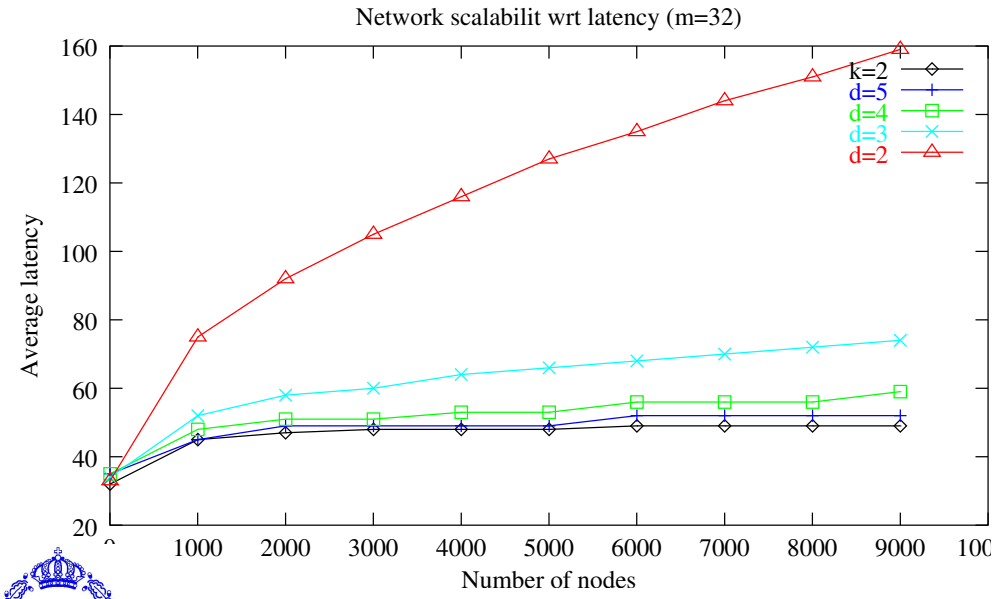
| Machine | Topology | Cycle Time [ns] | Channel width [bits] | Routing delay [cycles] | Flit size [bits] |
|---|---|---|---|---|---|
| nCUBE/2 | Hypercube | 25 | 1 | 40 | 32 |
| TMC CM-5 | Fat tree | 25 | 4 | 10 | 4 |
| IBM SP-2 | Banyan | 25 | 8 | 5 | 16 |
| Intel Paragon | 2D Mesh | 11.5 | 16 | 2 | 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 |

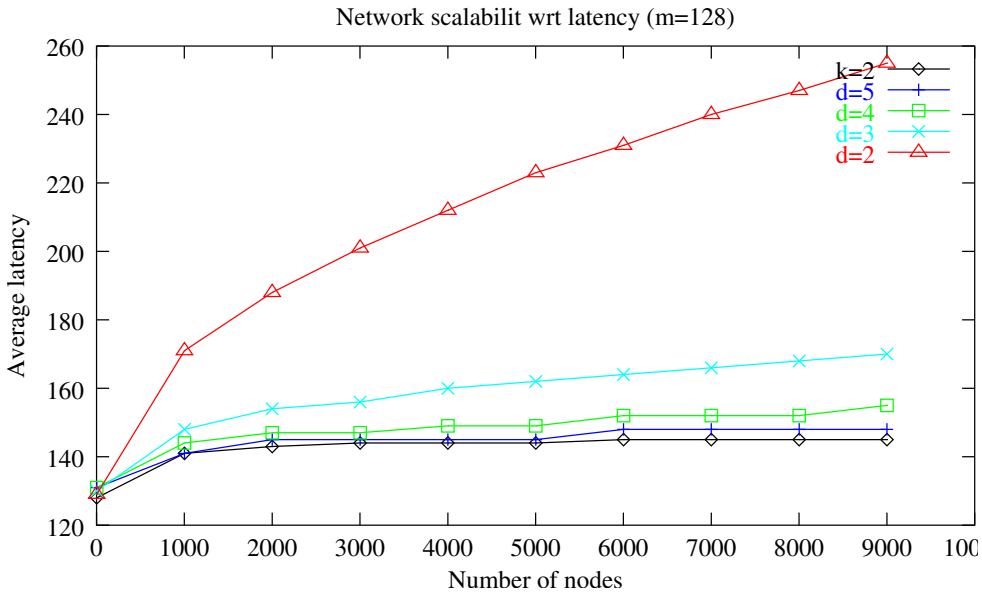
# Trade-offs in Topology Design for the $k$-ary $n$-Cube

- Unloaded Latency
- Latency under Load

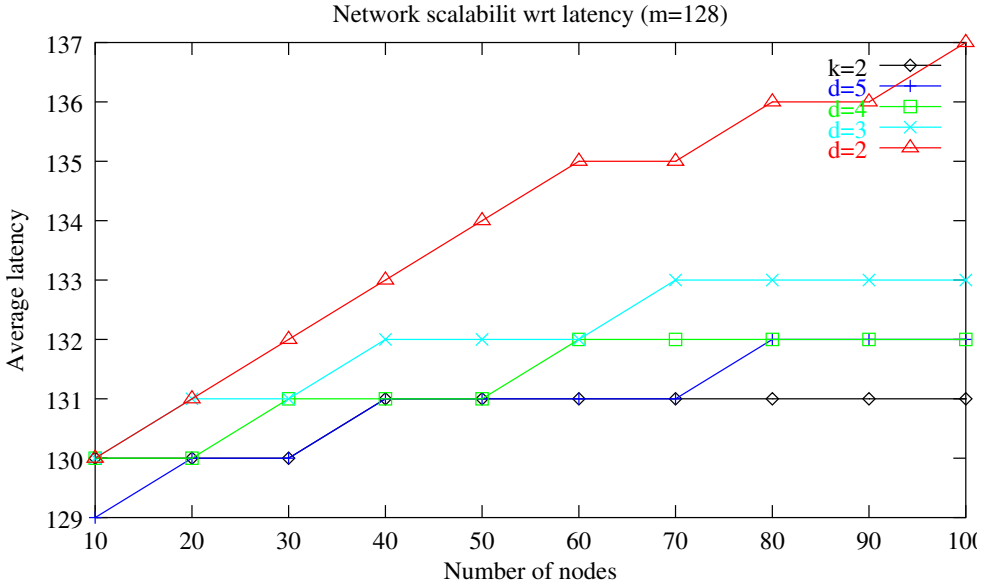
# Network Scaling for Unloaded Latency

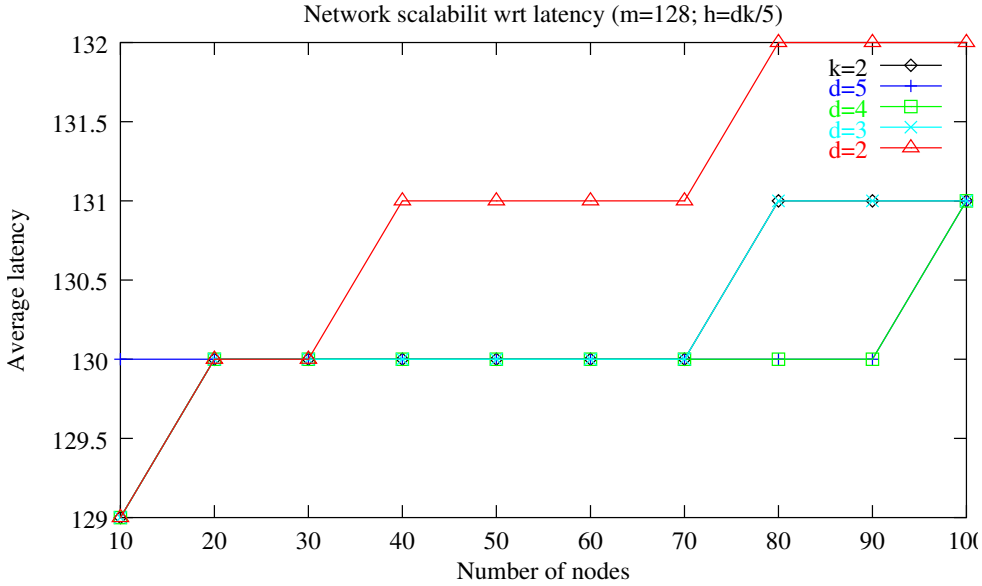
$$\begin{aligned}
\text{Latency}(n) &= \text{Admission} + \text{ChannelOccupancy} \\
&\quad + \text{RoutingDelay} + \text{ContentionDelay} \\
\text{RoutingDelay } T_{ct}(n,h) &= \frac{n}{b} + h\Delta \\
\text{RoutingDistance } h &= \frac{1}{2}d(k-1) = \frac{1}{2}(k-1)\log_k N = \frac{1}{2}(d\sqrt[d]{N} - 1)
\end{aligned}$$

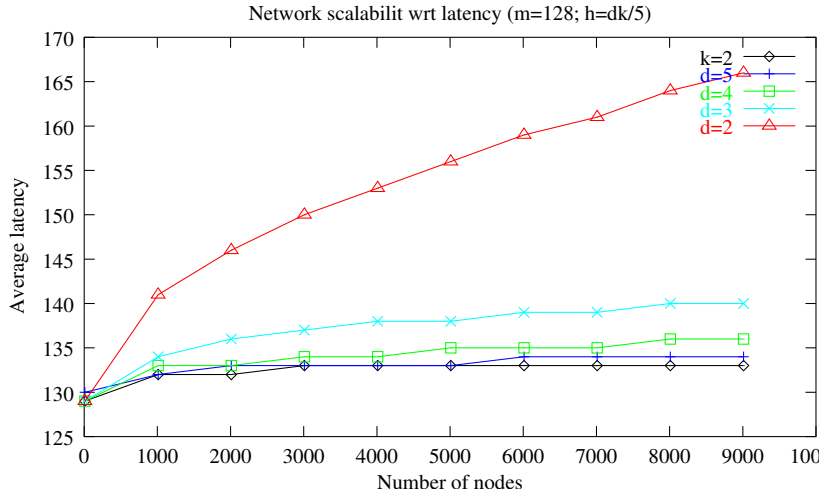


# Unloaded Latency for Small Networks and Local Traffic

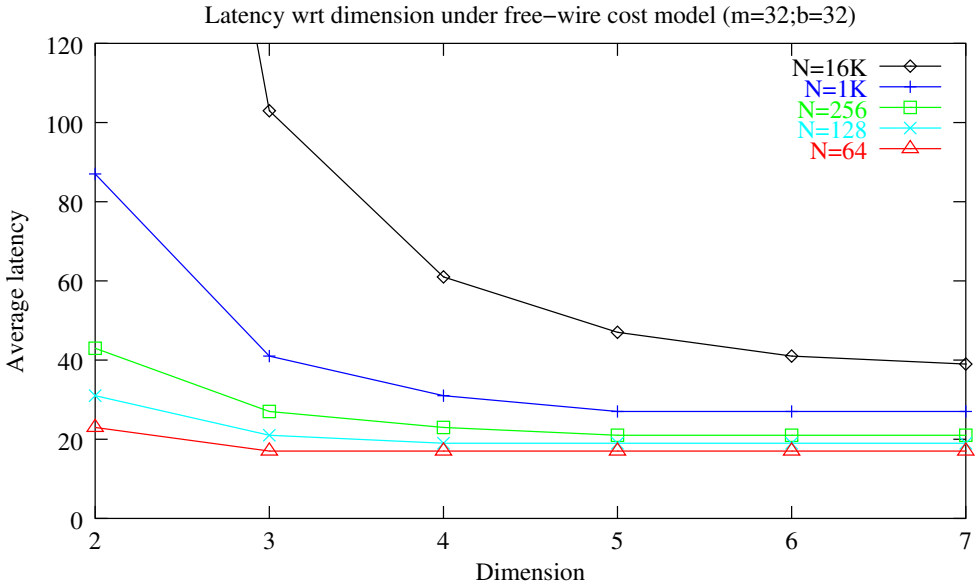


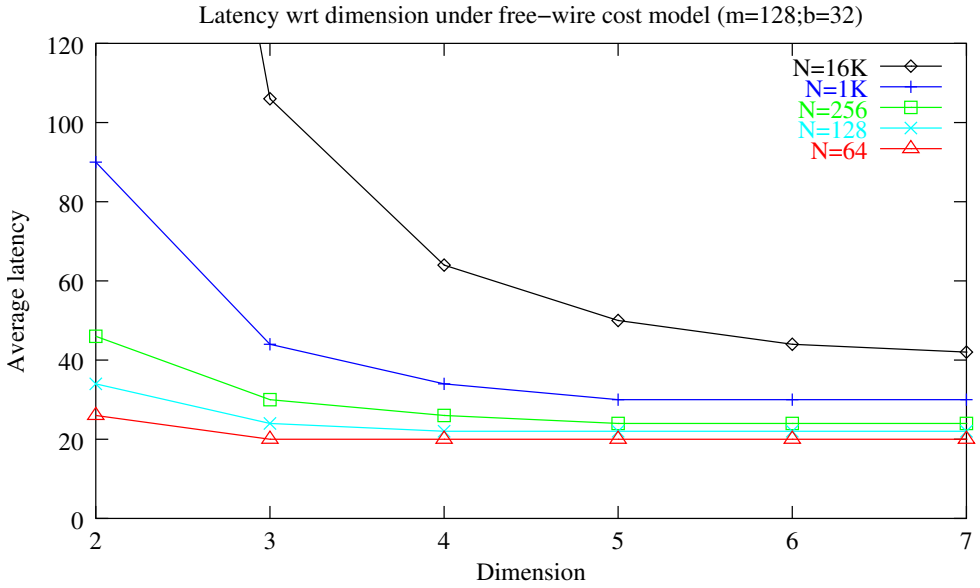



# Unloaded Latency under a Free-Wire Cost Model

**Free-wire** cost model: Wires are free and can be added without penalty.

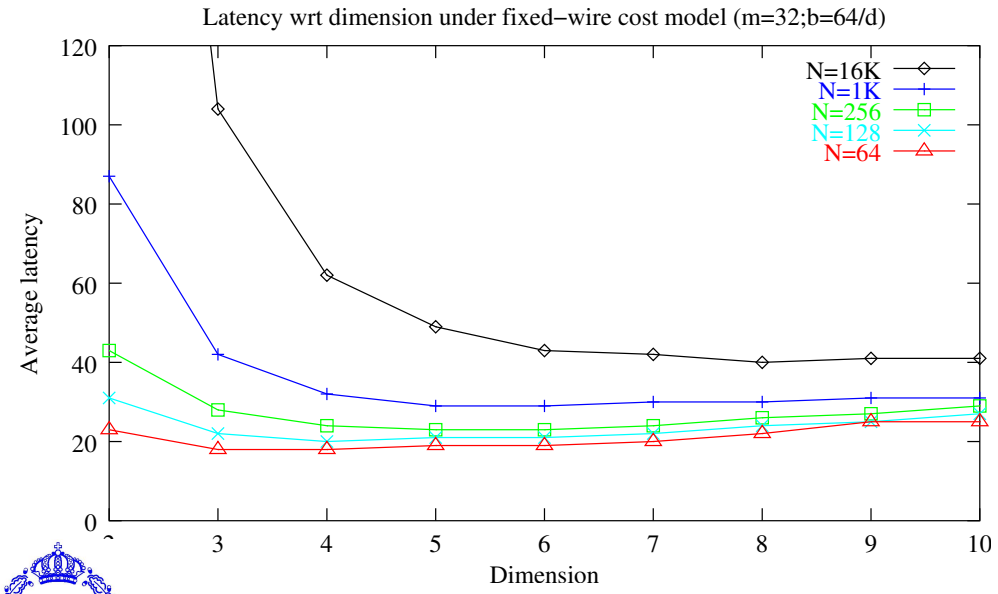


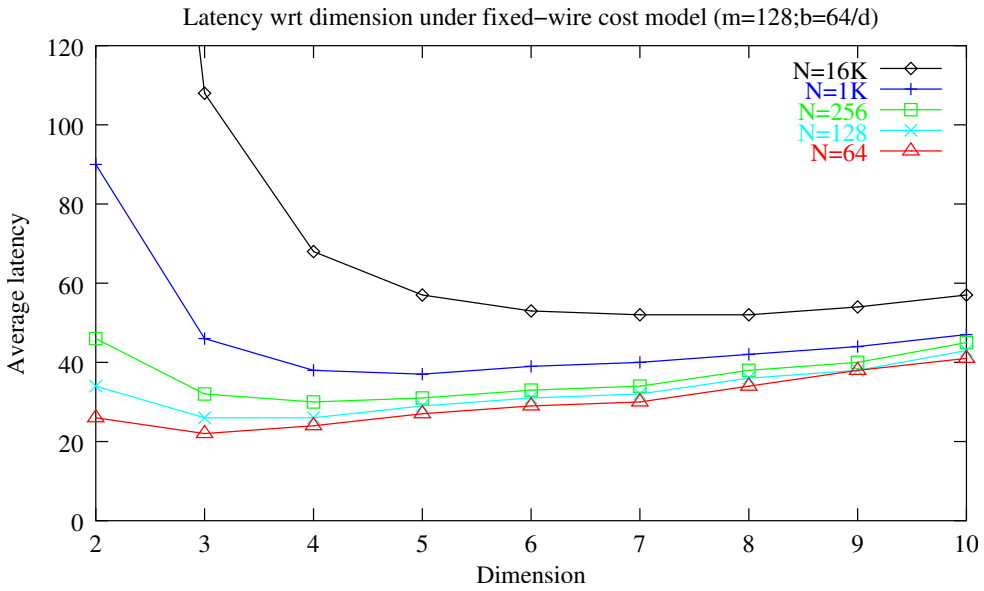



# Unloaded Latency under a Fixed-Wire Cost Models

**Fixed-wire** cost model: The number of wires is constant per node:
128 wires per node: $w(d) = \lfloor \frac{64}{d} \rfloor$.

| $d$ | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|
| $w(d)$ | 32 | 21 | 16 | 12 | 10 | 9 | 8 | 7 | 6 |

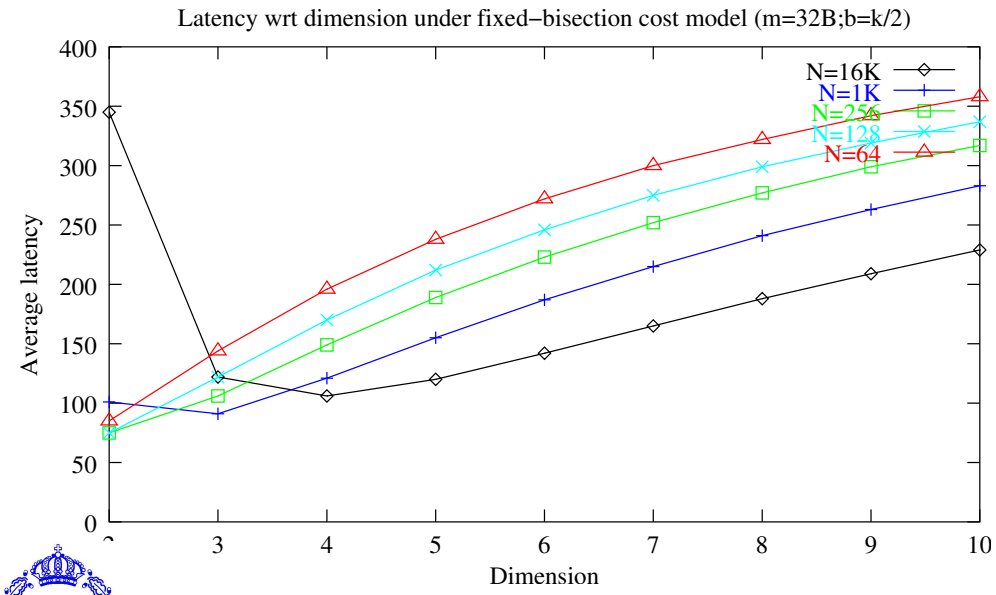


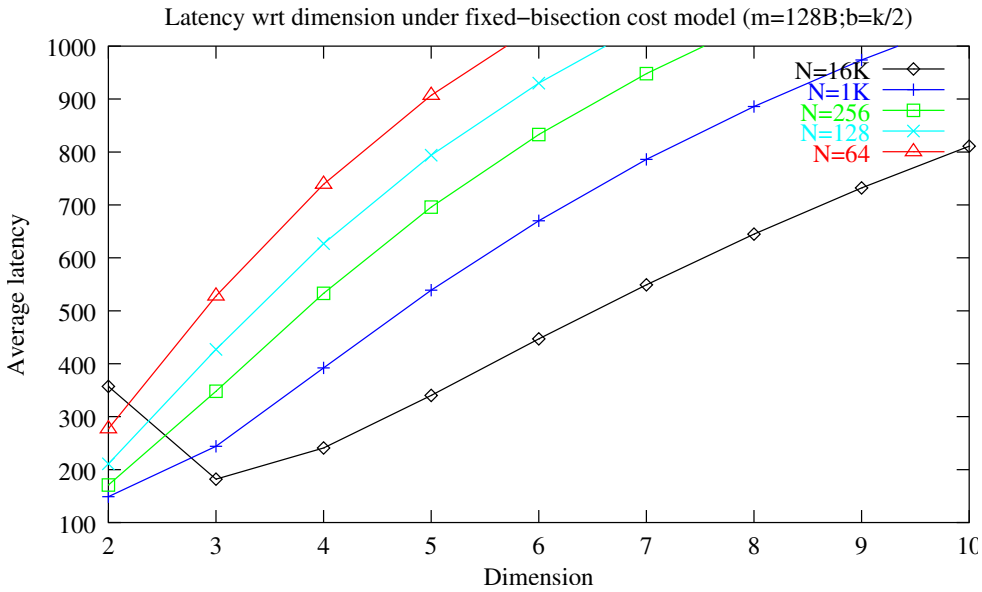



# Unloaded Latency under a Fixed-Bisection Cost Models

**Fixed-bisection** cost model: The number of wires across the bisection is constant:
bisection = 1024 wires: $w(d) = \frac{k}{2} = \frac{\sqrt[d]{N}}{2}$.
Example: N=1024:

| $d$ | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|
| $w(d)$ | 512 | 16 | 5 | 3 | 2 | 2 | 1 | 1 | 1 |

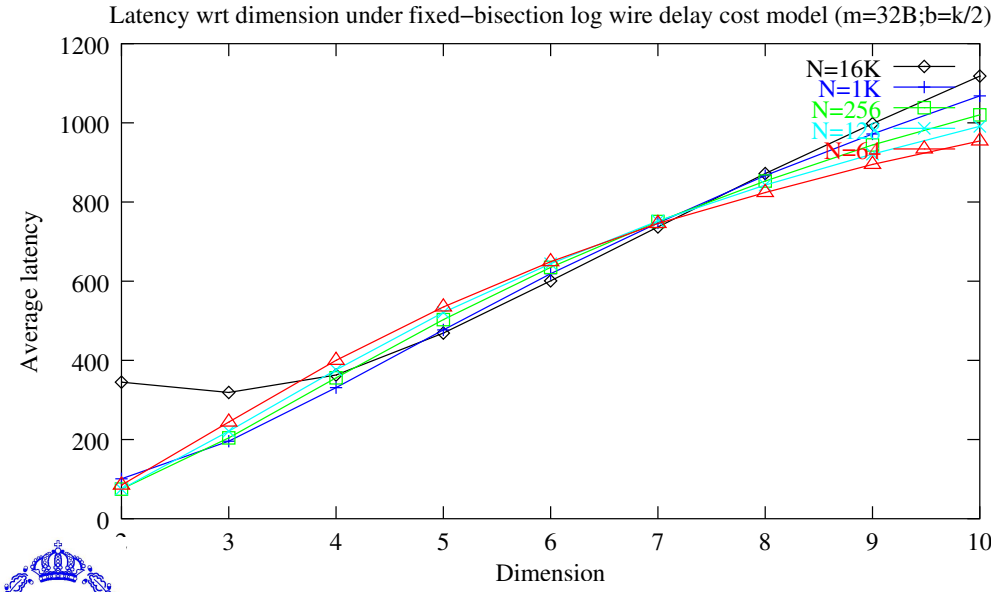


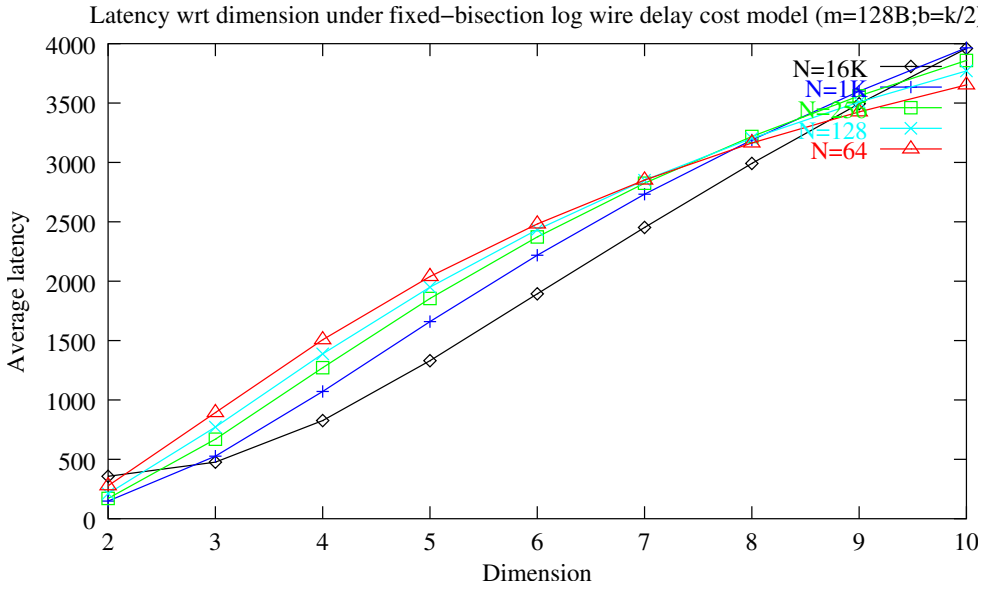



# Unloaded Latency under a Logarithmic Wire Delay Cost Models

**Fixed-bisection Logarithmic Wire Delay** cost model: The number of wires across the bisection is constant and the delay on wires increases logarithmically with the length [Dally, 1990]:
Length of long wires: $l = k^{\frac{n}{2}-1}$

$$T_c \propto 1 + \log l = 1 + (\frac{d}{2} - 1) \log k$$

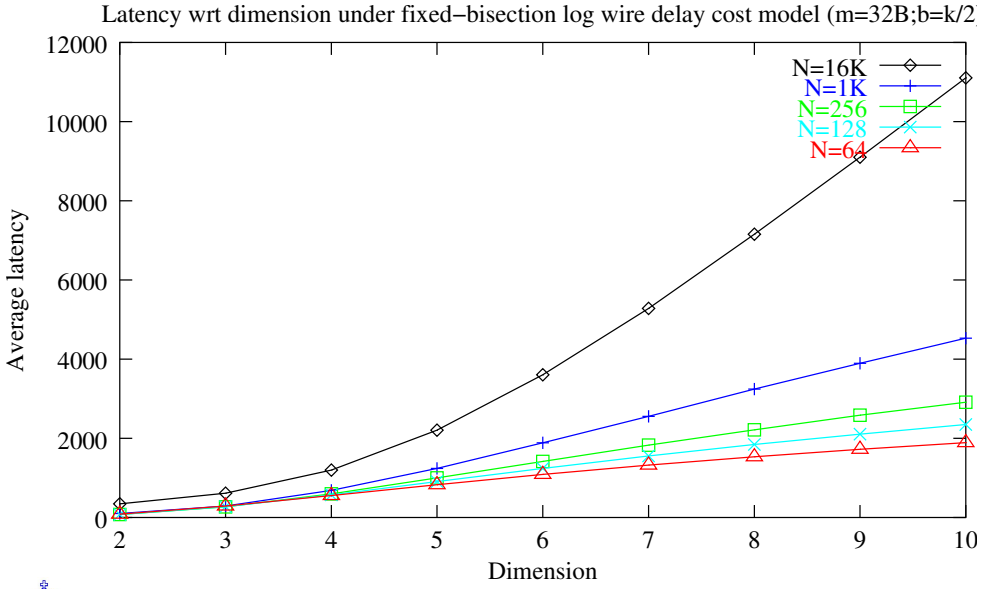


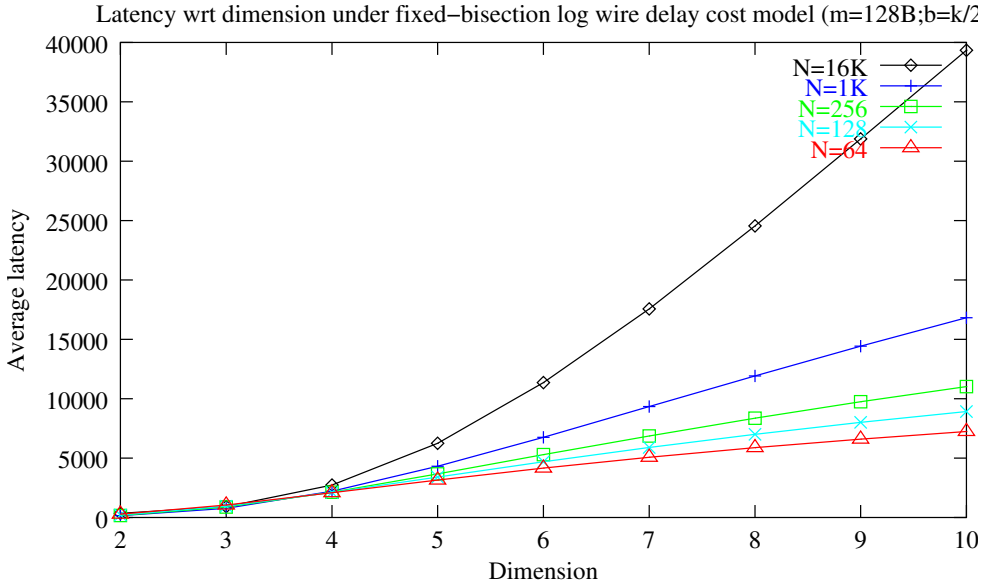



# Unloaded Latency under a Linear Wire Delay Cost Models

**Fixed-bisection Linear Wire Delay** cost model: The number of wires across the bisection is constant and the delay on wires increases linearly with the length [Dally, 1990]:
Length of long wires: $l = k^{\frac{n}{2}-1}$

$$T_c \propto l = k^{\frac{d}{2}-1}$$



# Latency under Load

Assumptions [Agarwal, 1991]:

- $k$-ary $n$-cubes
- random traffic
- dimension-order cut-through routing
- unbounded internal buffers (to ignore flow control and deadlock issues)

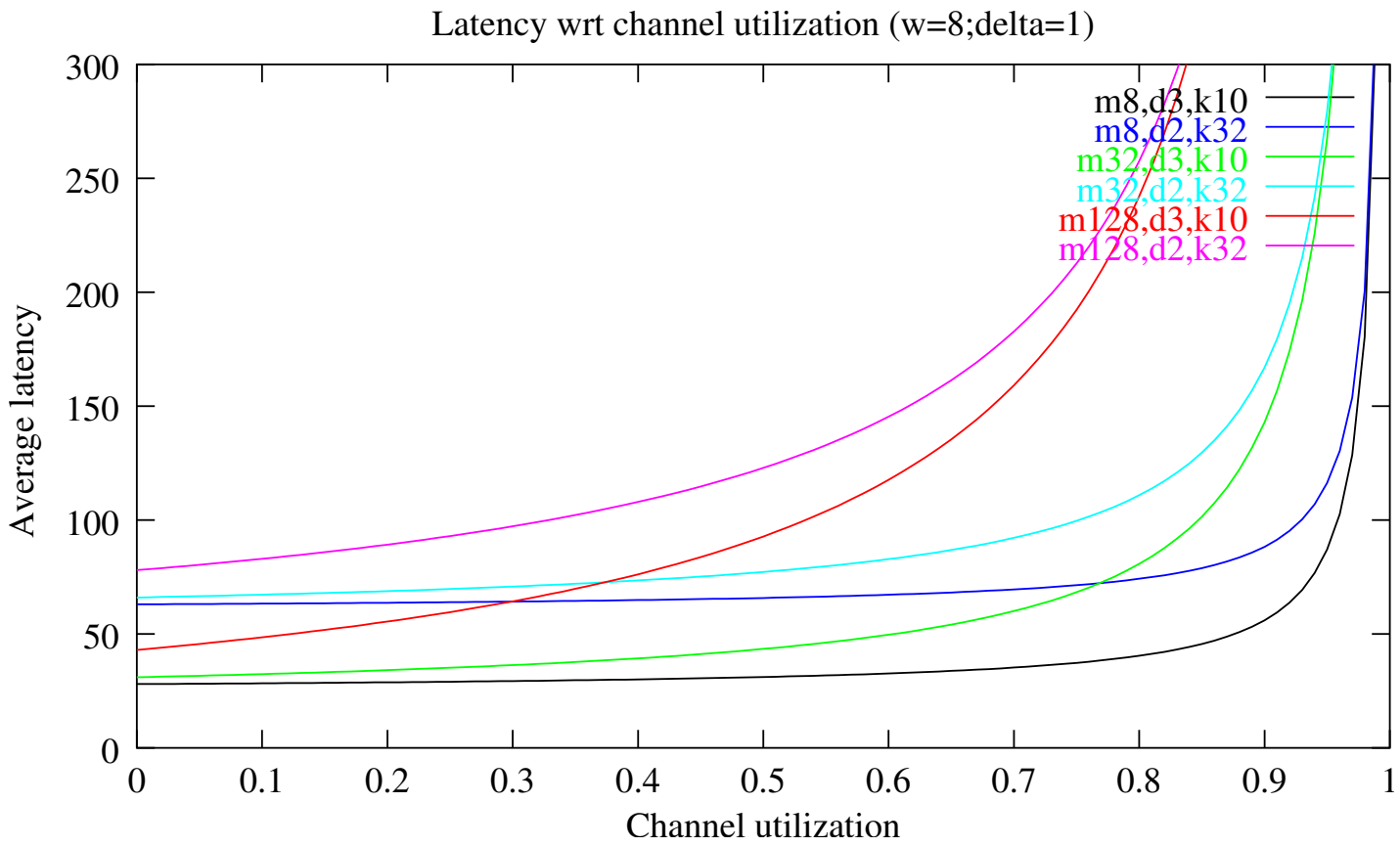
# Latency under Load - cont'd

$$\text{Latency}(n) = \text{Admission} + \text{ChannelOccupancy} + \text{RoutingDelay} + \text{ContentionDelay}$$

$$\begin{aligned}
T(m,k,d,w,\rho) &= \text{RoutingDelay} + \text{ContentionDelay} \\
T(m,k,d,w,\rho) &= \frac{m}{w} + dh_k(\Delta + W(m,k,d,w,\rho)) \\
W(m,k,d,w,\rho) &= \frac{m}{w} \cdot \frac{\rho}{(1-\rho)} \cdot \frac{h_k - 1}{h_k^2} \cdot \left(1 + \frac{1}{d}\right) \\
h &= \frac{1}{2} d(k-1)
\end{aligned}$$

$m$ ··· message size
$w$ ··· bitwidth of link
$\rho$ ··· aggregate channel utilization
$h_k$ ··· average distance in each dimension
$\Delta$ ··· switching time in cycles

# Latency vs Channel Load



# Routing

**Deterministic routing** The route is determined solely by source and destination locations.

**Arithmetic routing** The destination address of the incoming packet is compared with the address of the switch and the packet is routed accordingly. (relative or absolute addresses)

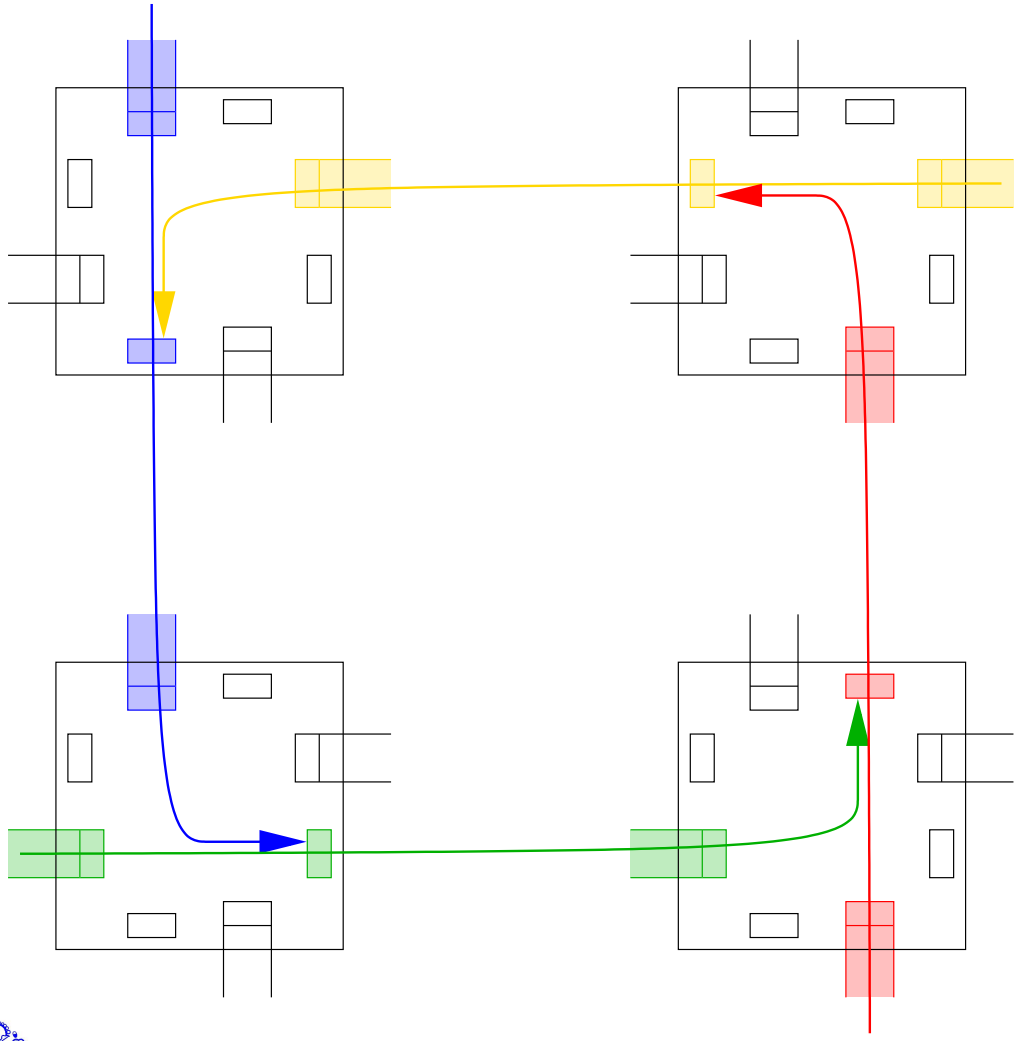
**Source based routing** The source determines the route and builds a header with one directive for each switch. The switches strip off the top directive.

**Table-driven routing** Switches have routing tables, which can be configured.

**Adaptive routing** The route can be adapted by the switches to balance the load.

**Minimal routing** allows only shortest paths while non-minimal routing allows even longer paths.

# Deadlock



**Deadlock** Two or several packets mutually block each other and wait for resources, which can never be free.

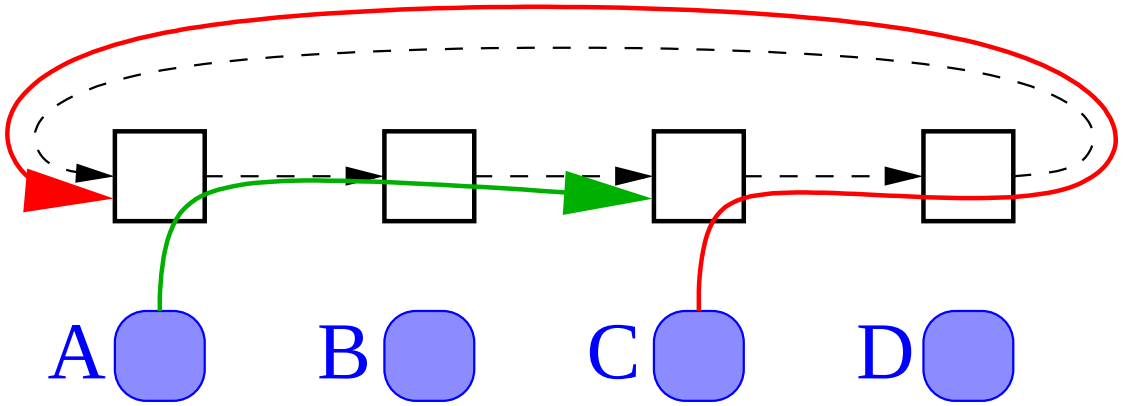
**Livelock** A packet keeps moving through the network but never reaches its destination.

**Starvation** A packet never gets a resource because it always looses the competition for that resource (fairness).

# Deadlock Situations

- Head-on deadlock;
- Nodes stop receiving packets;
- Contention for switch buffers can occur with store-and-forward, virtual-cut-through and wormhole routing. Wormhole routing is particularly sensible.
- Cannot occur in butterflies;
- Cannot occur in trees or fat trees if upward and downward channels are independent;
- Dimension order routing is deadlock free on $k$-ary $n$-arrays but not on tori with any $n \geq 1$.

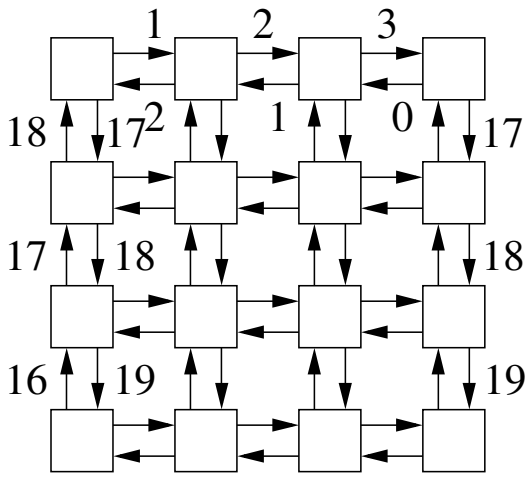
# Deadlock in a 1-dimensional Torus



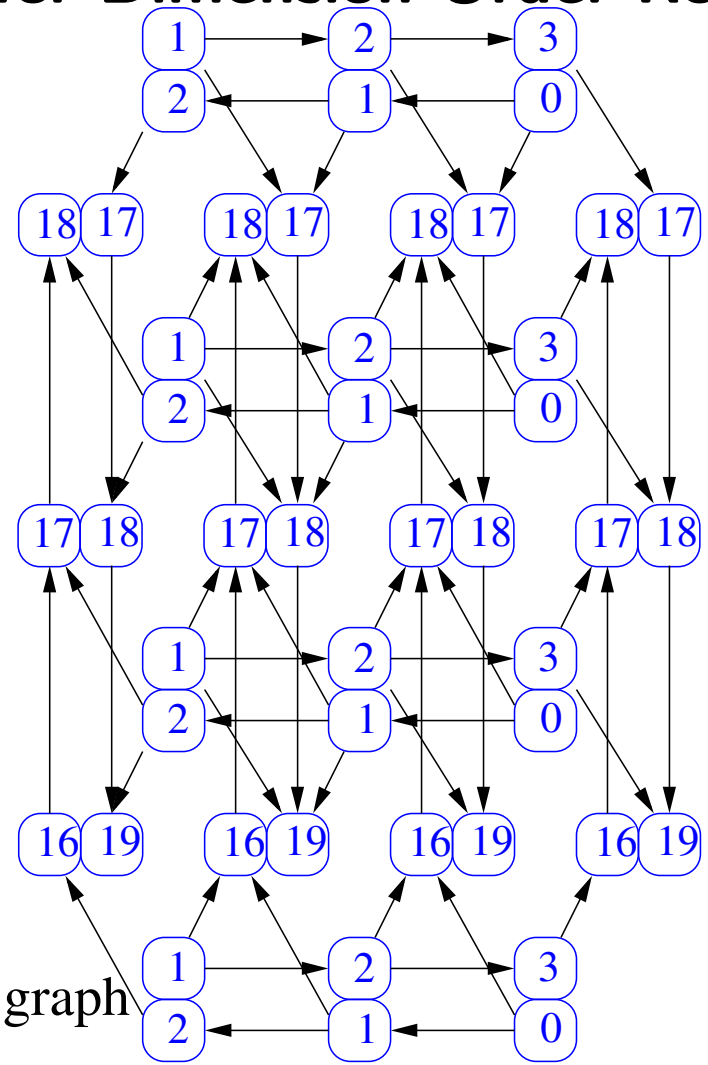

Message 1 from C−> B, 10 flits

Message 2 from A−> D, 10 flits

# Channel Dependence Graph for Dimension Order Routing




4–ary 2–array



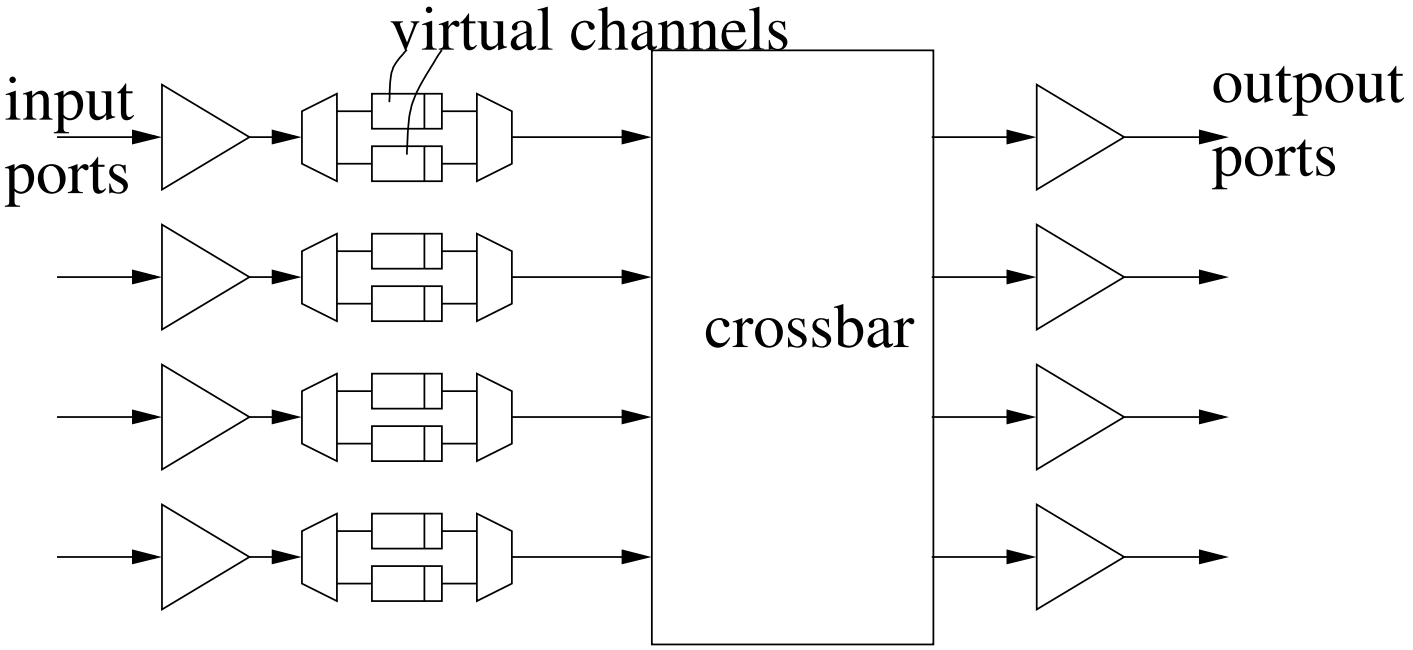

channel dependence graph

Routing is deadlock free if the channel dependence graph has no cycles.

# Deadlock-free Routing

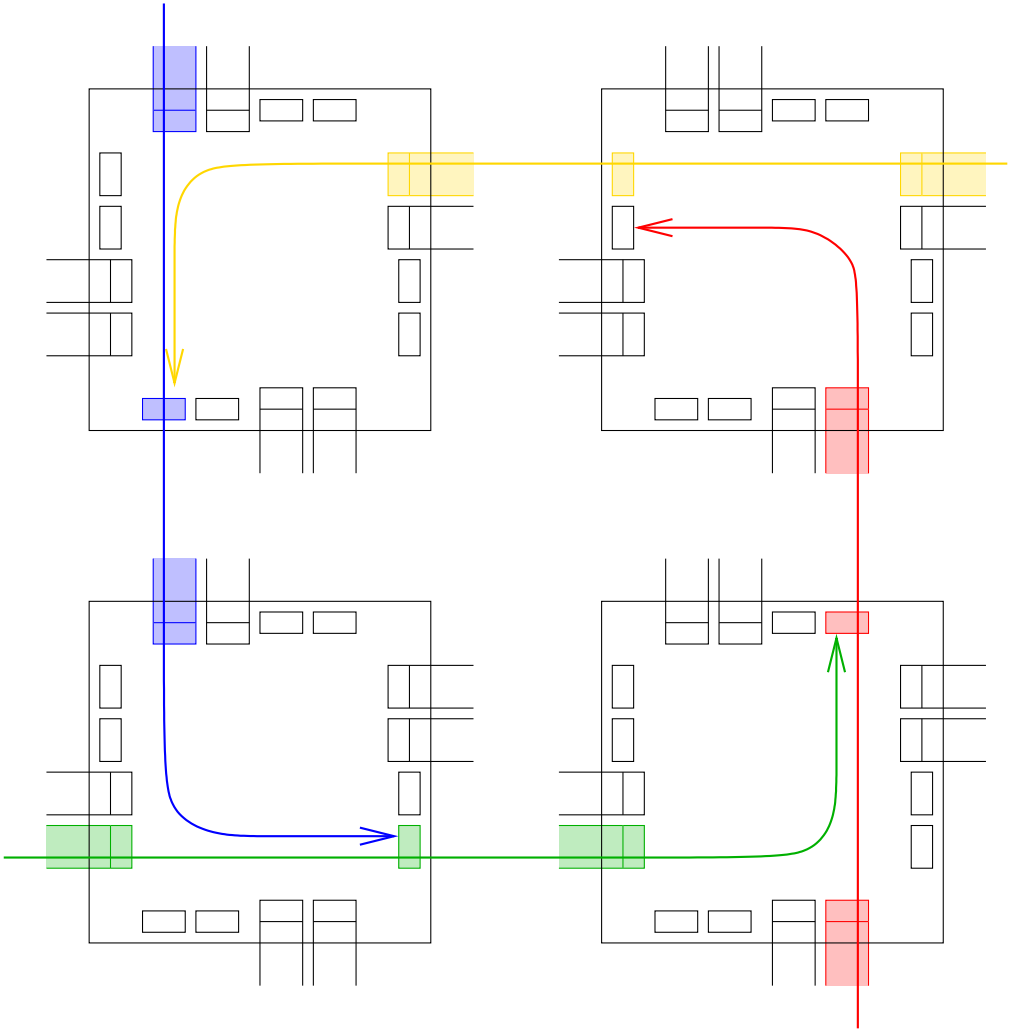
- Two main approaches:
  - ⋆ Restrict the legal routes;
  - ⋆ Restrict how resources are allocated;
- Number the channel cleverly
- Construct the channel dependence graph
- Prove that all legal routes follow a strictly increasing path in the channel dependence graph.

# Virtual Channels



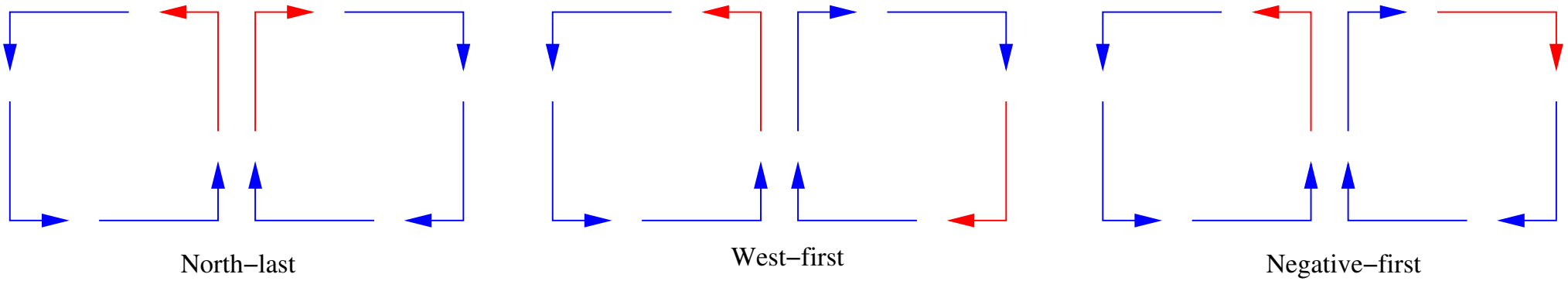

- Virtual channels can be used to break cycles in the dependence graph.
- E.g. all $n$-dimensional tori can be made deadlock free under dimension-order routing by assigning all wrap-around paths to a different virtual channel than other links.

# Virtual Channels and Deadlocks



# Turn-Model Routing

What are the minimal routing restrictions to make routing deadlock free?




- Three minimal routing restriction schemes:
  - ⋆ **North-last**
  - ⋆ **West-first**
  - ⋆ **Negative-first**
- Allow complex, non-minimal adaptive routes.
- Unidirectional $k$-ary $n$-cubes still need virtual channels.

# Adaptive Routing

- The switch makes routing decisions based on the load.
- Fully adaptive routing allows all shortest paths.
- Partial adaptive routing allows only a subset of the shortest path.
- Non-minimal adaptive routing allows also non-minimal paths.
- Hot-potato routing is non-minimal adaptive routing without packet buffering.

# Summary

- Communication Performance: bandwidth, unloaded latency, loaded latency
- Organizational Structure: NI, switch, link
- Topologies: wire space and delay domination favors low dimension topologies;
- Routing: deterministic vs source based vs adaptive routing; deadlock;

# Issues beyond the Scope of this Lecture

- Switch: Buffering; output scheduling; flow control;
- Flow control: Link level and end-to-end control;
- Power
- Clocking
- Faults and reliability
- Memory architecture and I/O
- Application specific communication patterns
- Services offered to applications; Quality of service

# NoC Research Projects

- Nostrum at KTH
- Æthereal at Philips Research
- Proteo at Tampere University of Technology
- SPIN at UPMC/LIP6 in Paris
- XPipes at Bologna U
- Octagon at ST and UC San Diego

# To Probe Further - Books and Classic Papers

[Agarwal, 1991] Agarwal, A. (1991). Limit on interconnection performance. *IEEE Transactions on Parallel and Distributed Systems*, 4(6):613–624.

[Culler et al., 1999] Culler, D. E., Singh, J. P., and Gupta, A. (1999). *Parallel Computer Architecture - A Hardware/Software Approach*. Morgan Kaufman Publishers.

[Dally, 1990] Dally, W. J. (1990). Performance analysis of k-ary n-cube interconnection networks. *IEEE Transactions on Computers*, 39(6):775–785.

[Duato et al., 1998] Duato, J., Yalamanchili, S., and Ni, L. (1998). *Interconnection Networks - An Engineering Approach*. Computer Society Press, Los Alamitos, California.

[Leighton, 1992] Leighton, F. T. (1992). *Introduction to Parallel Algorithms and Architectures*. Morgan Kaufmann, San Francisco.