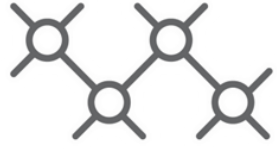
Network on Chip

March 2015

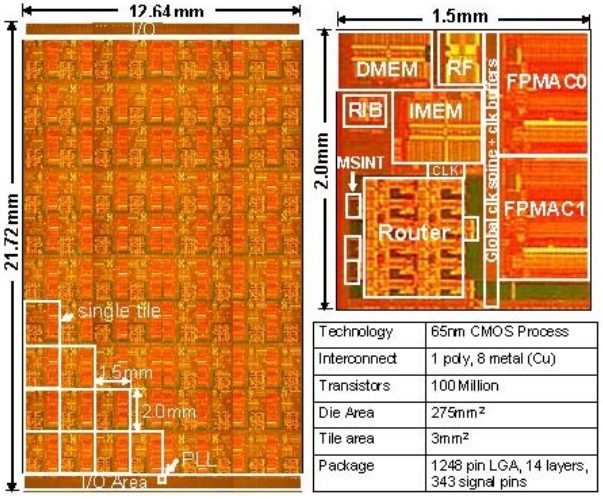
Axel Jantsch

TECHNISCHE
UNIVERSITÄT
WIEN
Vienna University of Technology

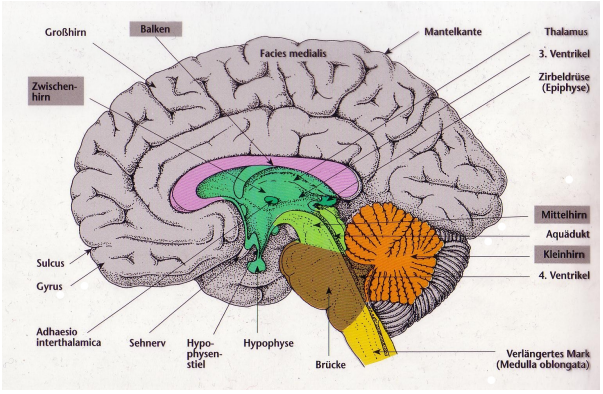


Institut für
Computertechnik
Institute of
Computer Technology

60 km fungus filaments

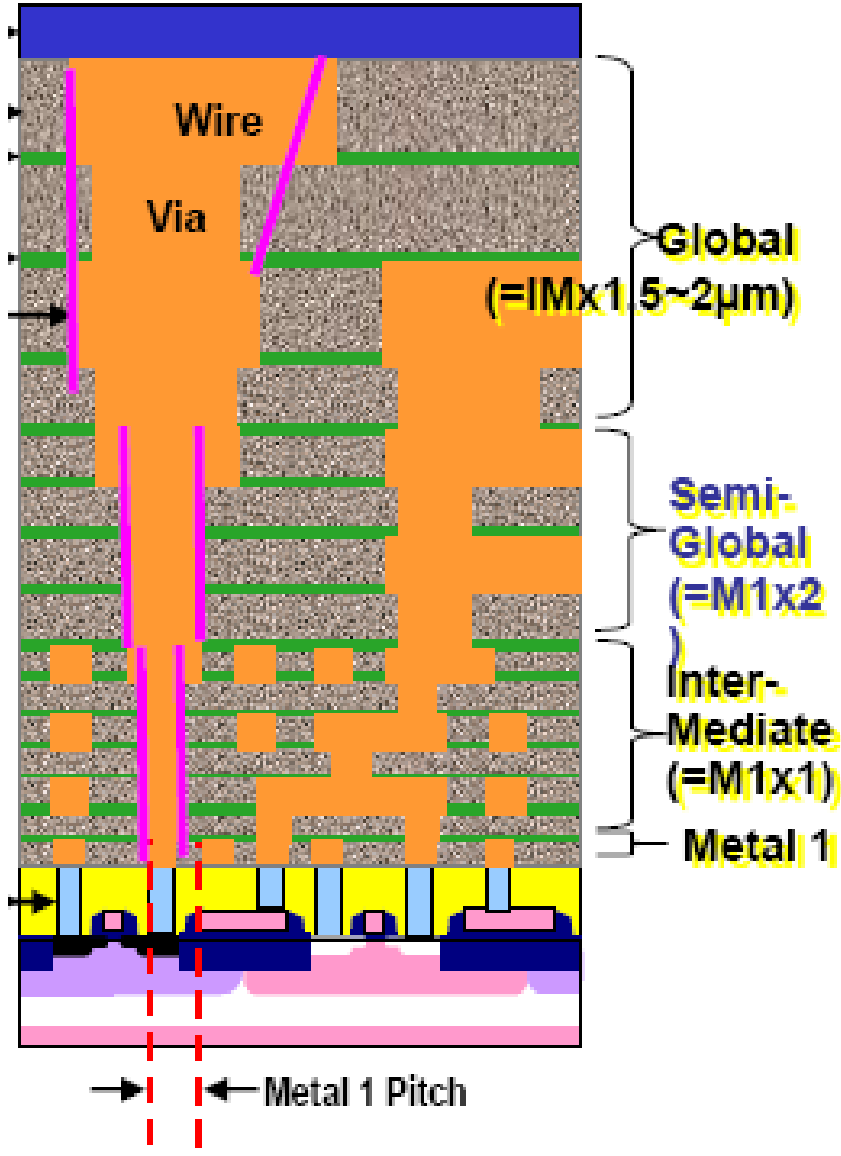


80 km wires



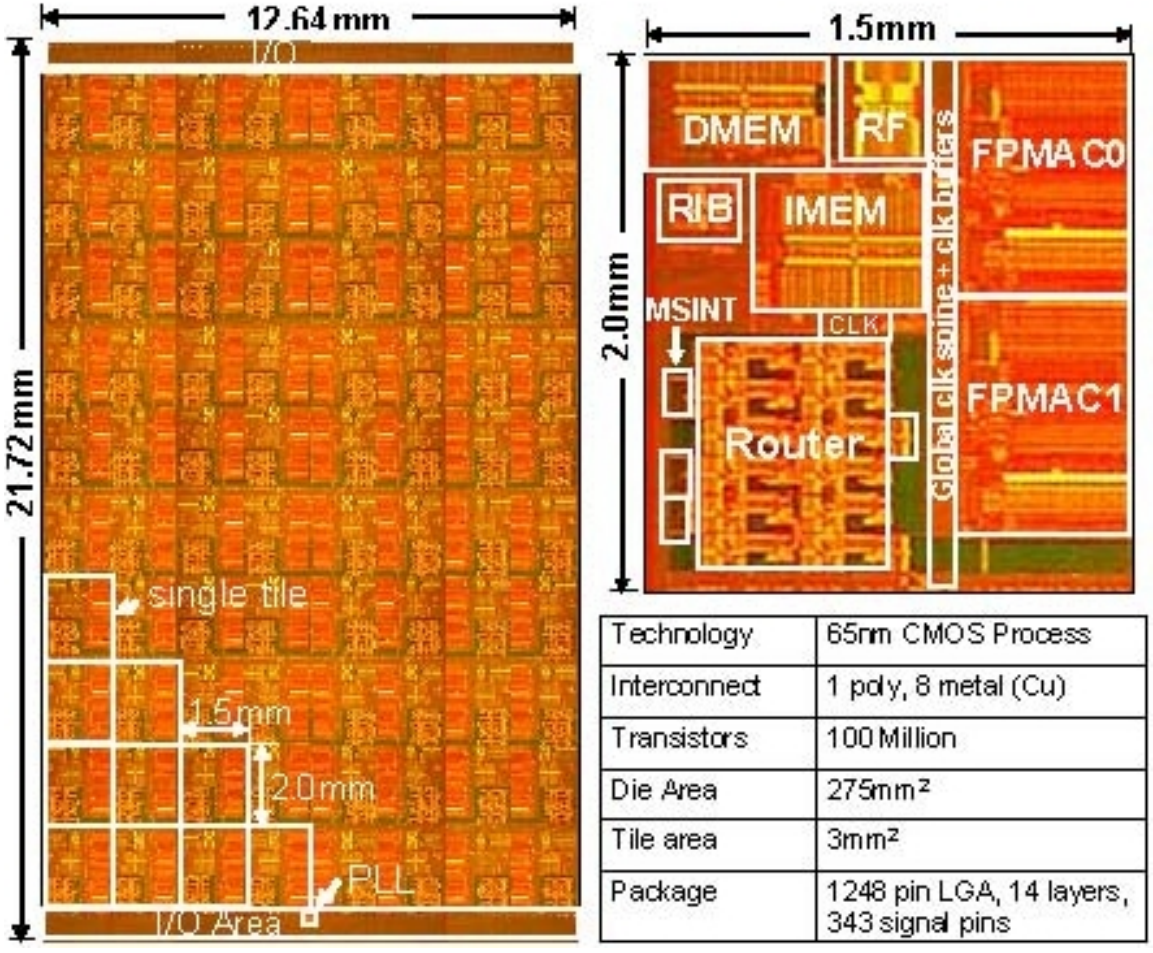
5 Million km Neuron connection

Wire Stack



ITRS 2012

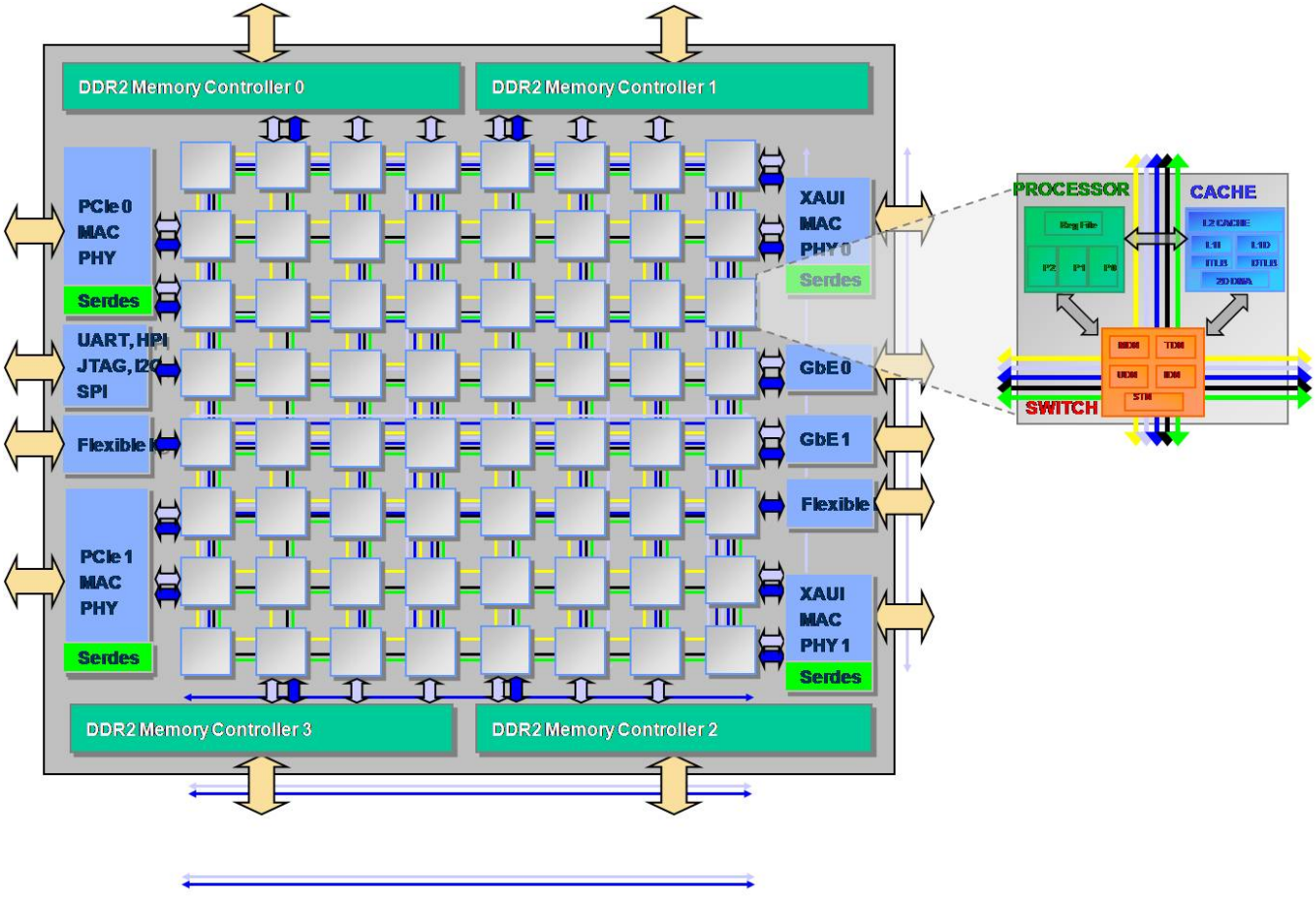
Intel's Teraflop Research Project



Technology	65nm CMOS Process
Interconnect	1 poly, 8 metal (Cu)
Transistors	100 Million
Die Area	275mm²
Tile area	3mm²
Package	1248 pin LGA, 14 layers, 343 signal pins

- 80 cores
- 100 Million transistors
- 65nm process
- 3.16 GHz
- 0.95 V
- 62 W
- 1.62 Terabit/s aggregate bandwidth
- 1.01 Teraflops

Tilera's TILEPro64



Tilera's TILEPro64™ Processor

Multicore Performance (90nm)

Number of tiles	64
Cache-coherent distributed cache	5 MB
Operations @ 750MHz (32, 16, 8 bit)	144-192-384 BOPS
Bisection bandwidth	2 Terabits per second

Power Efficiency

Power per tile (depending on app)	170 – 300 mW
Core power for h.264 encode (64 tiles)	12W
Clock speed	Up to 866 MHz

I/O and Memory Bandwidth

I/O bandwidth	40 Gbps
Main Memory bandwidth	200 Gbps

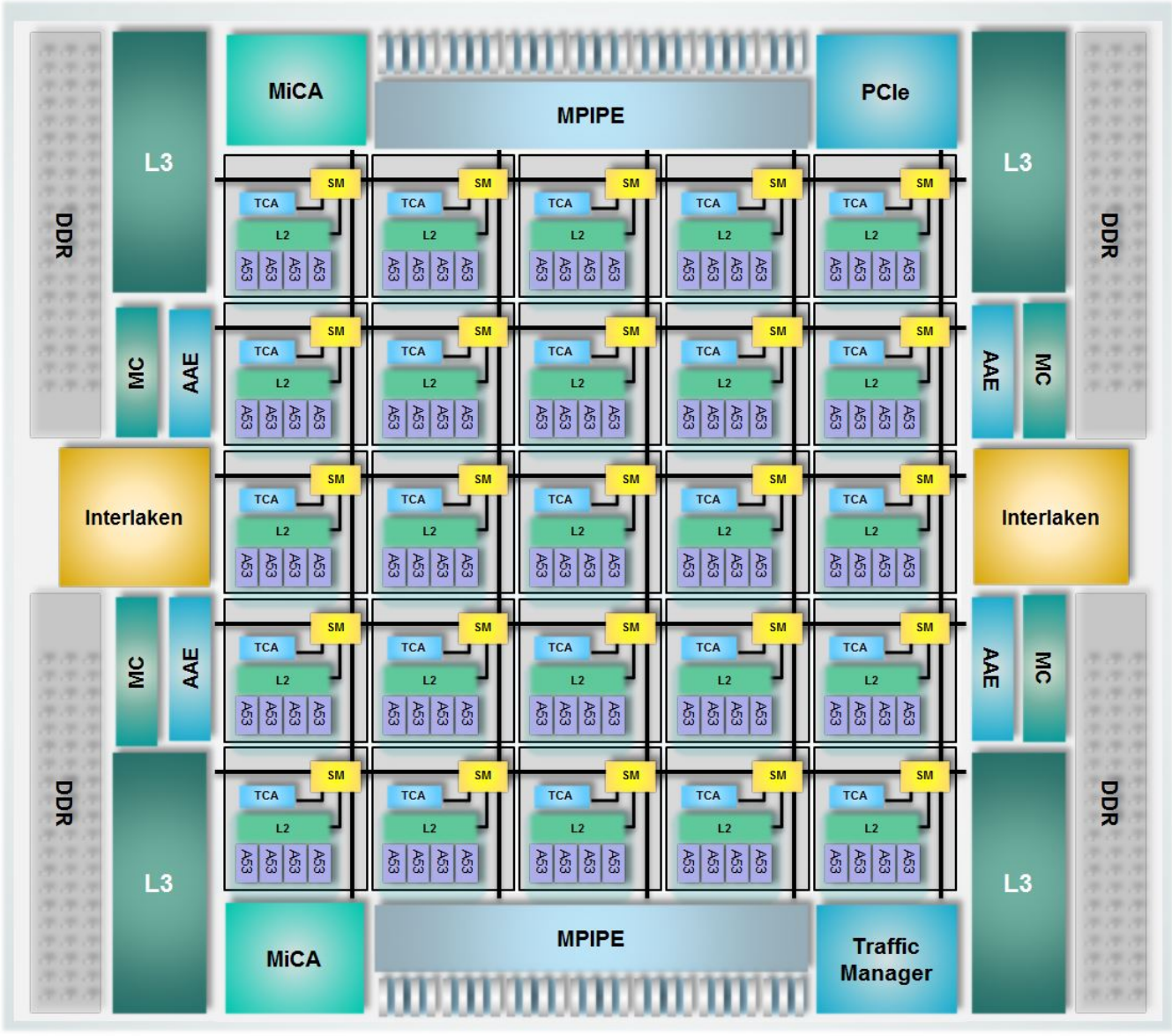
Programming

ANSI standard C
SMP Linux programming
Stream programming

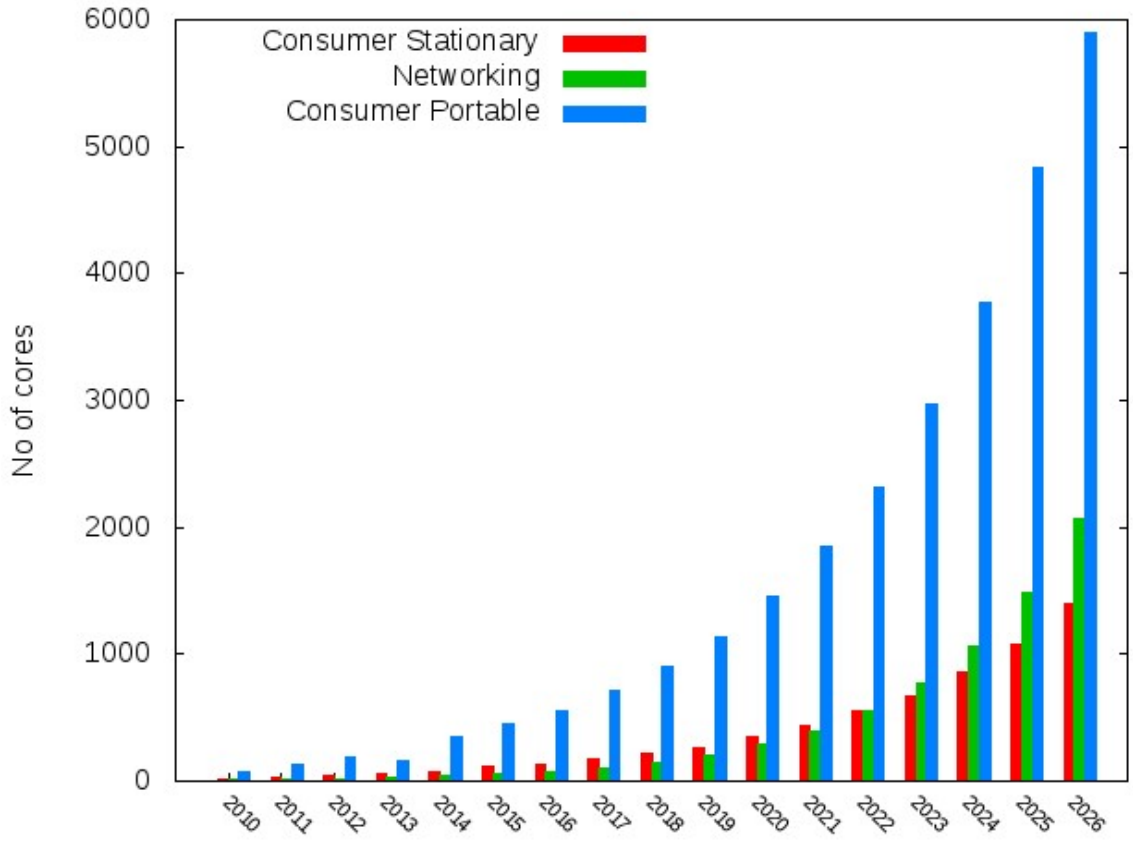
Product reality

[Tile64, Hotchips 2007]
[Tile64, Microprocessor Report Nov 2007]

EZChip / Tilera's TILE-Mx100



Number of Cores on Chip



International Roadmap for Semiconductors 2012 edition
(http://www.itrs.net/)

Moore's Law Rephrased

Number of transistors double every 18 months.

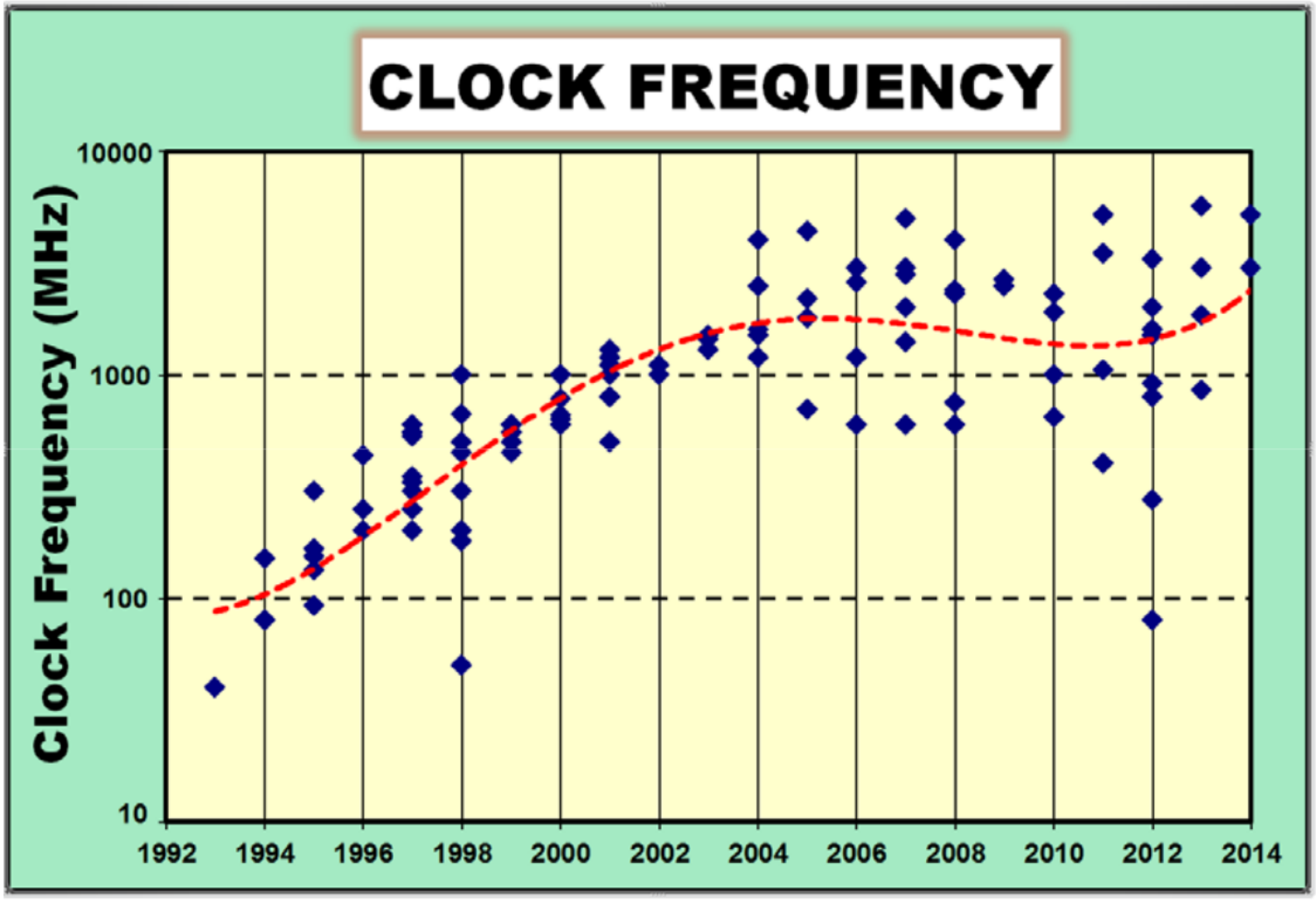
Number of cores will double every 26 months.

The Core is the logic gate of the 21^{st} century.

Why Multi-Core?

- Parallelism is power efficient
- Single core frequency is leveling off around 1 GHz

Clock Frequency is leveling off

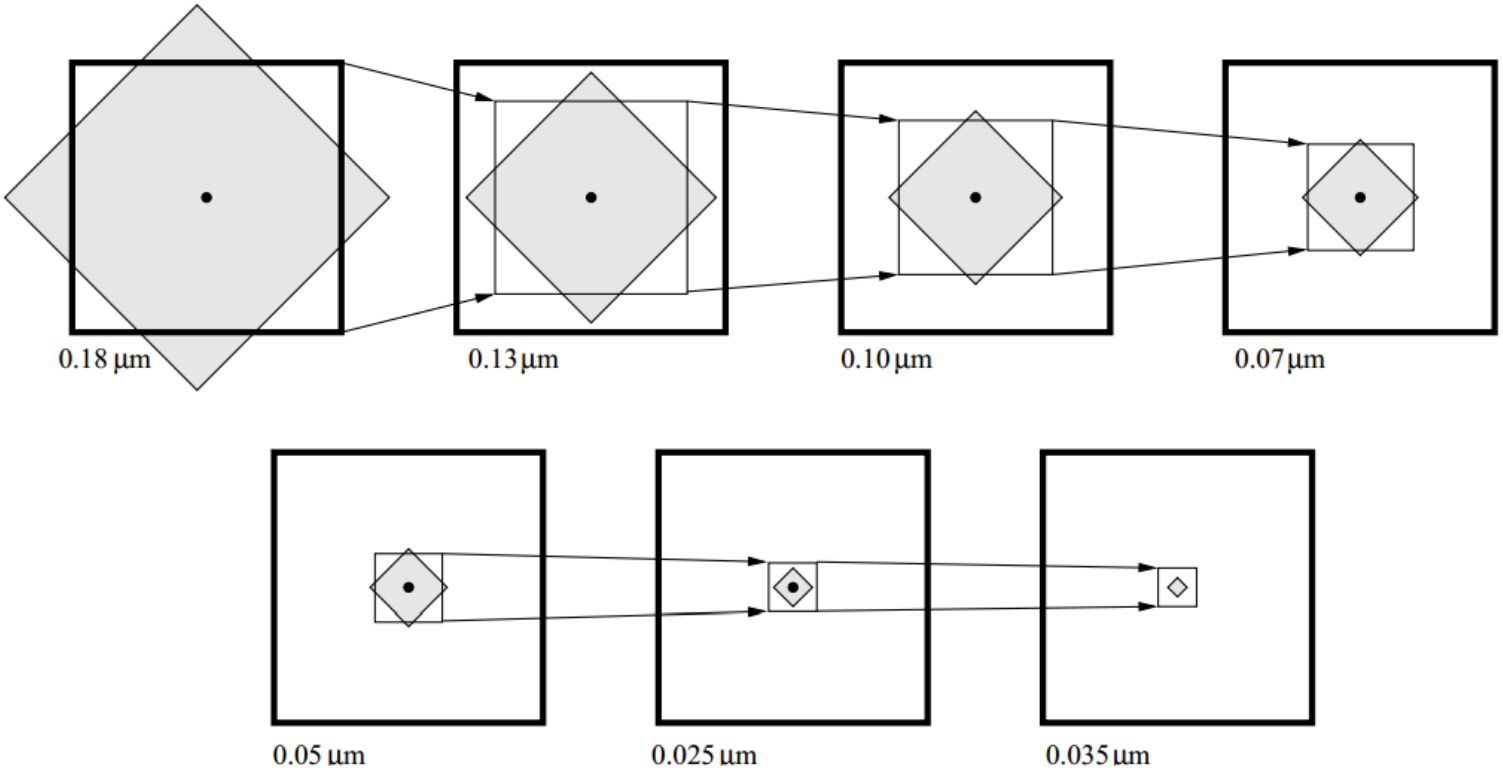


Source: ISSCC 2015 Technology Trends

Why Multi-Core?

- Parallelism is power efficient
- Single core frequency is leveling off around 1 GHz
- Global wires do not scale
- On-chip global synchronicity is impossible

On-Chip Synchronicity is Hard

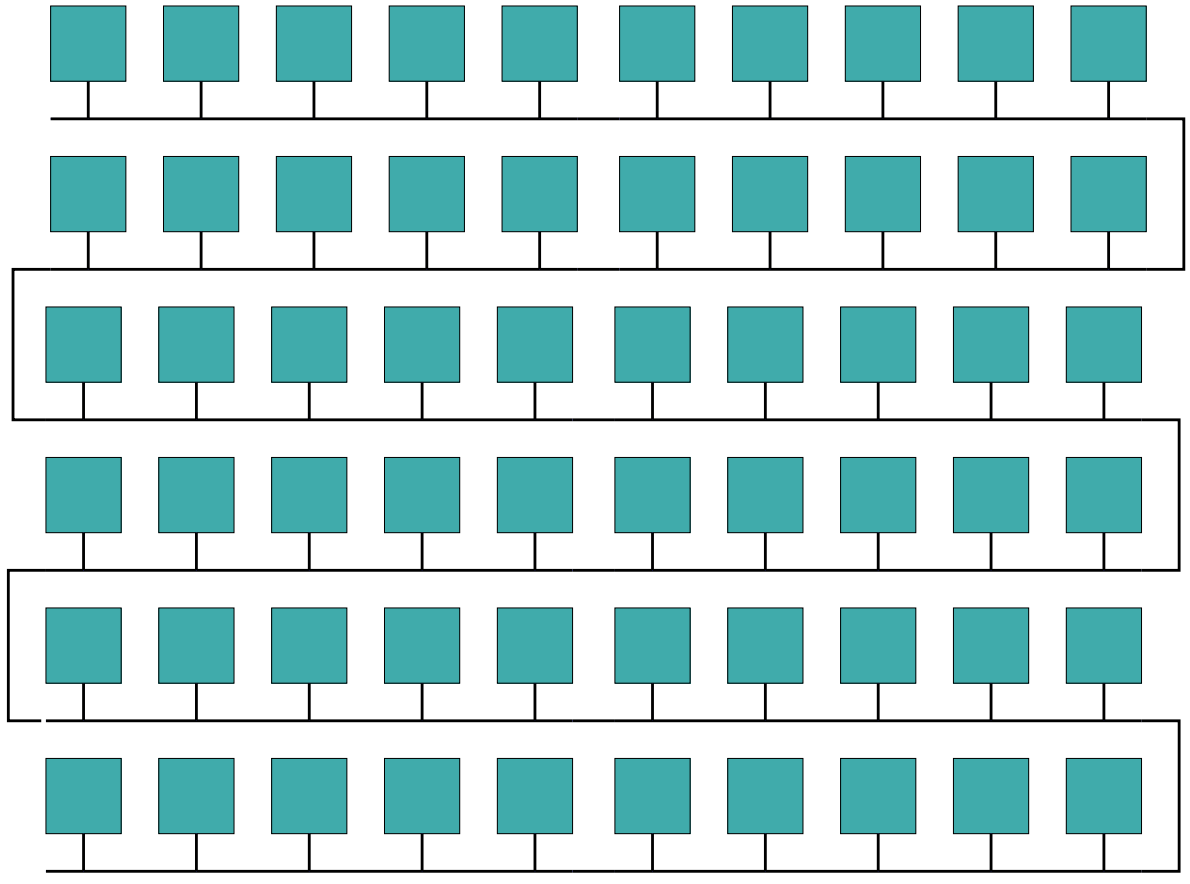


Source: Ron Ho. On-chip wires: Scaling and efficiency. PhD thesis. Stanford University, 2003

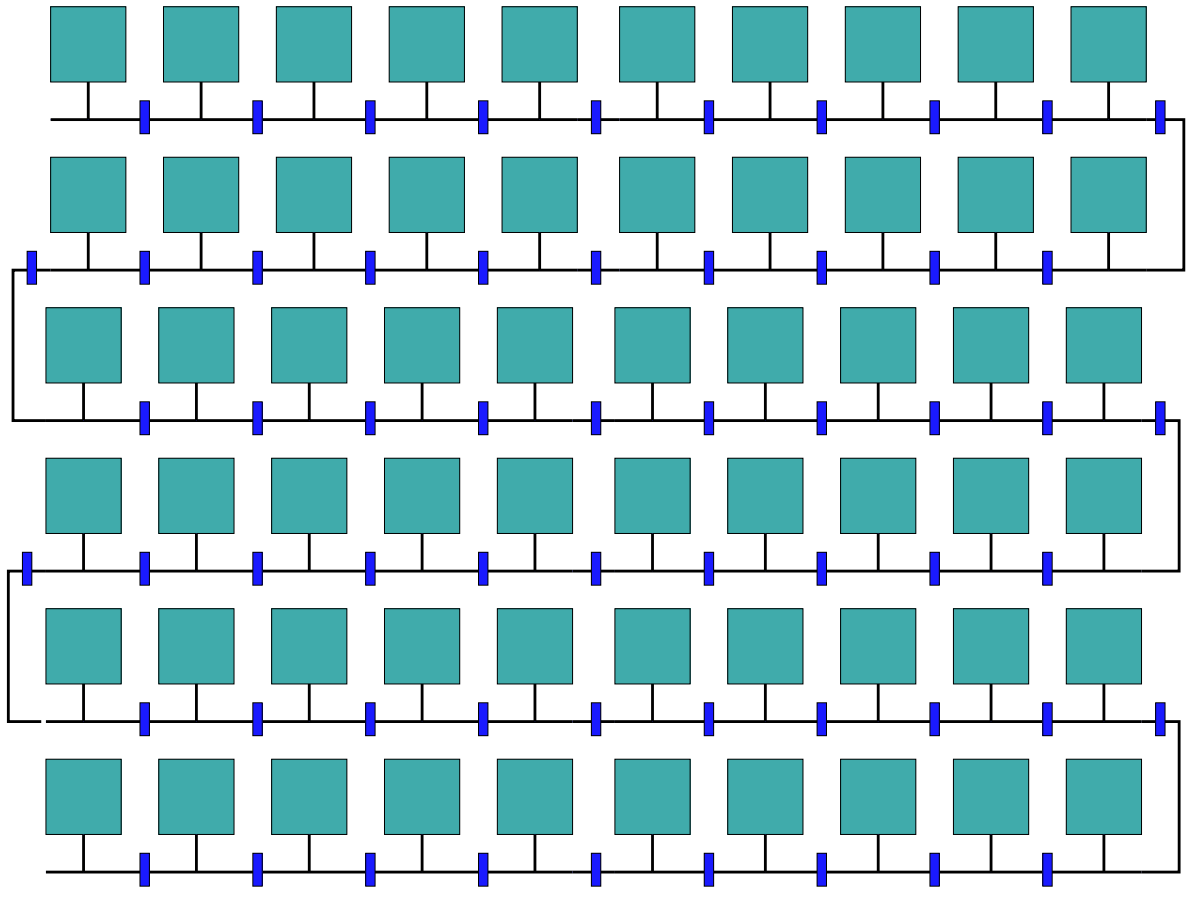
Why Multi-Core?

- Parallelism is power efficient
- Single core frequency is leveling off around 1 GHz
- Global wires do not scale
- On-chip global synchronicity is impossible
- The investment to make single cores more powerful is not paying back

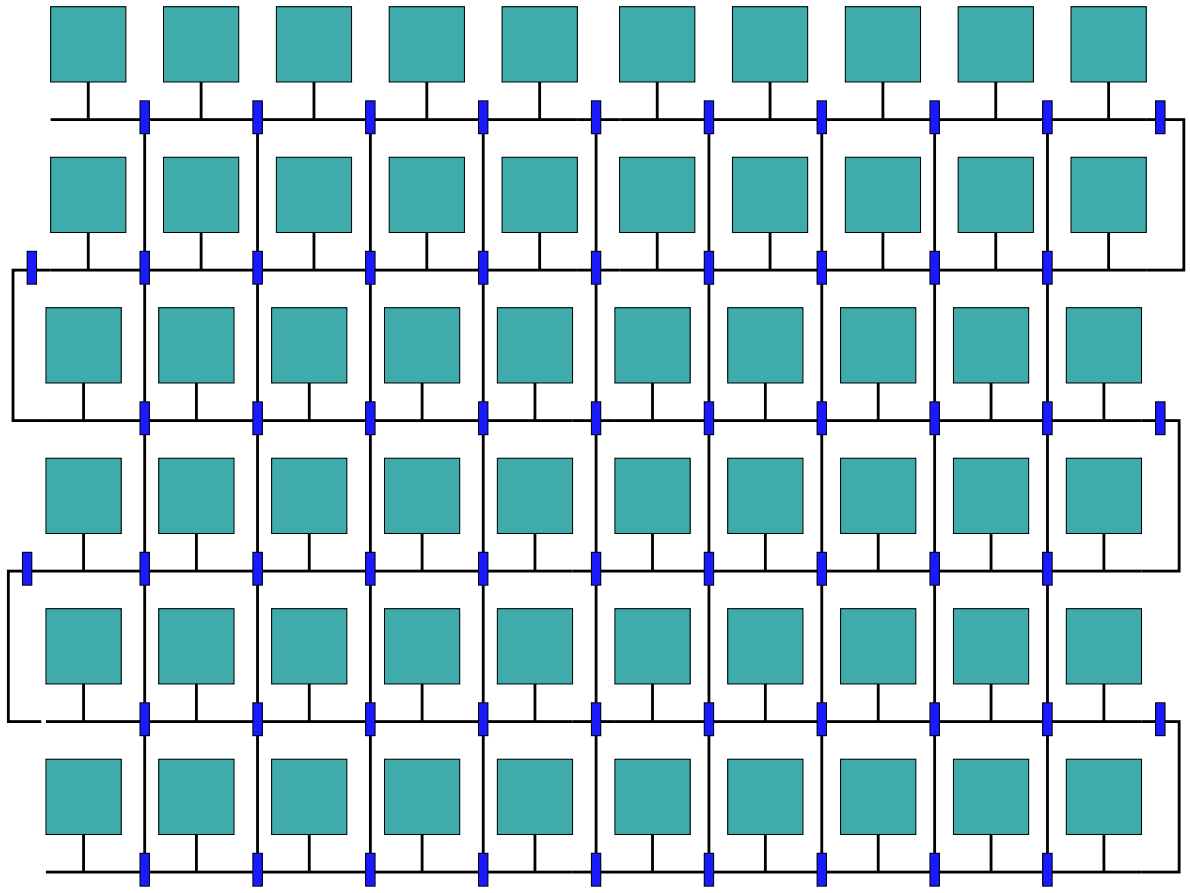
Why Networks on Chip? Buses Do Not Scale



Buses + Pipelining



Buses + Pipelining + Parallelism



Why Network on Chip?

- Buses do not scale
  - Neither in terms of performance
  - Nor in terms of power
- Due to many cores, communication must be parallel
- Due to non-scaling of global wires, communication must be pipelined
- If you have parallel and pipelined communication resources, you have to provide
  - Routing
  - Switching
  - Flow control

$\Rightarrow$ On-chip communication networks!

Network Layer Communication Performance in Network-on-Chips

Introduction

Communication Performance

Organizational Structure

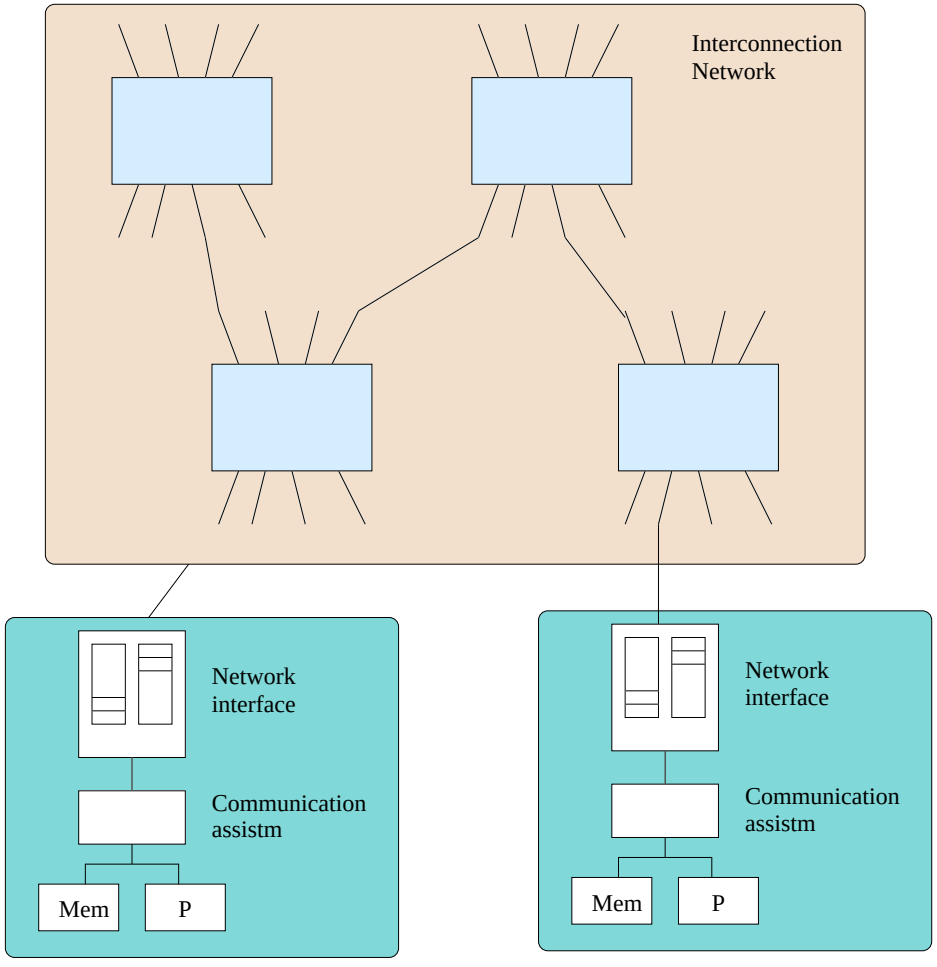
Interconnection Topologies

Trade-offs in Network Topology

Routing

Quality of Service

Introduction



- **Topology**: how switches and nodes are connected
- **Routing algorithm**: which route to take
- **Switching strategy**: how a message traverses the route
- **Flow control**: what do do when a message blocks

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.

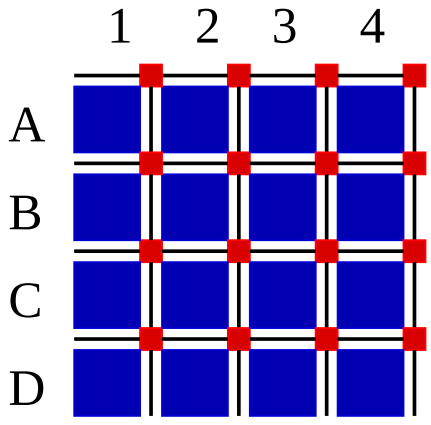
Store and Forward Packet Switching The entire packet is stored and then forwarded at each switch.

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



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

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

RoutingDelay is the delay for the route.

ContentionDelay is the delay of a message due to contention.

Routing Delay

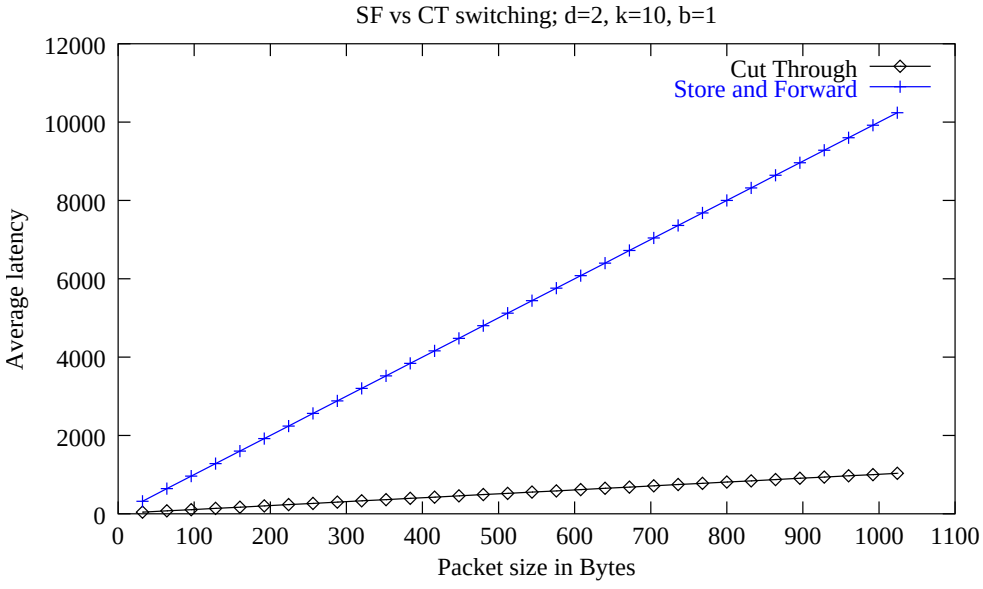
Store and Forward: $T_{sf}(m,h) = h(\frac{m}{b} + \Delta)$

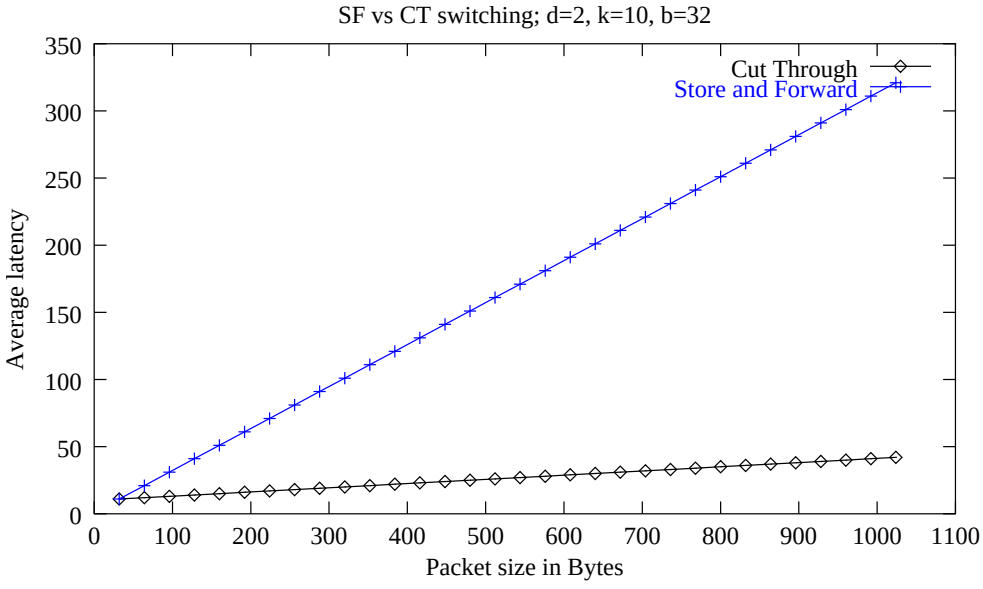
Circuit Switching: $T_{cs}(m,h) = \frac{m}{b} + h\Delta$

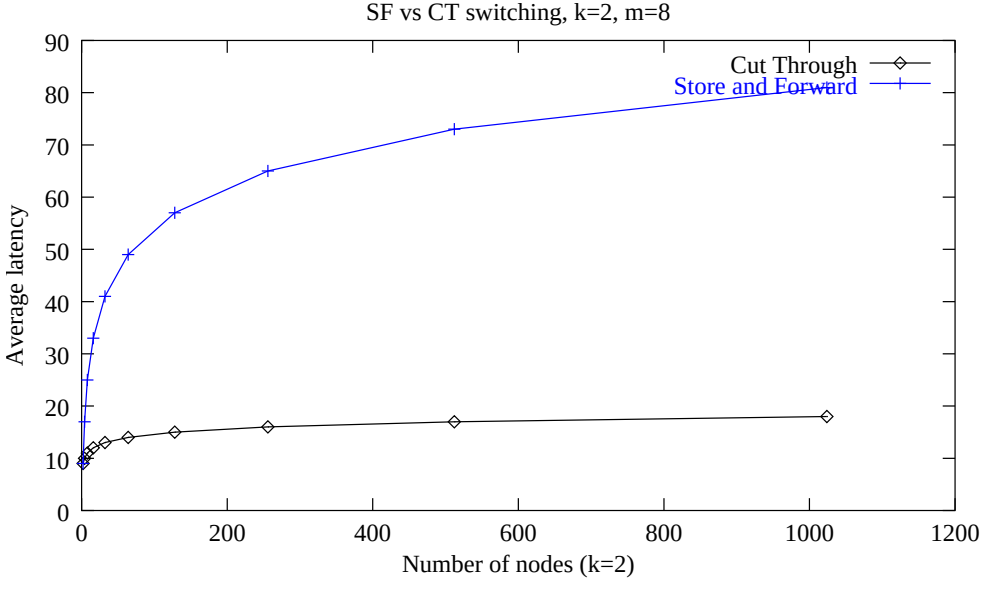
Cut Through: $T_{ct}(m,h) = \frac{m}{b} + h\Delta$

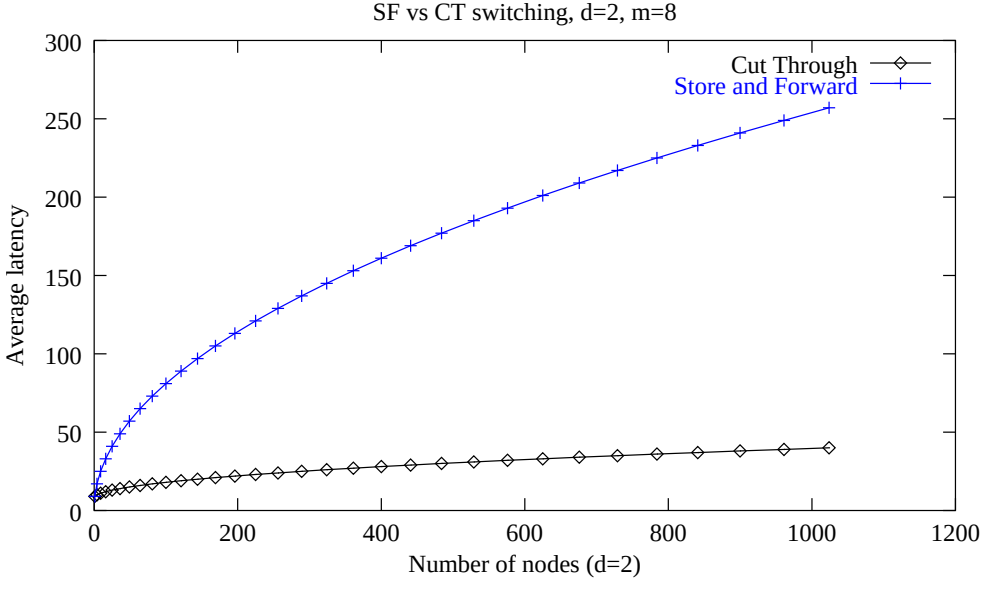
m ... message size in bits
h ... number of hops
b ... raw bandwidth of the channel
Δ ... switching delay per hop

Routing Delay: Store and Forward vs Cut Through









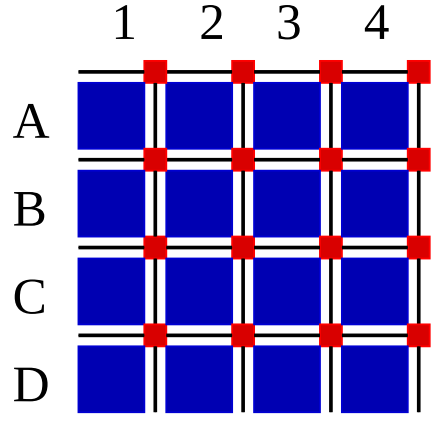
Local and Global Bandwidth

Local bandwidth	=	b [bits/second]
Total bandwidth	=	C b [bits/second]
Bisection bandwidth	...	minimum bandwidth to cut the net into two equal parts.

b ... raw bandwidth of a link; C ... total number of channels;

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

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



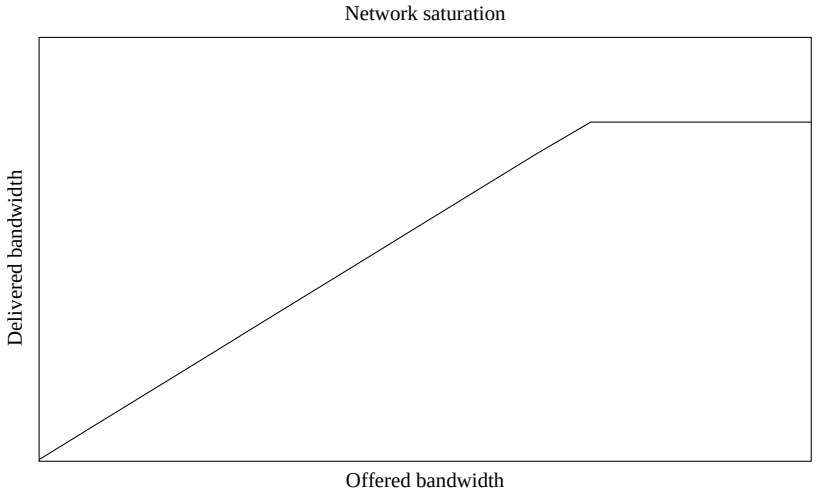
Link and Network Utilization

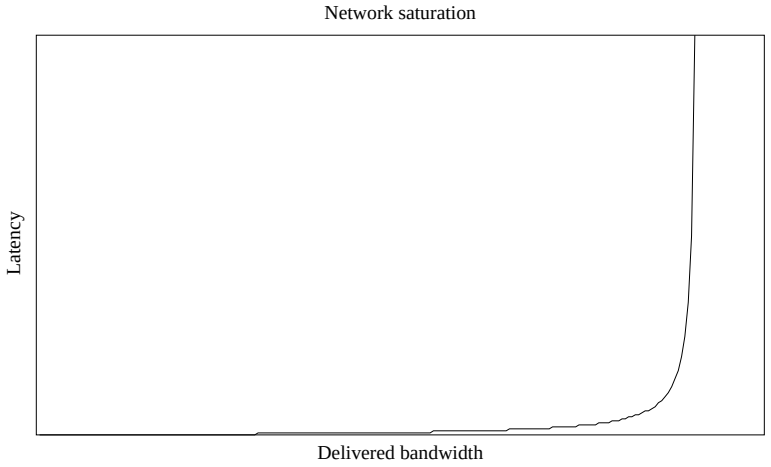
total load on the network: $L = \frac{Nhl}{M}$ [phits/cycle]

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

- Network topology
- Network size
- Link and switch bandwidth
- 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.

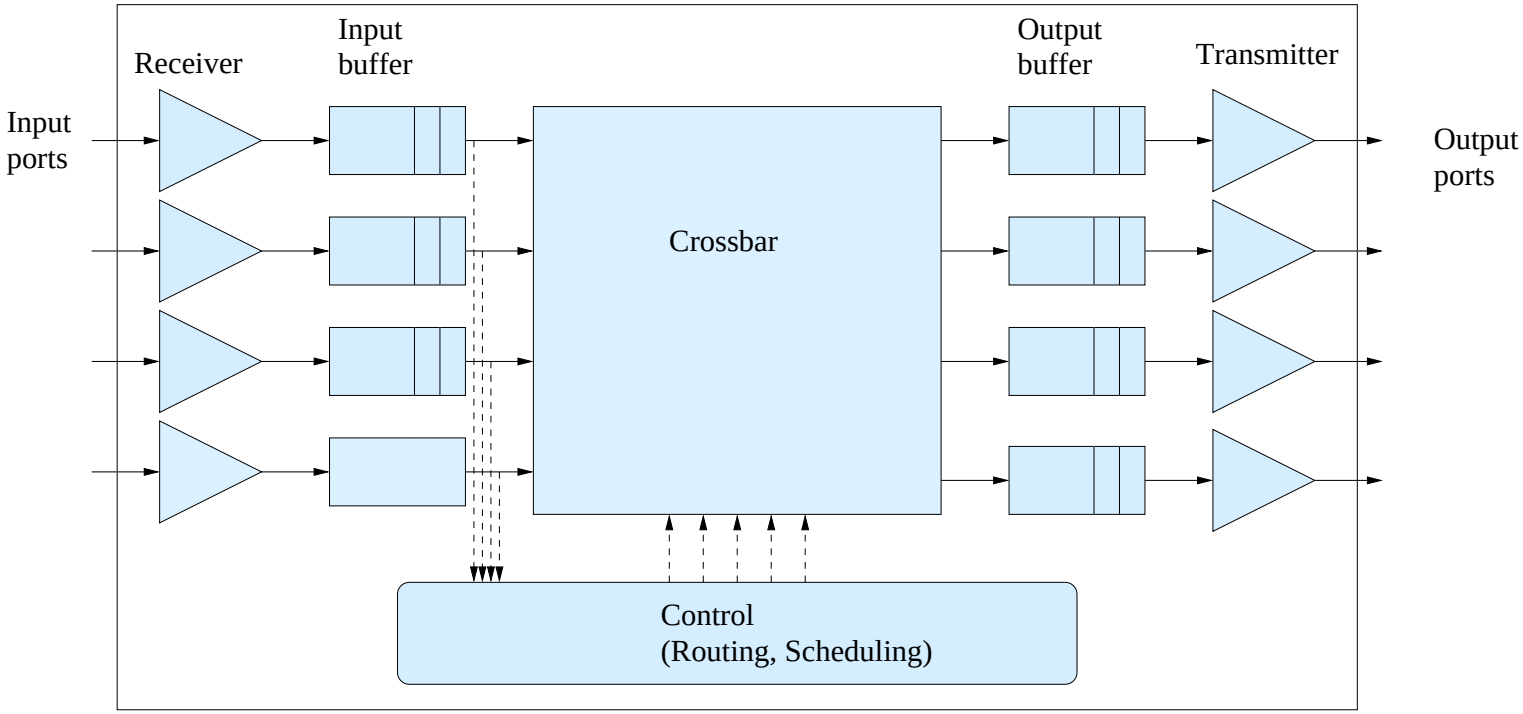
Narrow link Data and control information is multiplexed on the same wires.

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

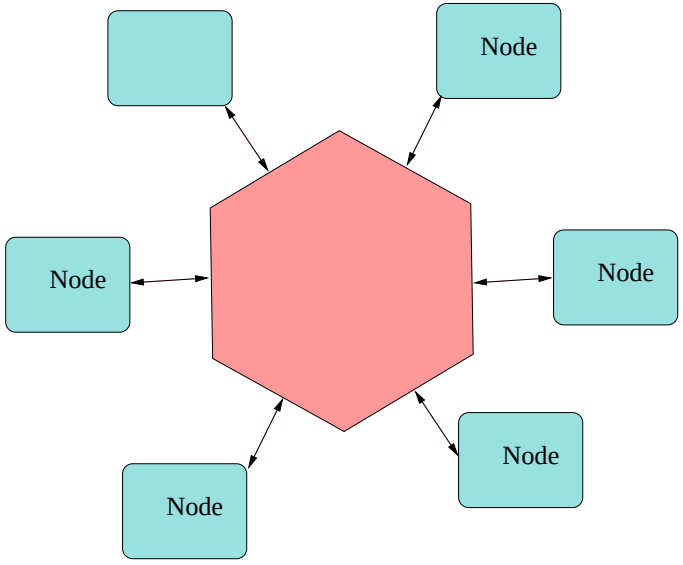
- Admission protocol
- Reception obligations
- Buffering
- Assembling and disassembling of messages
- Routing
- Higher level services and protocols

Interconnection Topologies

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

Which Topology to Choose?

Fully Connected Networks



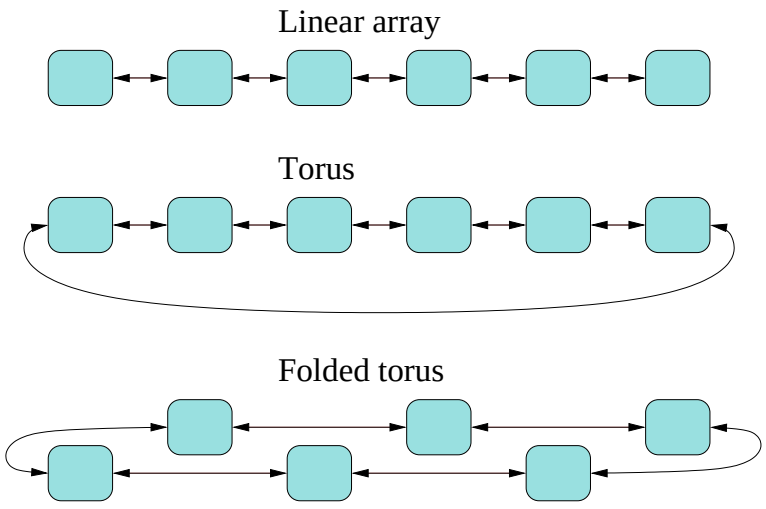
Bus:

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	=	$N(N-1)b$
bisection bandwidth	=	$N(N-1)b$

Linear Arrays and Rings



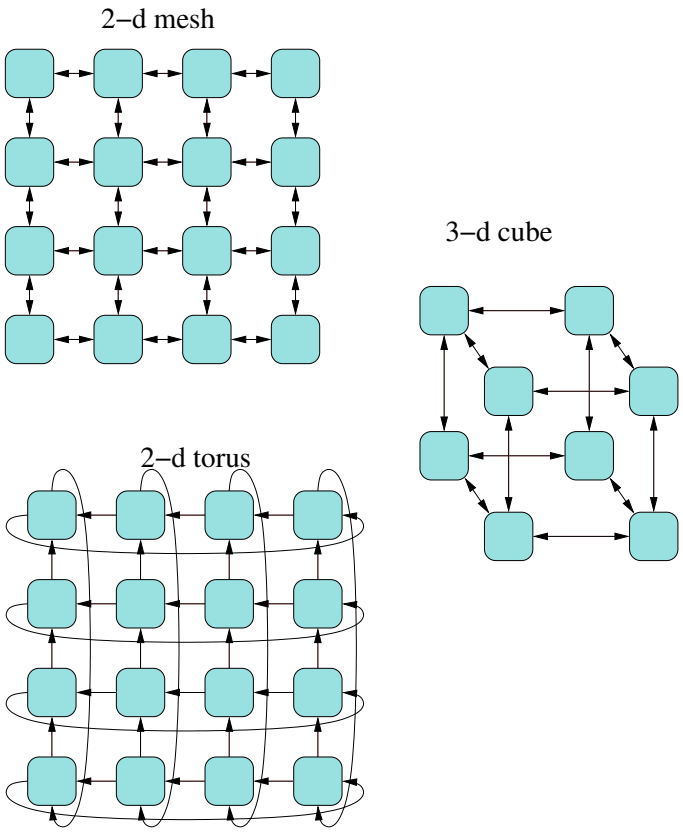
Linear array:

switch degree	$=$	2
diameter	$=$	$N-1$
distance	$\sim$	$\frac{2}{3}N$
network cost	$=$	$O(N)$
total bandwidth	$=$	$2(N-1)b$
bisection bandwidth	$=$	$2b$

Torus:

switch degree	$=$	2
diameter	$=$	$N/2$
distance	$\sim$	$\frac{1}{3}N$
network cost	$=$	$O(N)$
total bandwidth	$=$	$2Nb$
bisection bandwidth	$=$	$4b$

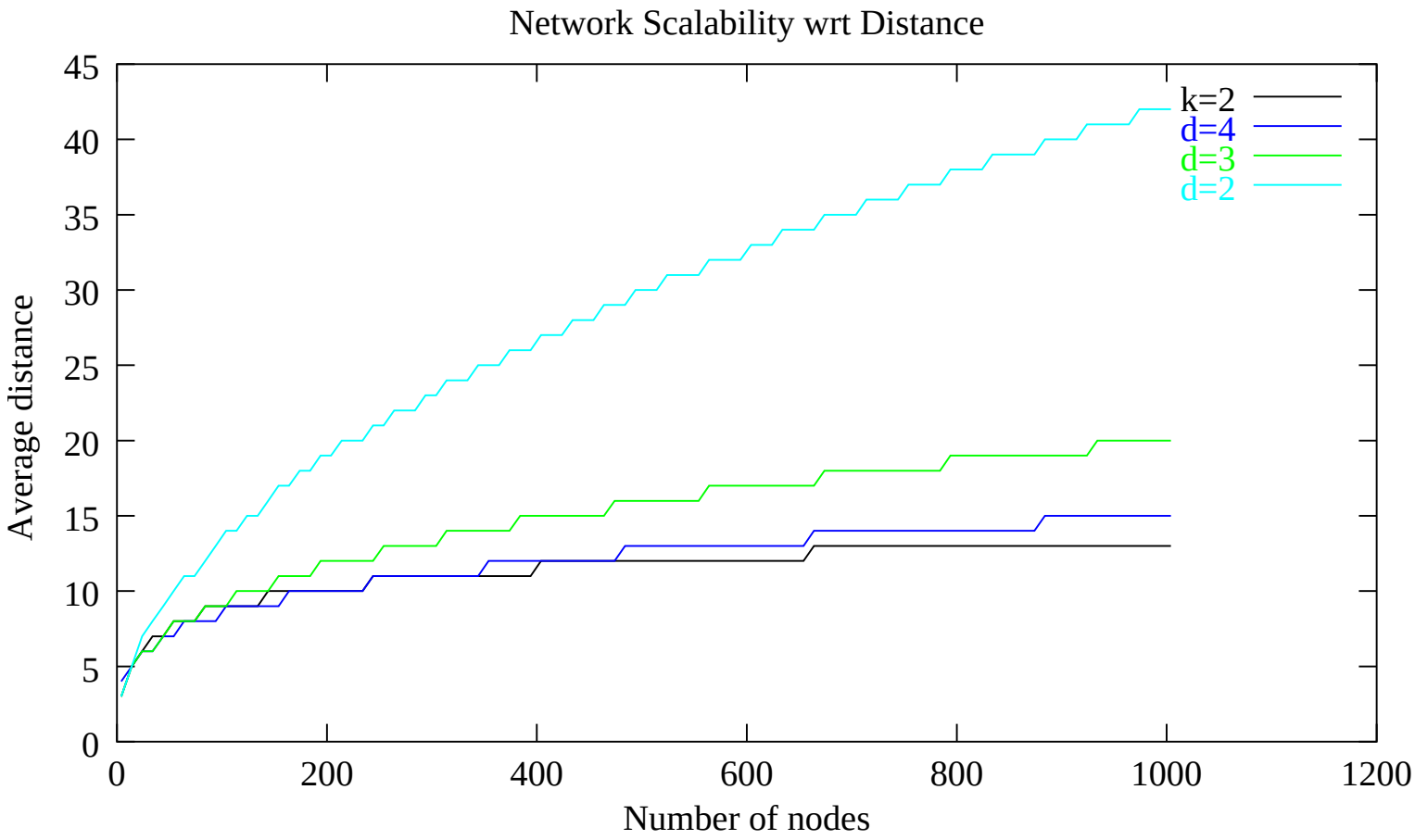
Multidimensional Meshes and Tori



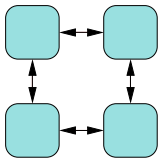
k-**ary** *d*-**cubes** are *d*-dimensional tori with unidirectional links and k nodes in each dimension:

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	$=$	$2k^{(d-1)}b$

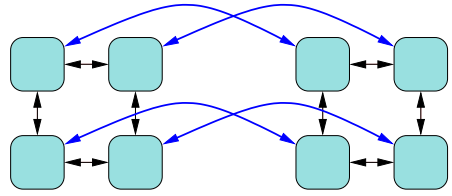
Routing Distance in k-ary d-Cubes



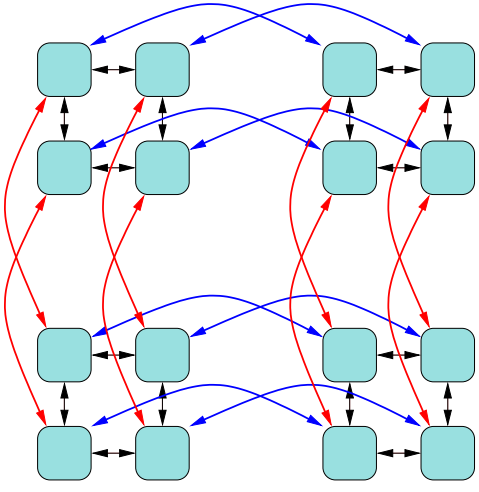
Projecting High Dimensional Cubes



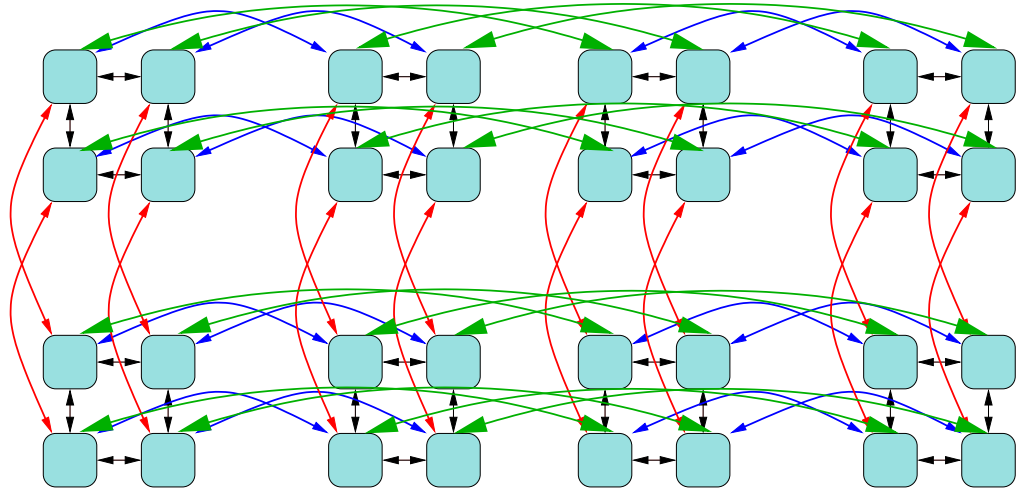
2−ary 2−cube



2−ary 3−cube

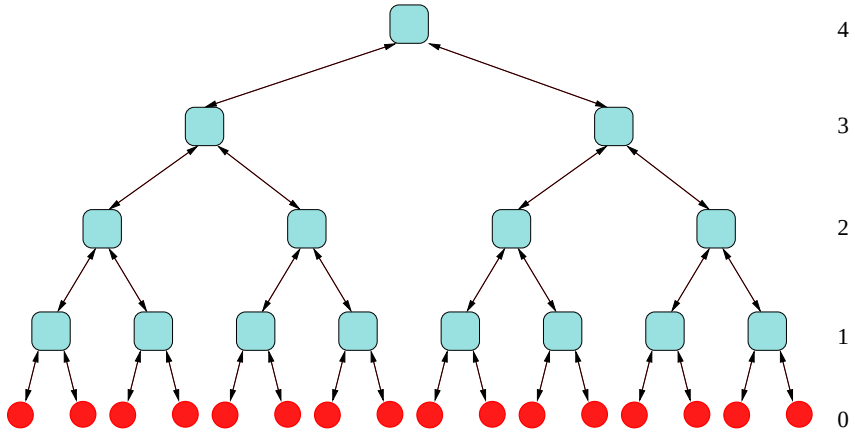


2−ary 4−cube



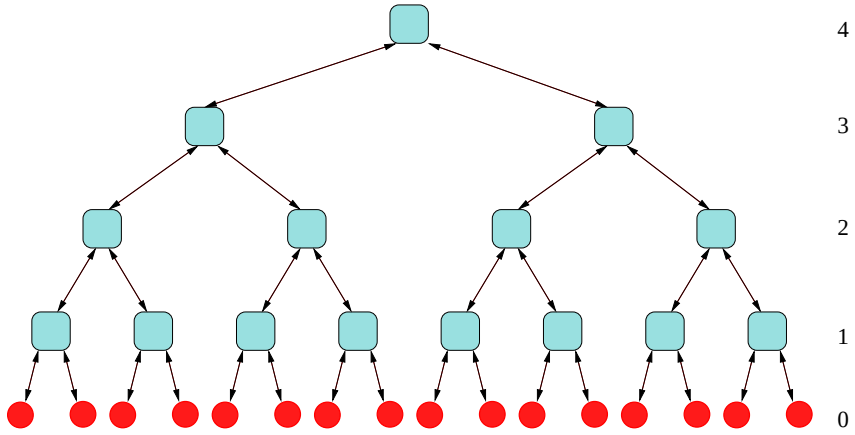
2−ary 5−cube

Binary Trees



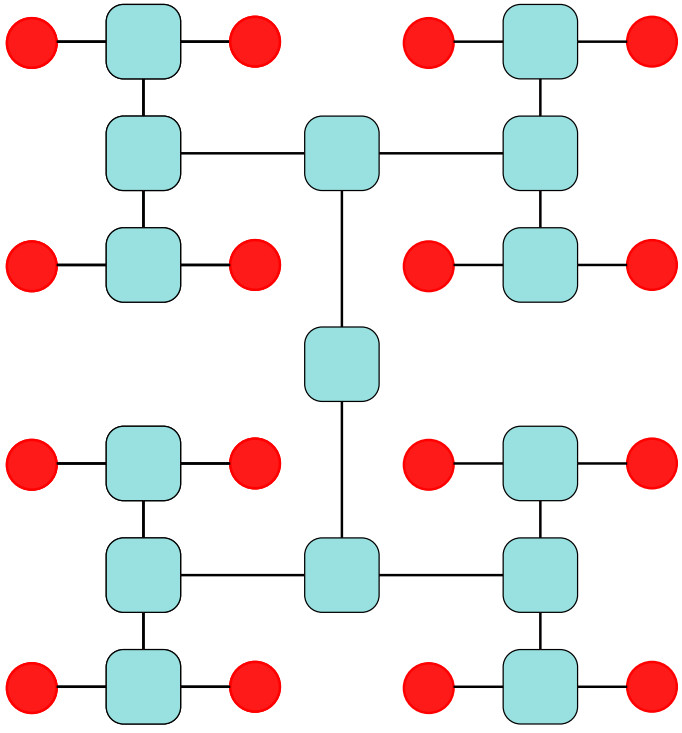
number of nodes N	$=$	2^d
number of switches	$=$	$2^d - 1$
switch degree	$=$	3
diameter	$=$	$2d$
distance	$\sim$	$d + 2$
network cost	$=$	$O(N)$
total bandwidth	$=$	$2 \cdot 2(N-1)b$
bisection bandwidth	$=$	$2b$

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

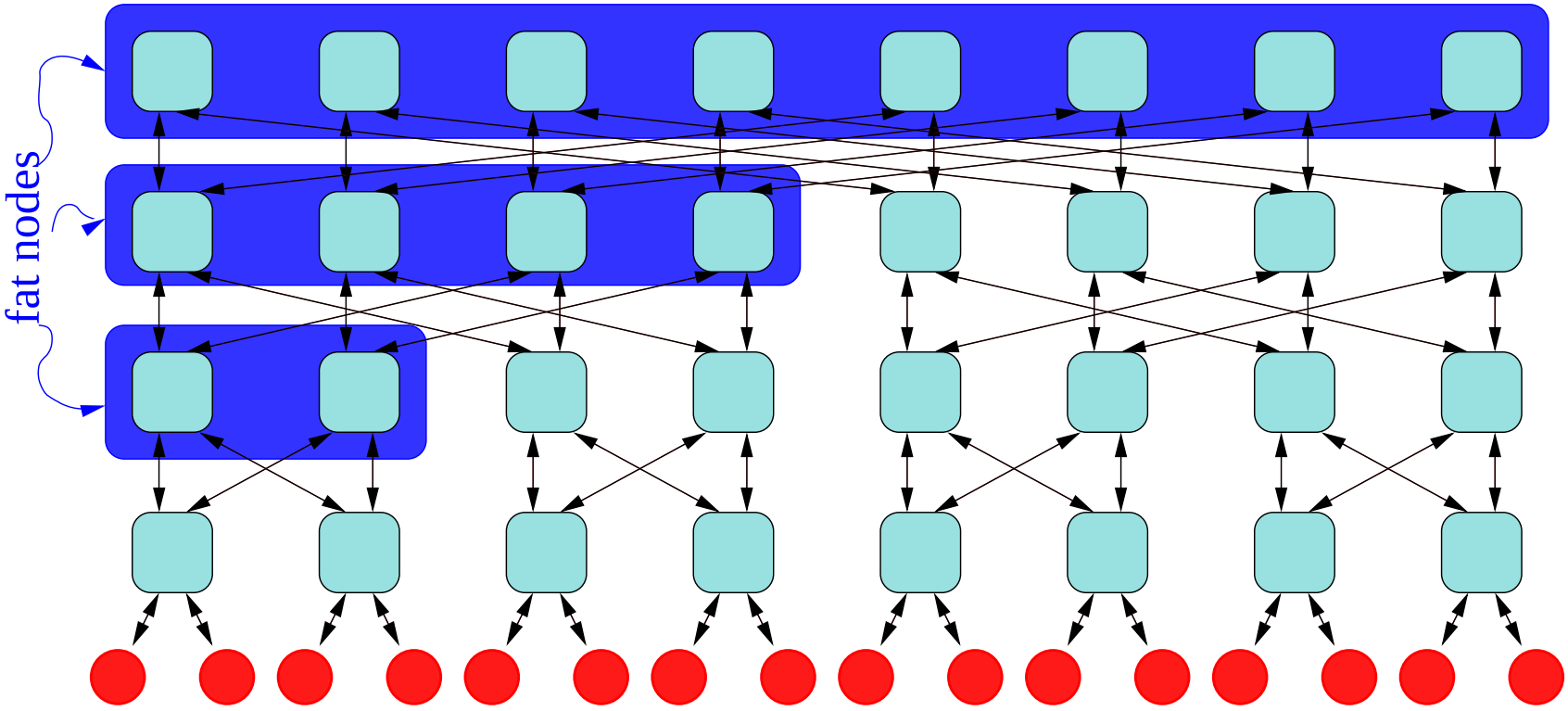


- Efficient and regular 2-layout;
- Longest wires in resource width:

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

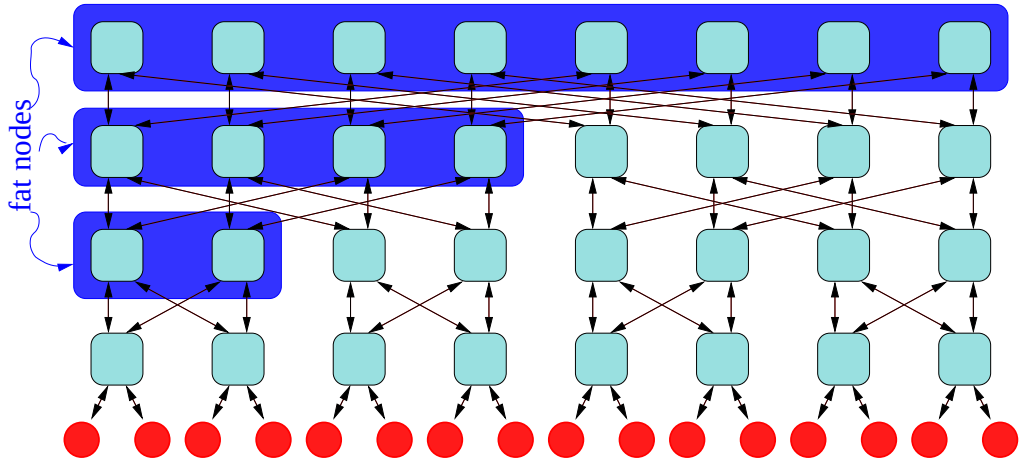
d	2	3	4	5	6	7	8	9	10
N	4	8	16	32	64	128	256	512	1024
IW	0	1	1	2	2	4	4	8	8

Fat Trees



16−node 2−ary fat−tree

k-ary n-dimensional Fat Tree Characteristics



16−node 2−ary fat−tree

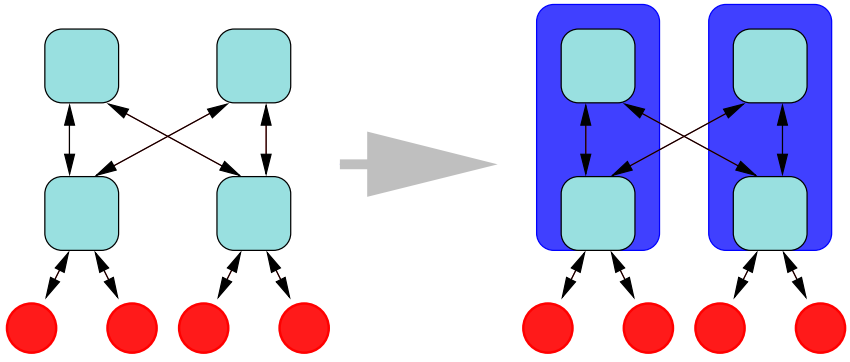
number of nodes N	$=$	k^d
number of switches	$=$	$k^{d-1}d$
switch degree	$=$	$2k$
diameter	$=$	$2d$
distance	$\sim$	d
network cost	$=$	$O(Nd)$
total bandwidth	$=$	$2Ndb$
bisection bandwidth	$=$	$2k^{d-1}b$

Shall we Choose a Fat Tree?

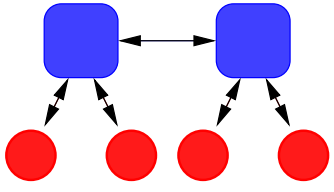
k-ary d-Cubes versus k-ary d-dimensional Fat Trees

	k-ary d-cubes	k-ary n-fat trees
number of nodes N	k^d	k^d
number of switches	k^d	$k^{d-1}d$
switch degree	d $k=2$	$2k$
diameter	$d(k-1)$	$2d$
distance	$d\frac{1}{2}(k-1)$	d
network cost	$O(N)$	$O(Nd)$
total bandwidth	$2Nb$	$2Ndb$
bisection bandwidth	$2k^{d-1}b$	$2k^{d-1}b$

Relation between Fat Tree and Hypercube

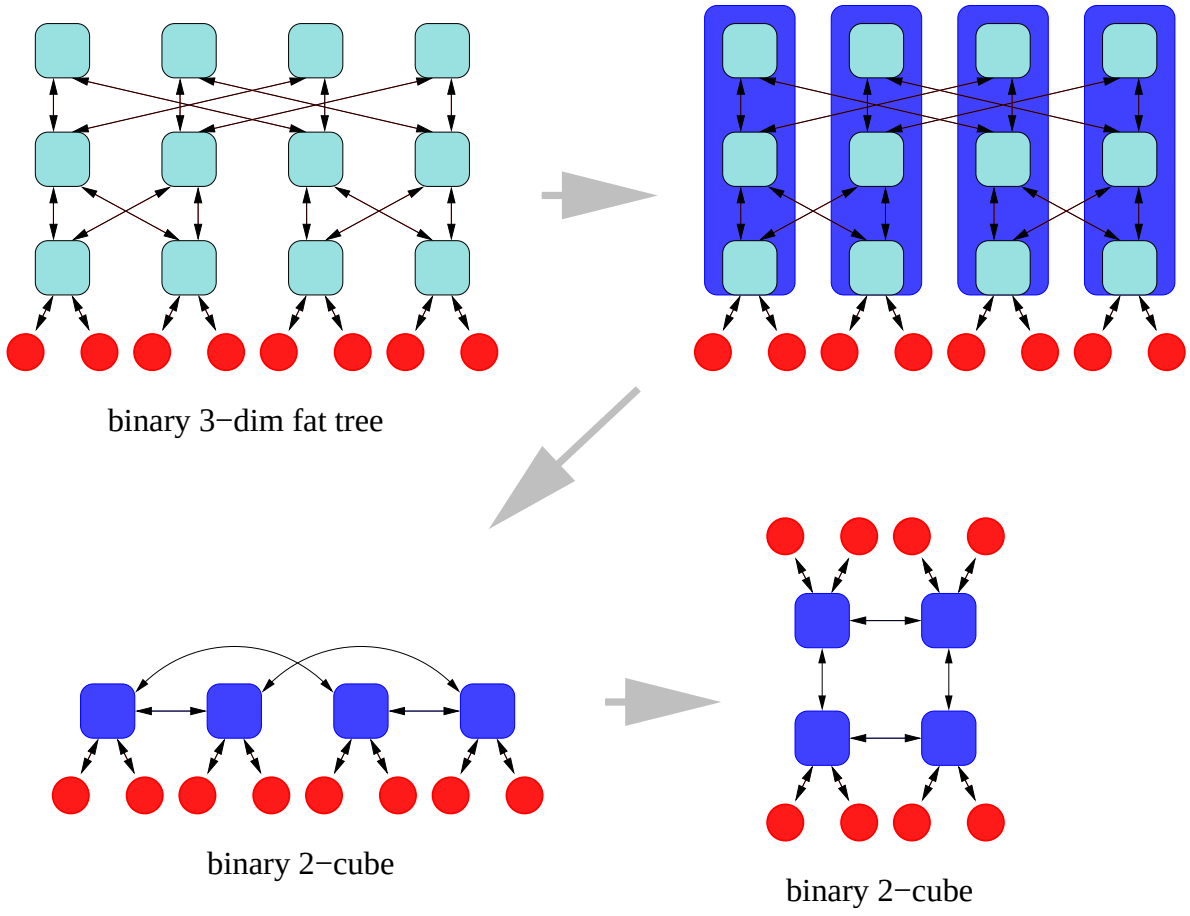


binary 2−dim fat tree

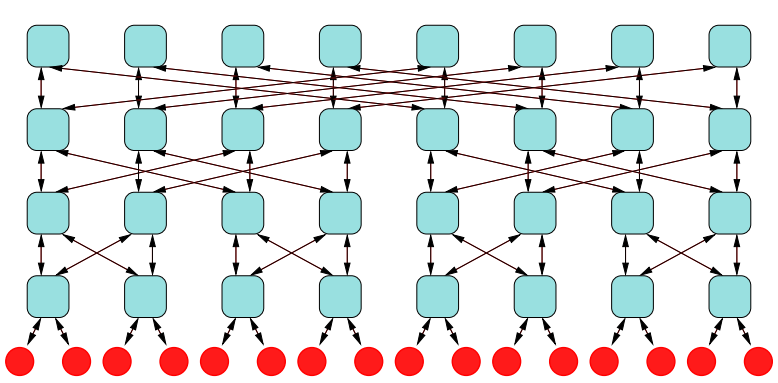


binary 1−cube

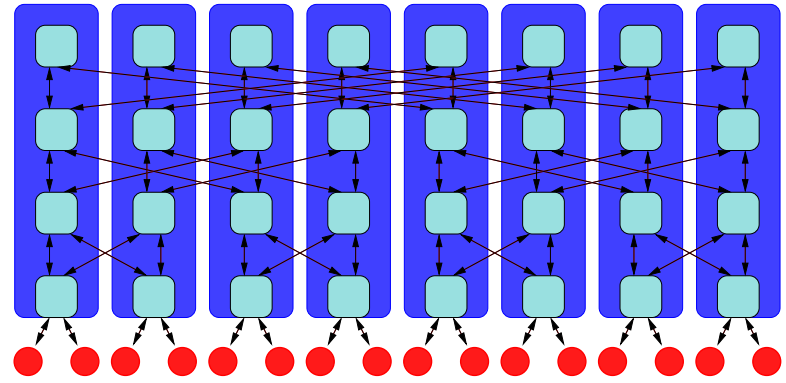
Relation between Fat Tree and Hypercube - cont'd

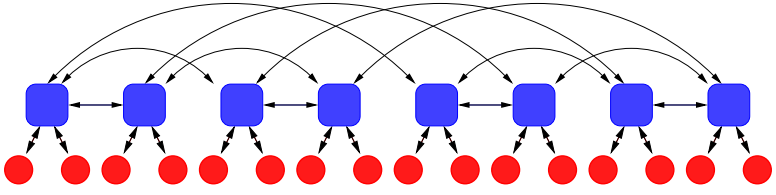


Relation between Fat Tree and Hypercube - cont'd

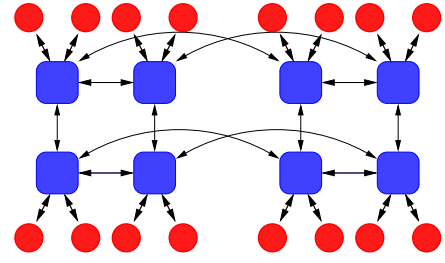


binary 4−dim fat tree





binary 3−cube



binary 3−cube

Let's Go for k-ary d-Cube!

Which d?

Trade-offs in Topology Design for the k-ary d-Cube

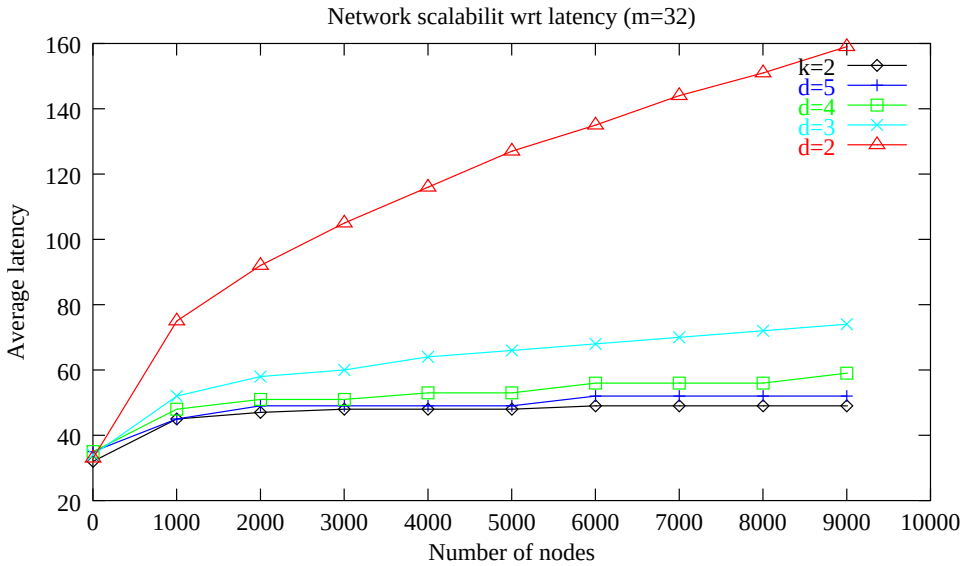
- Unloaded Latency
- Latency under Load

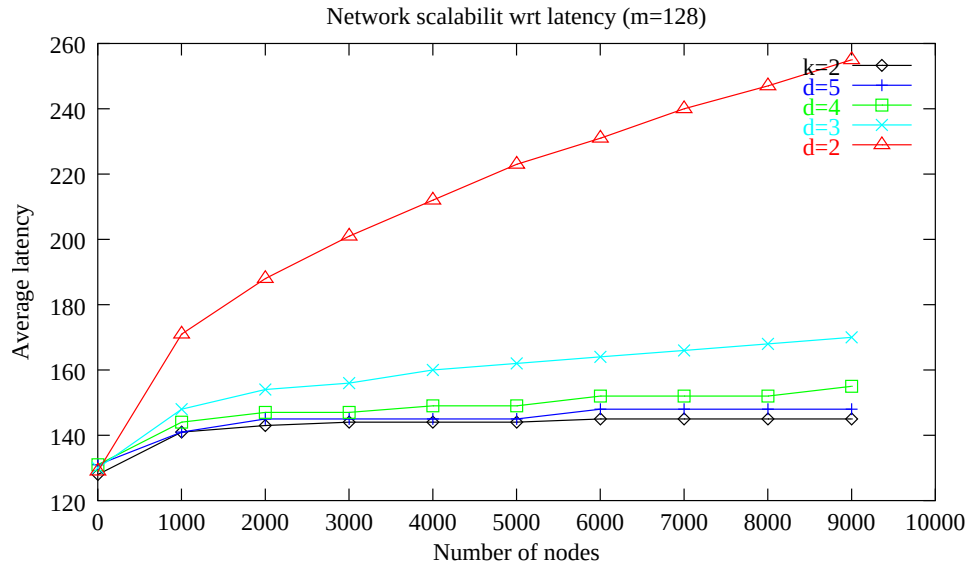
Network Scaling for Unloaded Latency

$$\text{Latency} = \text{Admission} + \text{RoutingDelay} + \text{ContentionDelay}$$

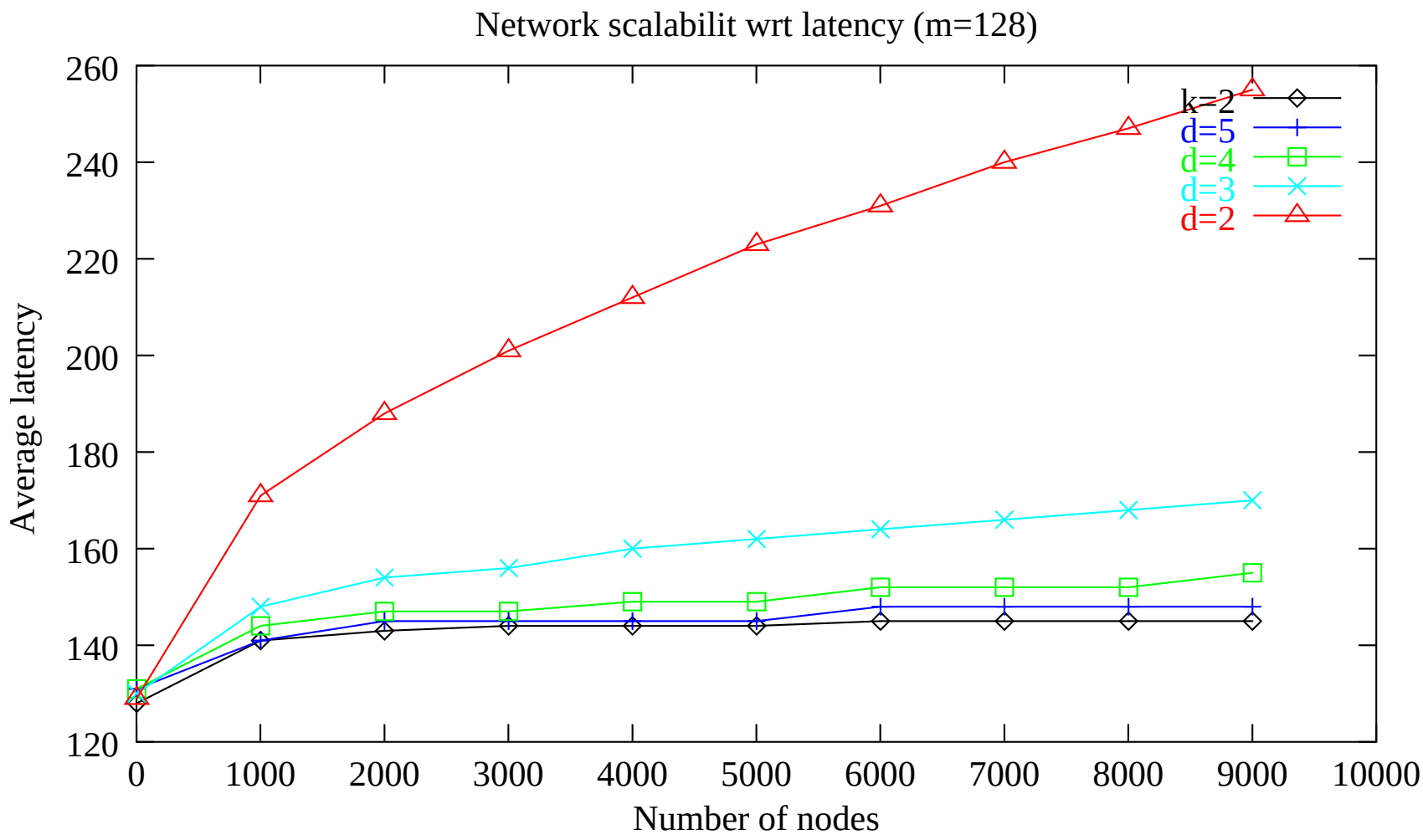
$$\text{RoutingDelay } T_{ct}(m,h) = \frac{m}{b} + h\Delta$$

$$\text{RoutingDistance } h = \frac{1}{2}(k-1)d$$



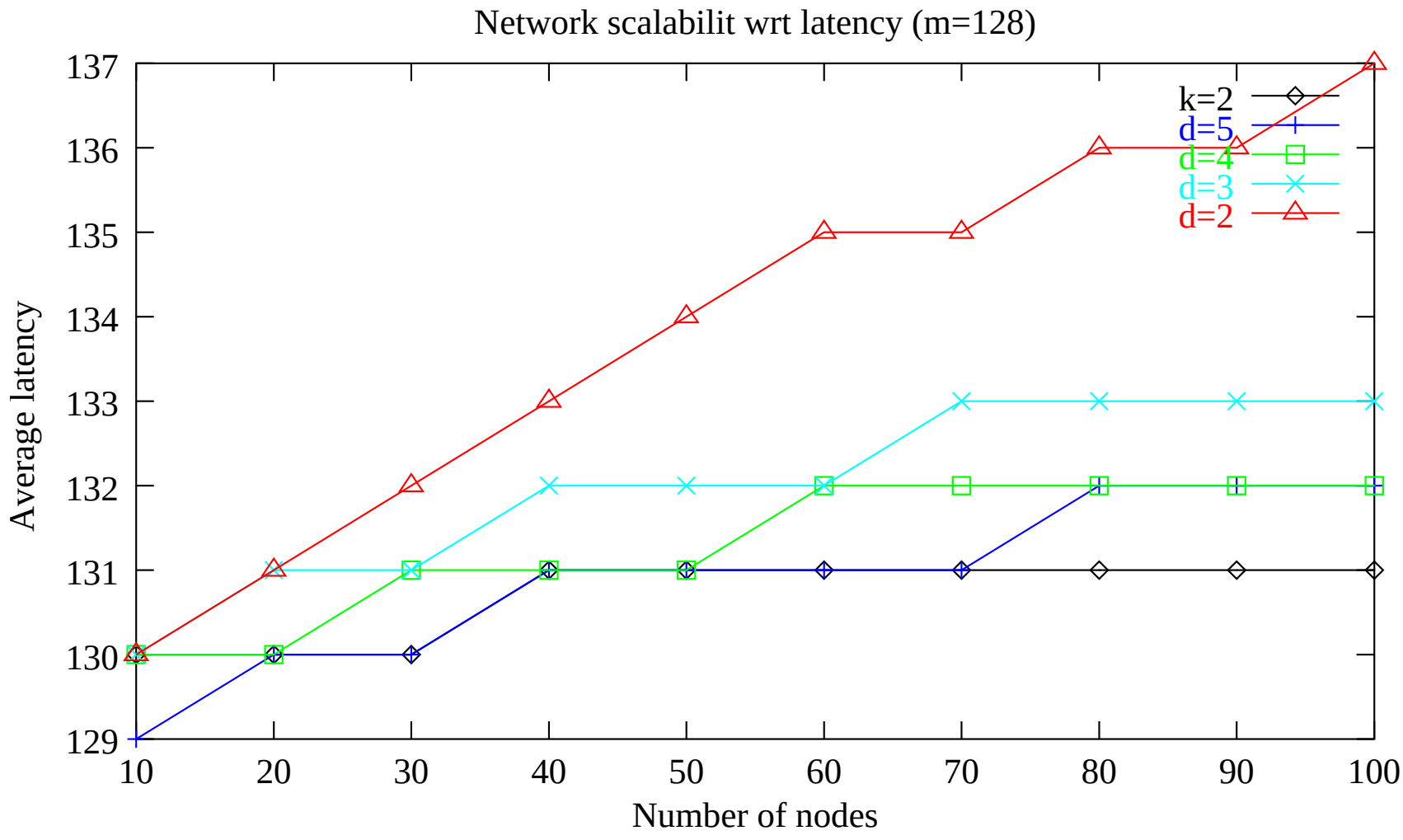


Scalability of Topology

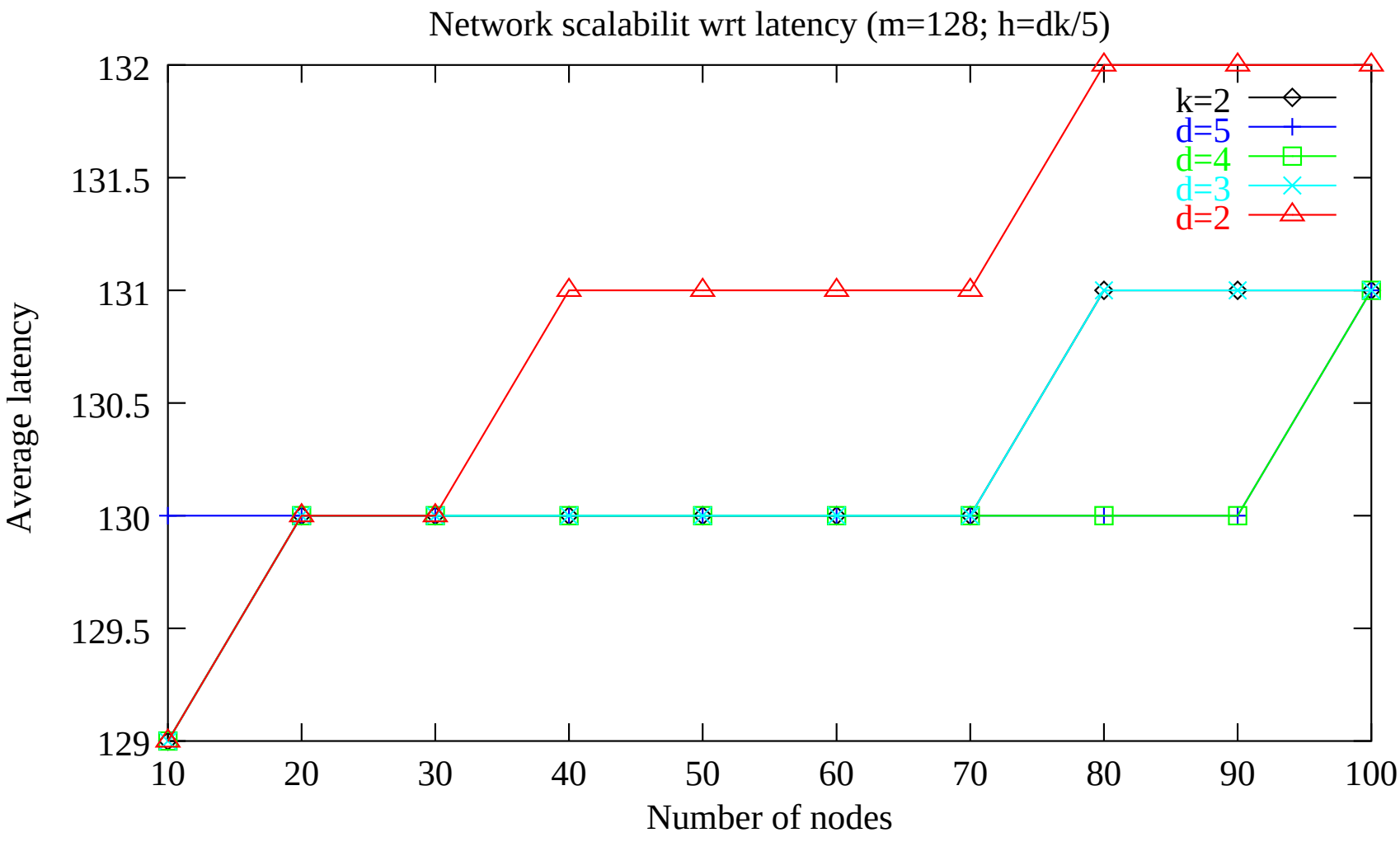


Are Hypercubes with $k = 2$ Best?

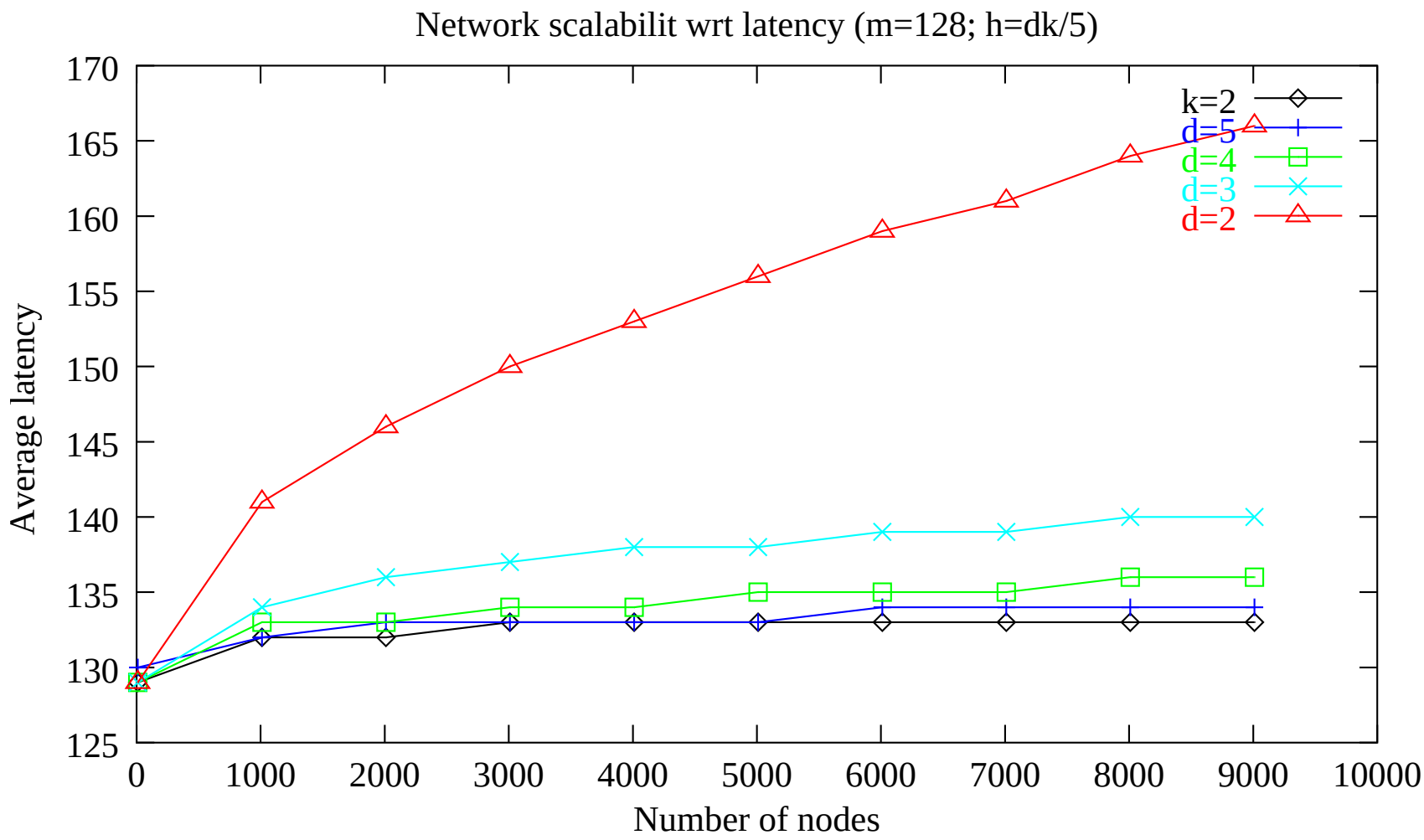
Unloaded Latency for Small Networks



Unloaded Latency for Small Networks and Local Traffic

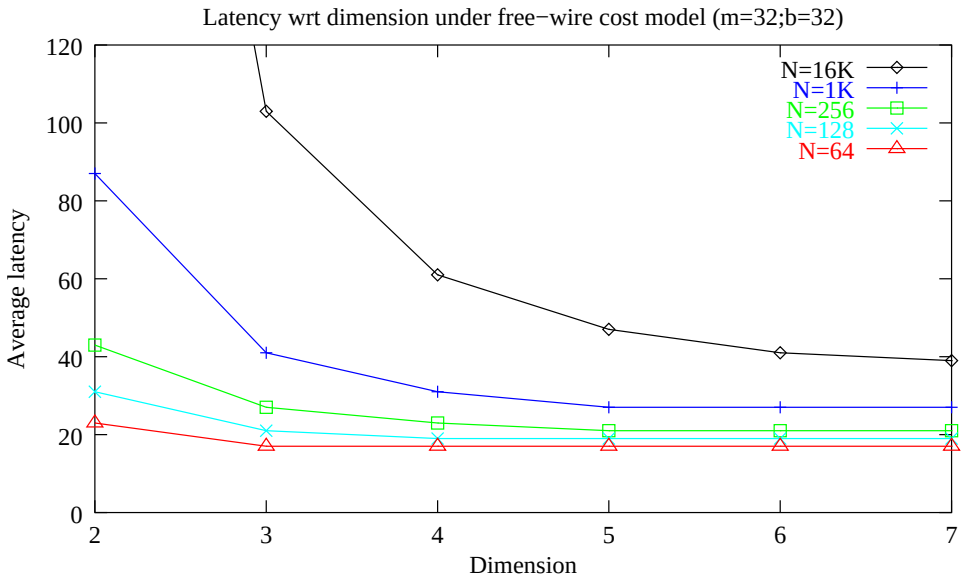


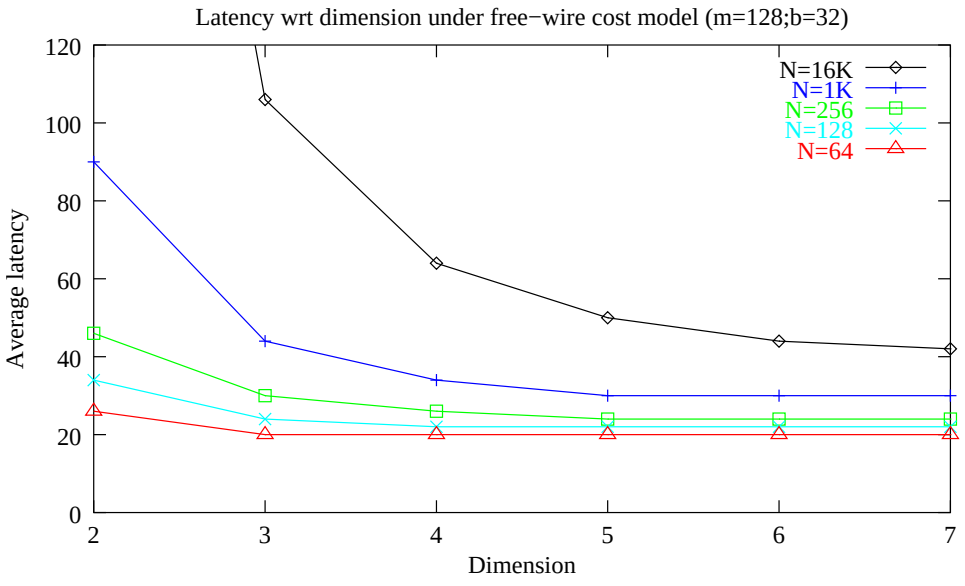
Unloaded Latency for Larger Networks and Local Traffic



Unloaded Latency under a Free-Wire Cost Model

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

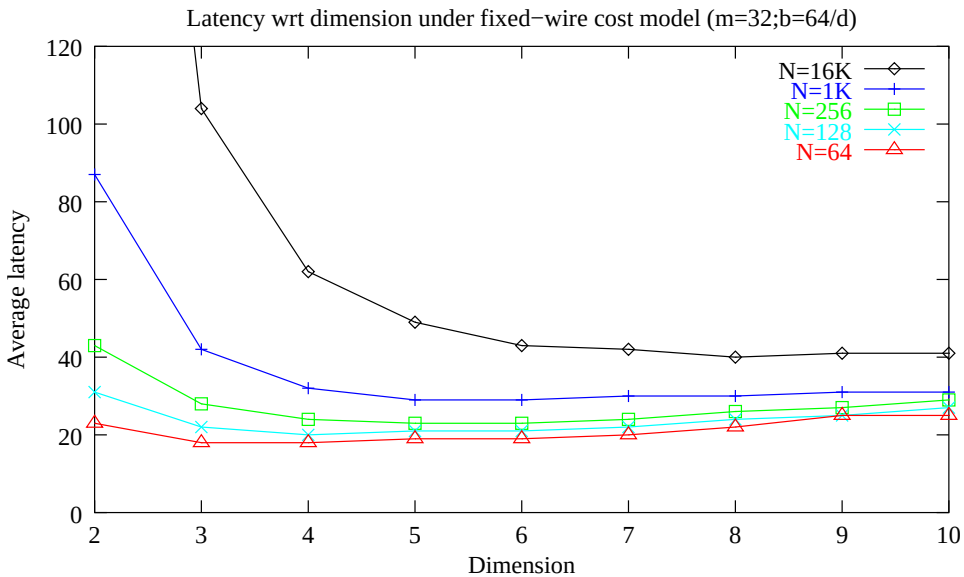


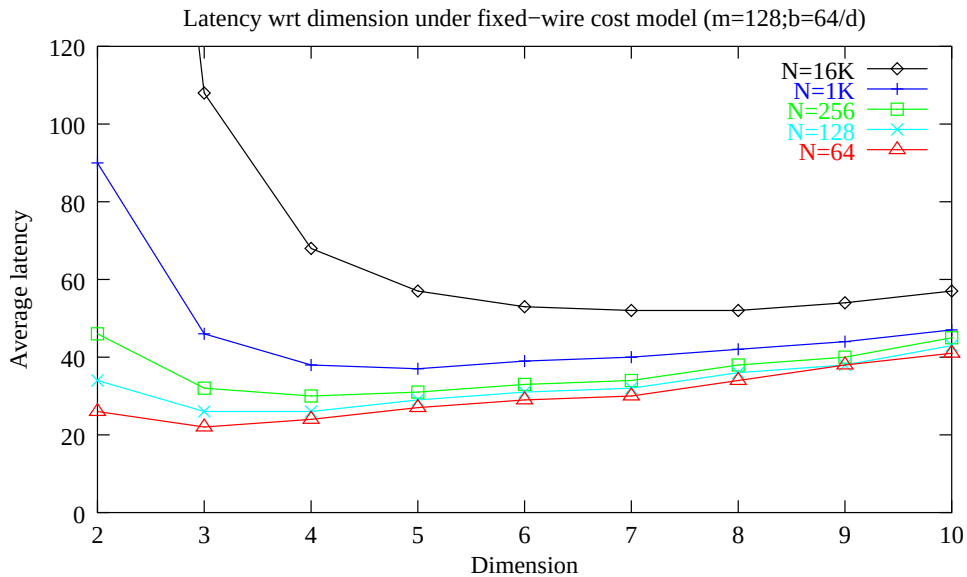


Unloaded Latency under a Fixed-Wire Cost Models

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

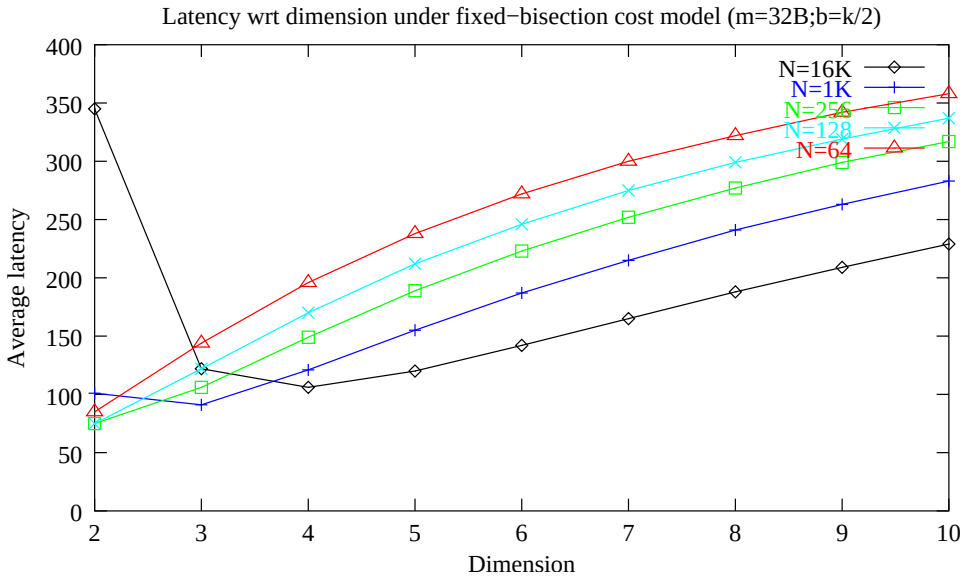


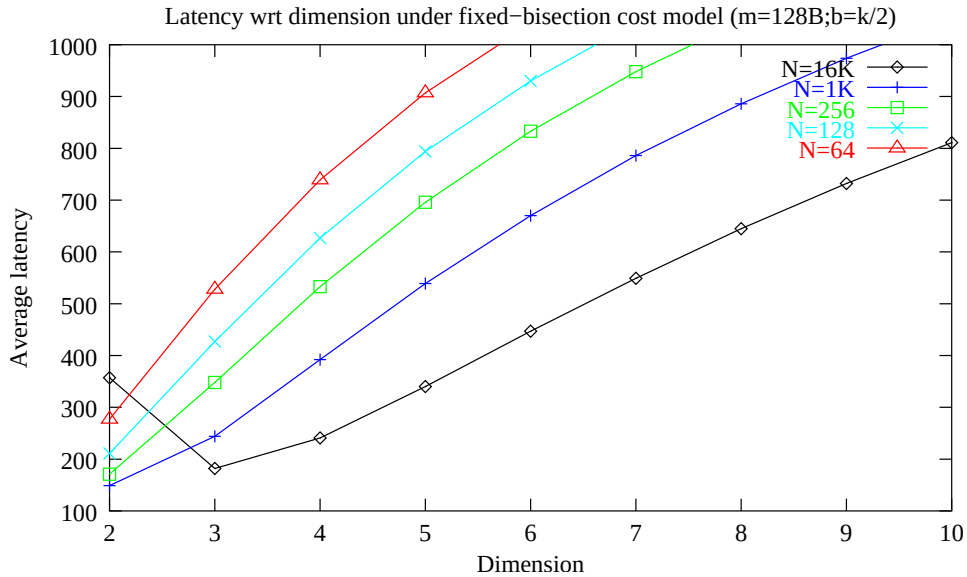


Unloaded Latency under a Fixed-Bisection Cost Models

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

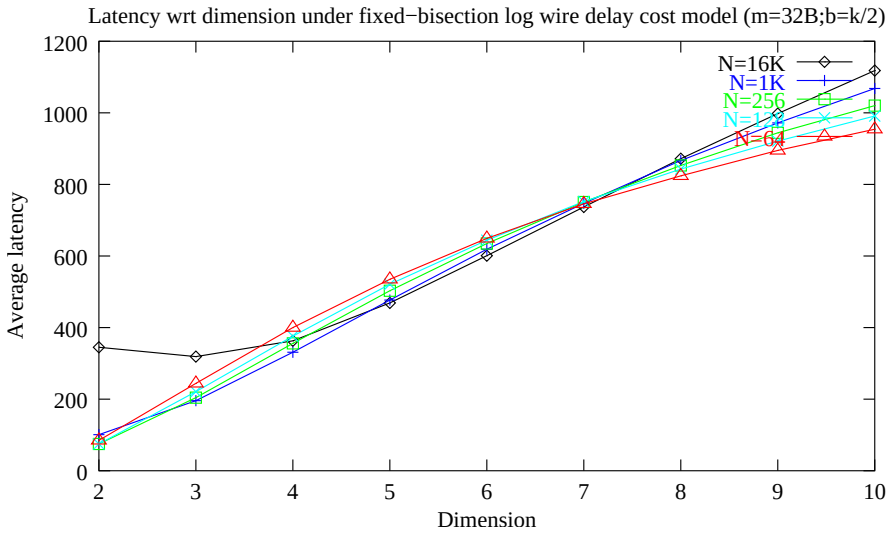


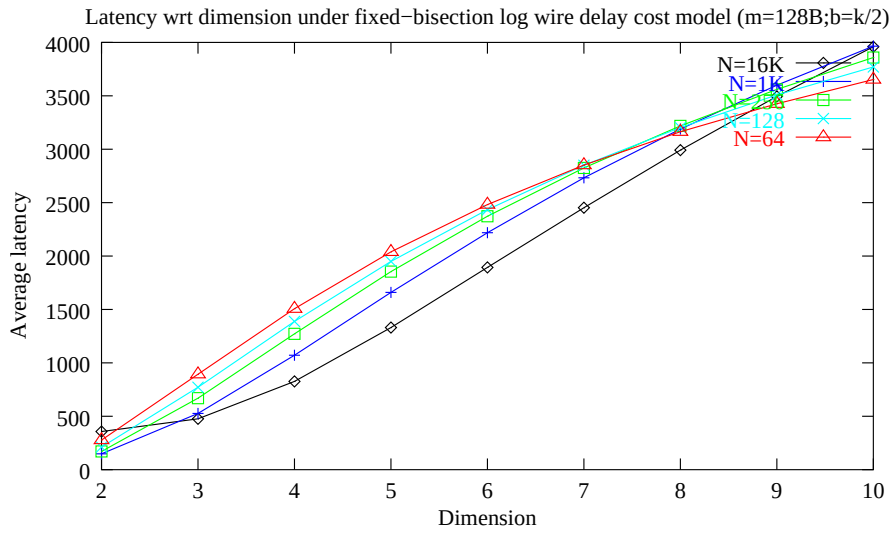


Unloaded Latency under a Logarithmic Wire Delay Cost Models

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: $l = k^{\frac{n}{2}-1}$

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

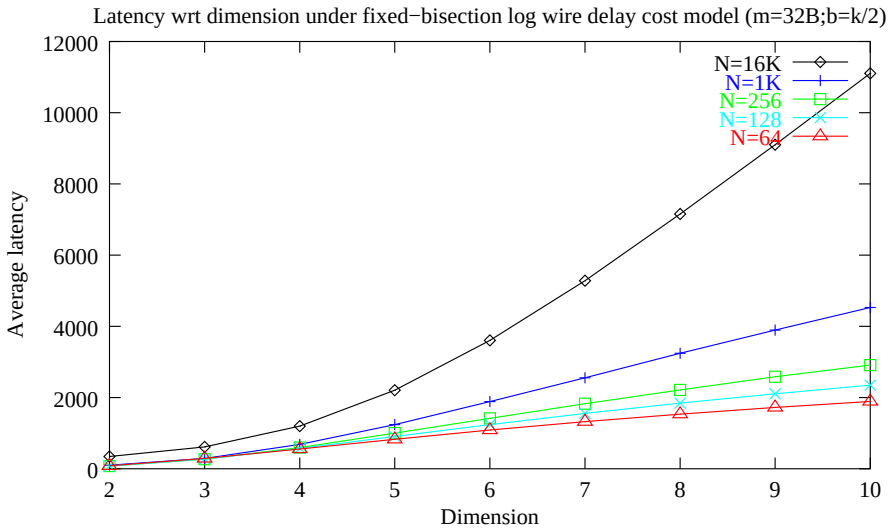


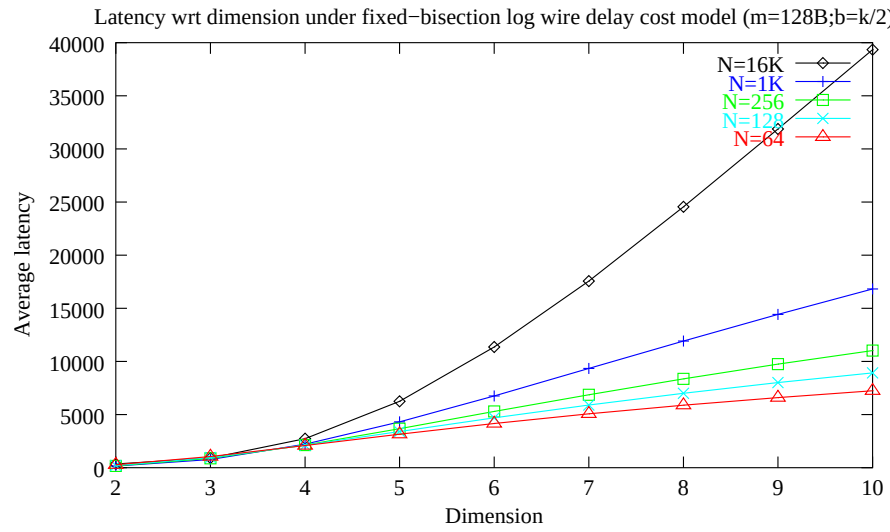


Unloaded Latency under a Linear Wire Delay Cost Models

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: $l = k^{\frac{n}{2}-1}$

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





We Take a 2-d Mesh!

Latency under Load

Assumptions [Agarwal, 1991]:

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

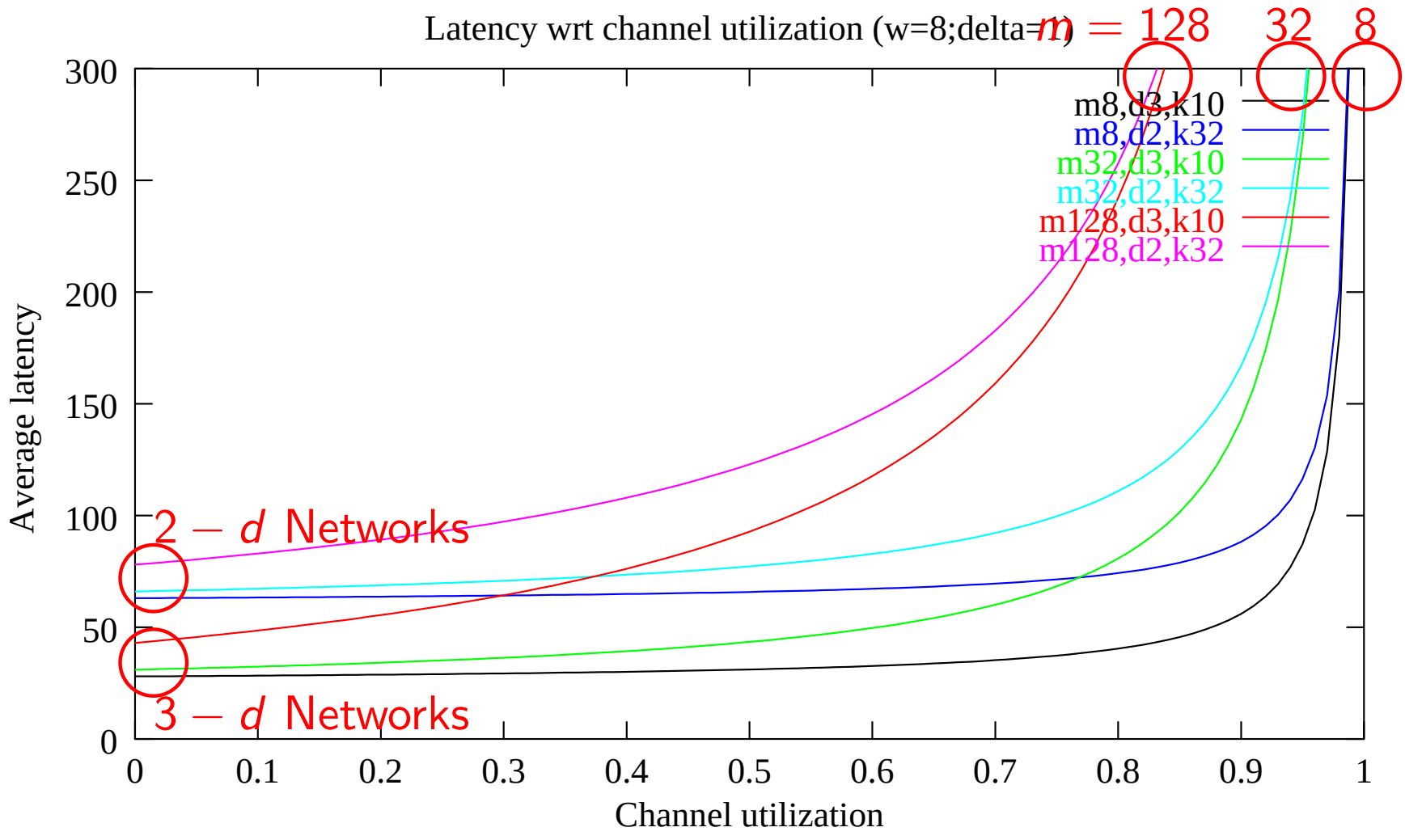
Latency under Load - cont'd

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

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

m $\cdots$ message size
w $\cdots$ bitwidth of link
ρ $\cdots$ aggregate channel utilization
h_k $\cdots$ average distance in each dimension
Δ $\cdots$ switching time in cycles

Latency vs Channel Load



So Topology does not Matter?

Routing

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

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

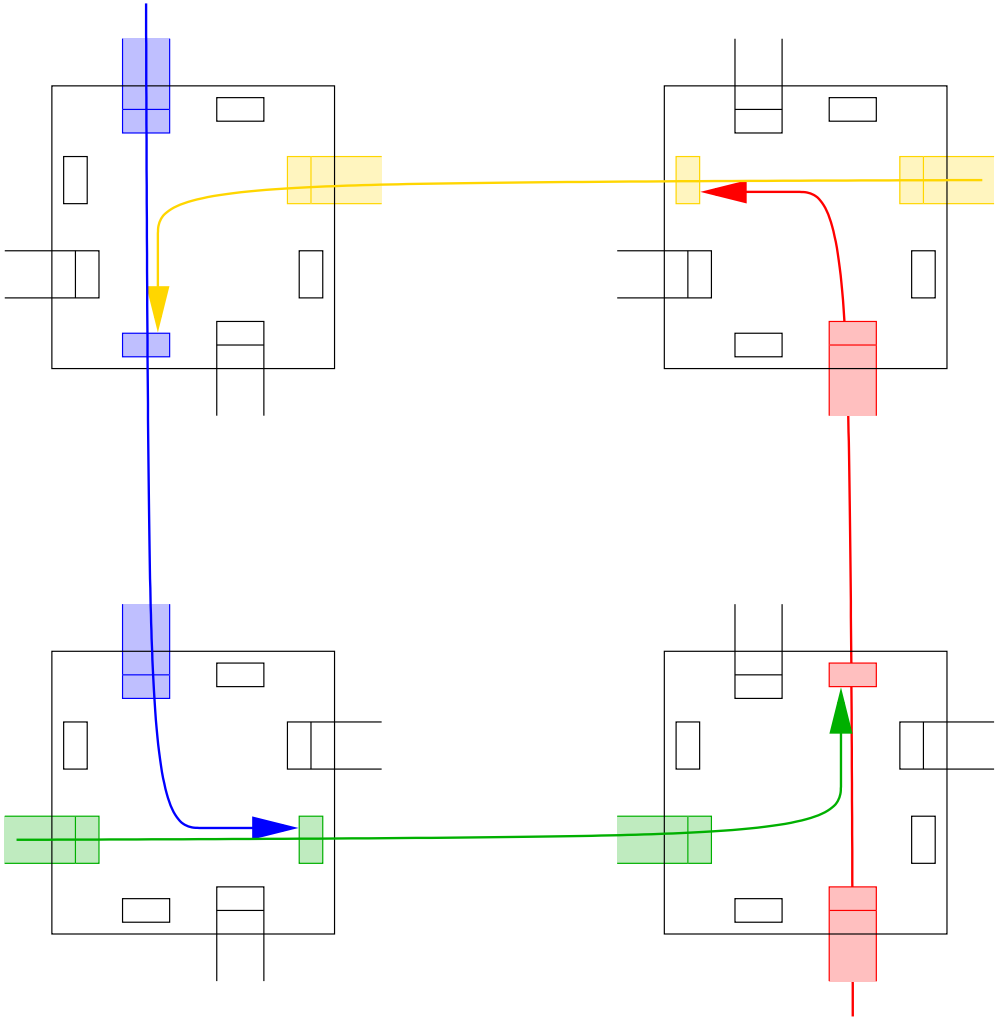
Minimal routing allows only shortest paths while non-minimal routing allows even longer paths.

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.

Deadlock



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

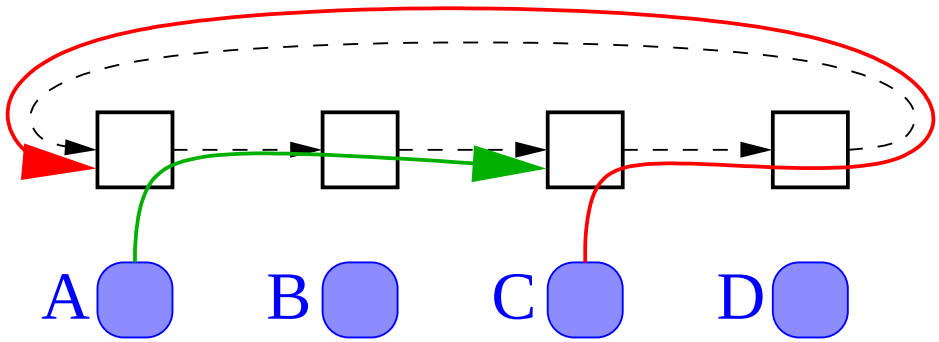
Livelock A packet keeps moving through the network but never reaches its destination.

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 $n \geq 1$.

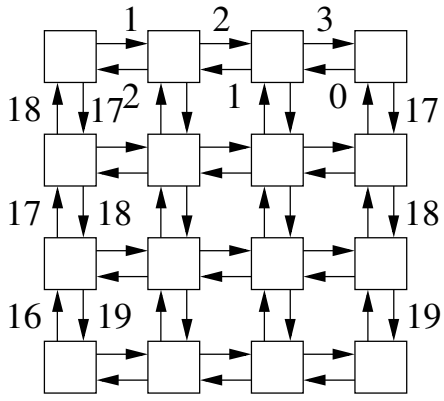
Deadlock in a 1-dimensional Torus



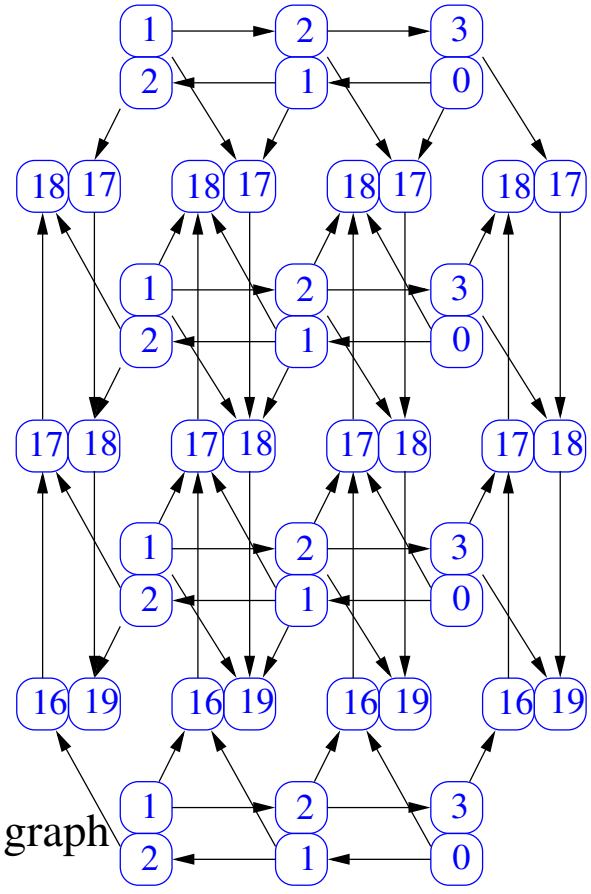
Message 1 from C−> B, 10 flits

Message 2 from A−> D, 10 flits

Channel Dependence Graph for Dimension Order Routing



4–ary 2–array



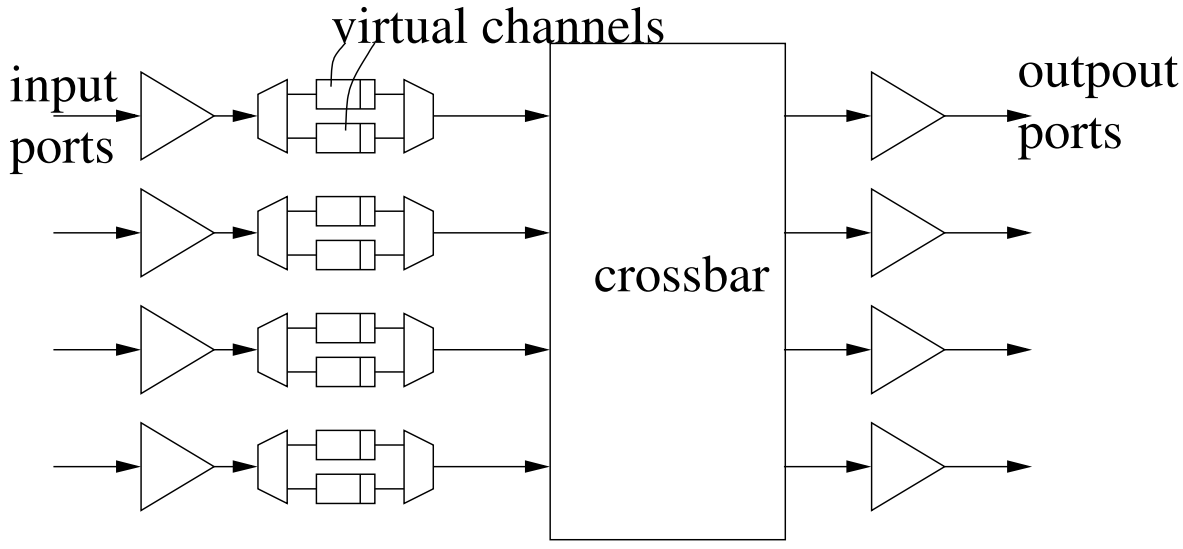
channel dependence graph

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

Deadlock-free Routing

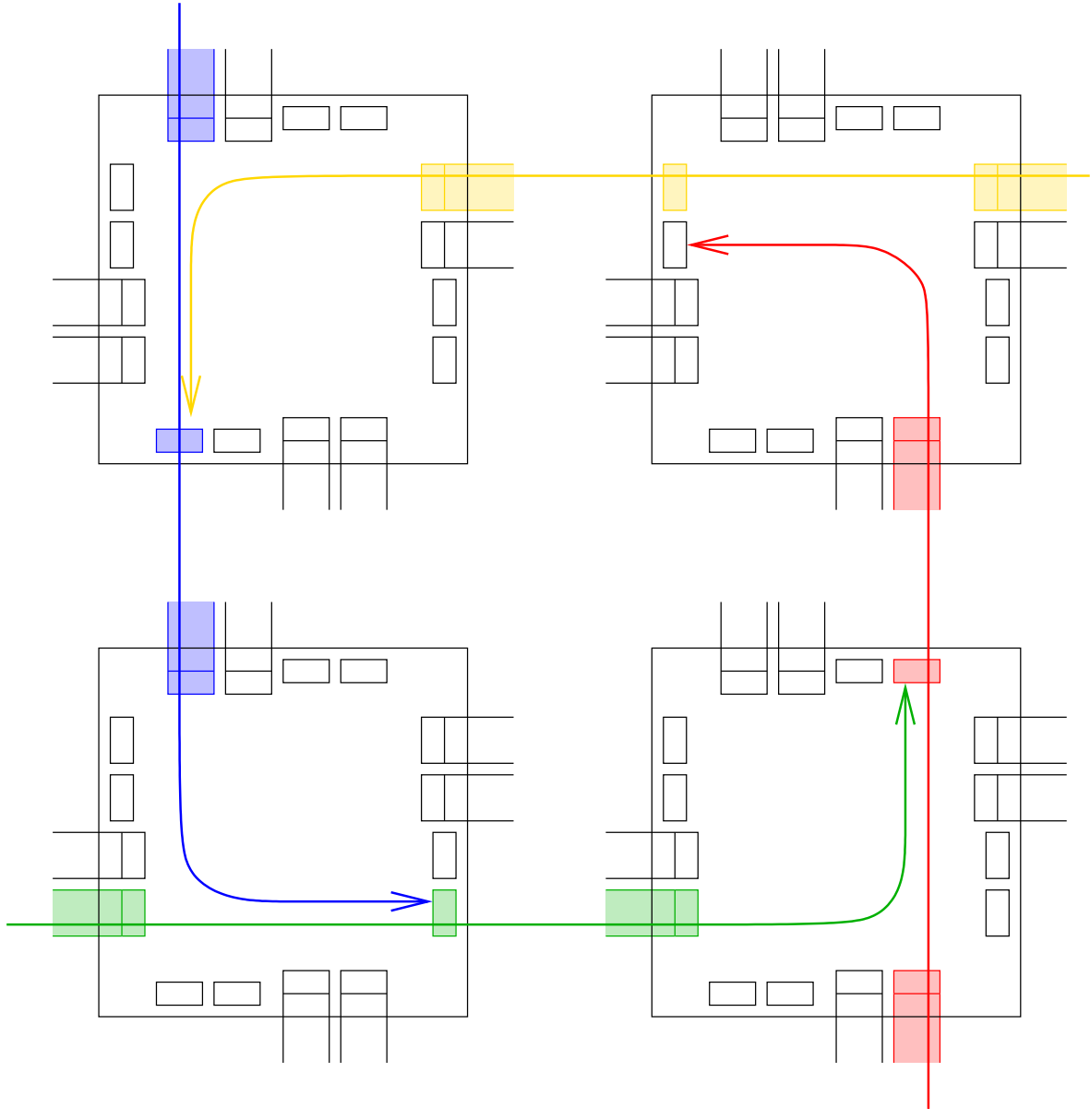
- Two main approaches:
  - Restrict the legal routes;
  - Restrict how resources are allocated;
- Number the channel cleverly
- Construct the channel dependence graph
- Prove that all legal routes follow a strictly increasing path in the channel dependence graph.

Virtual Channels



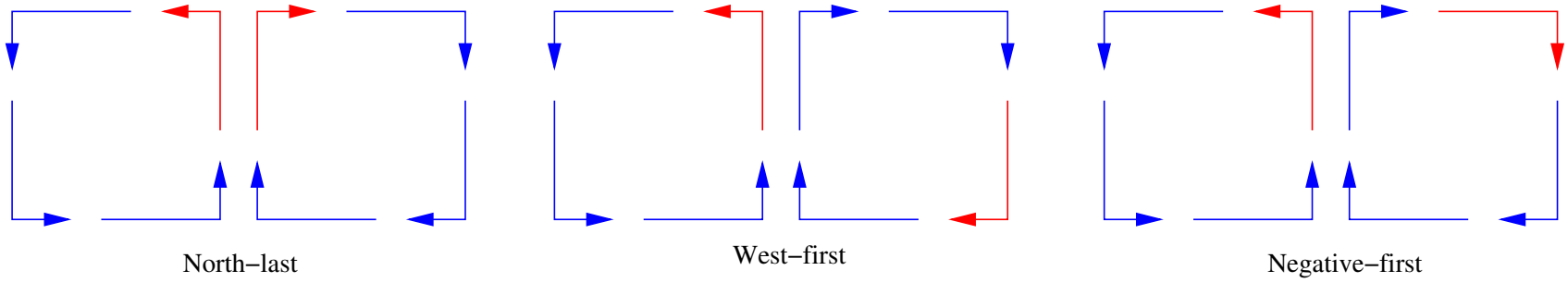
- Virtual channels can be used to break cycles in the dependence graph.
- E.g. all d-dimensional tori can be made deadlock free under dimension-order routing by assigning all wrap-around paths to a different virtual channel than other links.

Virtual Channels and Deadlocks



Turn-Model Routing

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



- For $2-d$ meshes we have three minimal routing restriction schemes:
  - **North-last**
  - **West-first**
  - **Negative-first**
- Allow complex, non-minimal adaptive routes.
- Unidirectional k-ary d-cubes still need virtual channels.

Adaptive Routing

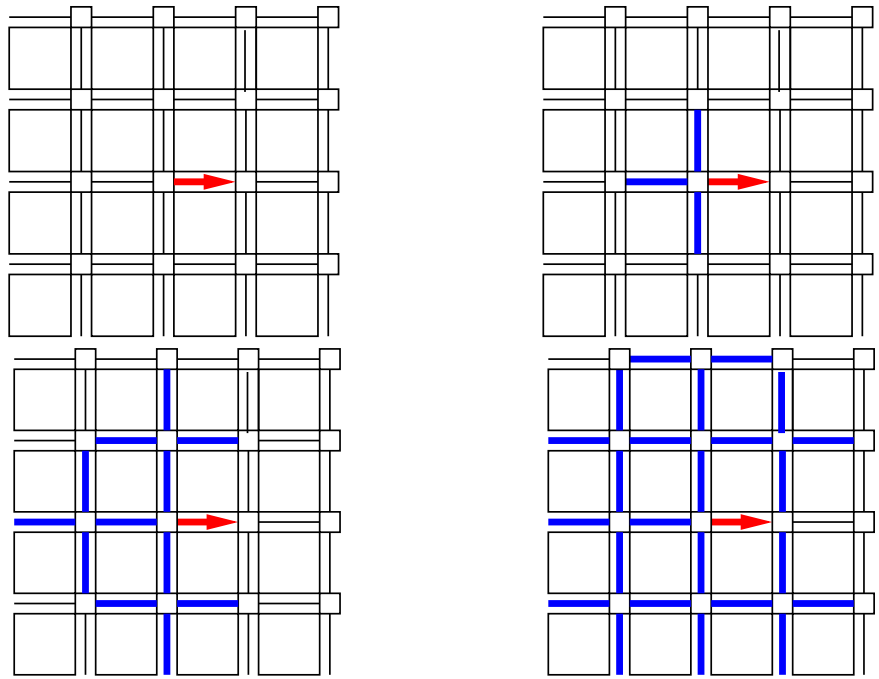
- The switch makes routing decisions based on the load.
- Fully adaptive routing allows all shortest paths.
- Partial adaptive routing allows only a subset of the shortest path.
- Non-minimal adaptive routing allows also non-minimal paths.
- Hot-potato routing is non-minimal adaptive routing without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

Quality of Service

- Best Effort (BE)
  - Optimization of the average case
  - Loose or non-existent worst case bounds
  - Cost effective use of resources
- Guaranteed Service (GS)
  - Maximum delay
  - Minimum bandwidth
  - Maximum Jitter
  - Requires additional resources

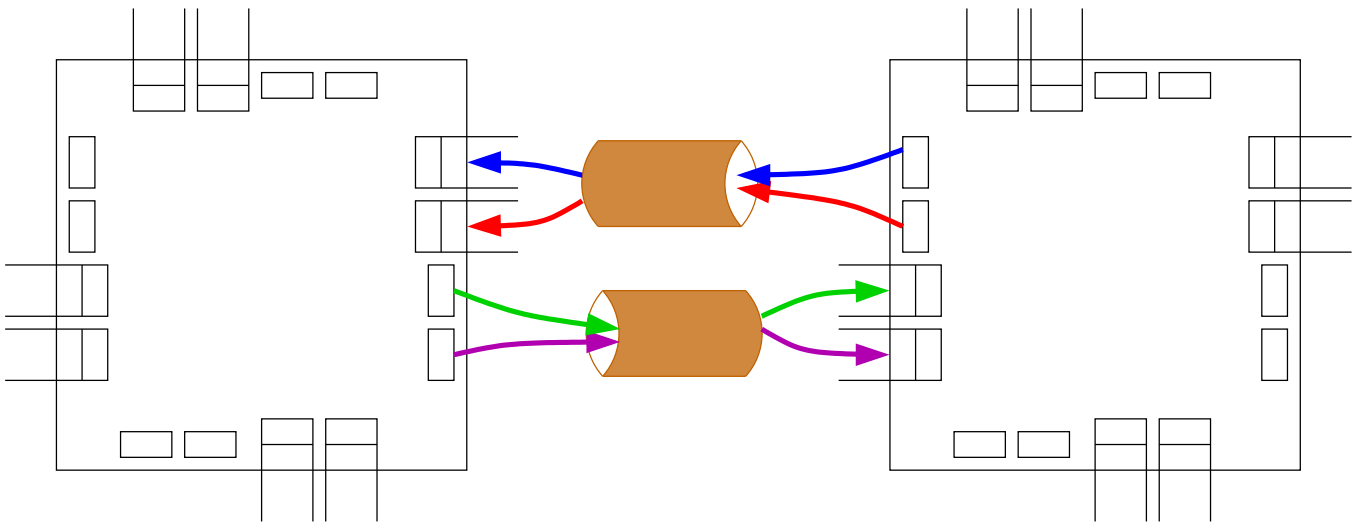
Tree Saturation

Hot spots build up a congestion tree due to back pressure



Non-interfering Networks

To isolate two traffic classes **A** and **B** there cannot be any resource shared between **A** and **B** that can be held and indefinite amount of time by **A** such that **B** cannot interrupt the usage of that resource.



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;
- Quality of Service

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

To Probe Further: Surveys

Bjerregaard, T. and Mahadevan, S. (2006).

A survey of research and practice of network-on-chip.

ACM Computing Surveys.

Ogras, U. Y. and Marculescu, R. (2008).

Analysis and optimization of prediction-based flow control in networks-on-chip.

ACM Transactions on Design Automation of Electronic Systems, 13(1).

Agarwal, A. and Iskander, C. (2009).

Survey of network on chip (noc) architectures & contributions.

Journal of engineering, Computing and Architecture, 3(1).

Fernandez-Alonso, E., Castells-Rufas, D., Joven, J., and Carrabina, J. (2012).

Survey of noc and programming models proposals for mpsoc.

International Journal of Computer Science Issues, 9(2).

To Probe Further: Surveys

Kiasari, A. E., Jantsch, A., and Lu, Z. (2013).
Mathematical formalisms for performance evaluation of networks-on-chip.
ACM Computing Surveys, 45(3).

Radetzki, M., Feng, C., Zhao, X., and Jantsch, A. (2013).
Methods for fault tolerance in networks-on-chip.
ACM Computing Surveys, 46(1):8:1–8:38.

Sahu, P. K. and Chattopadhyay, S. (2013).
A survey on application mapping strategies for network-on-chip design.
Journal of Systems Architecture, 59(1):60 – 76.

To Probe Further: Classic Papers

Agarwal, A. (1991).

Limit on interconnection performance.

IEEE Transactions on Parallel and Distributed Systems, 4(6):613–624.

Dally, W. J. (1990).

Performance analysis of k-ary d-cube interconnection networks.

IEEE Transactions on Computers, 39(6):775–785.

To Probe Further: Text books

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

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

Dally, W. J. and Towels, B. (2004).
Principles and Practices of Interconnection Networks.
Morgan Kaufman Publishers.

DeMicheli, G. and Benini, L. (2006).
Networks on Chip.
Morgan Kaufmann.

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