

Lecture 11

Thread Level Parallelism (5)

EEC 171 Parallel Architectures

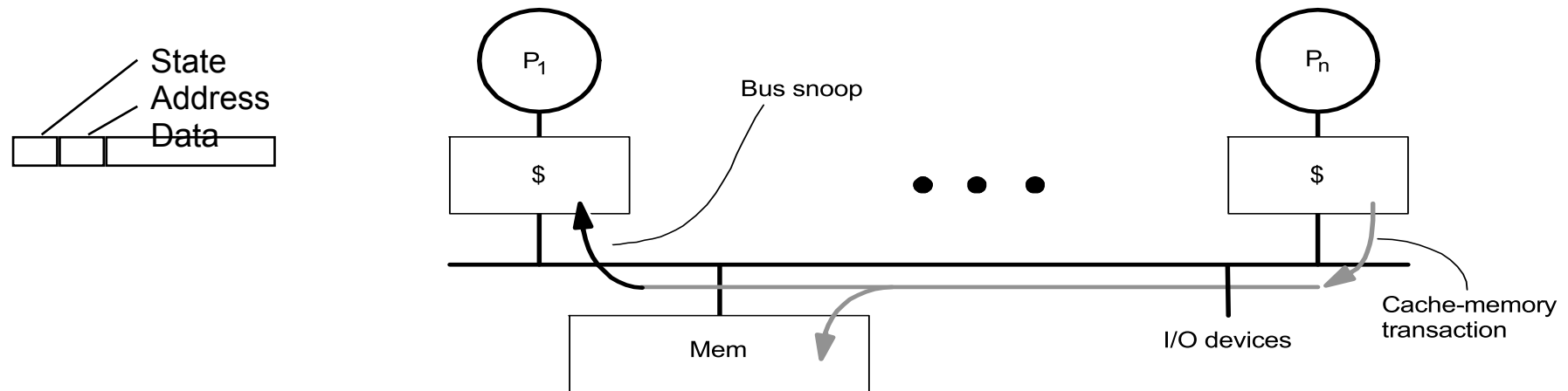
John Owens

UC Davis

Credits

- © John Owens / UC Davis 2007–17.
- Thanks to many sources for slide material: Computer Organization and Design (Patterson & Hennessy) © 2005, Computer Architecture (Hennessy & Patterson) © 2007, Inside the Machine (Jon Stokes) © 2007, © Dan Connors / University of Colorado 2007, © Kathy Yelick / UCB 2007, © Wen-Mei Hwu/David Kirk, University of Illinois 2007, © David Patterson / UCB 2003–7, © John Lazzaro / UCB 2006, © Mary Jane Irwin / Penn State 2005, © John Kubiawicz / UCB 2002, © Krste Asinovic/Arvind / MIT 2002, © Morgan Kaufmann Publishers 1998.

Snoopy Cache-Coherence Protocols



- Cache Controller “snoops” all transactions on the shared medium (bus or switch)
 - “Does this transaction concern data that I have?”
 - If so, take action to ensure coherence
 - **invalidate (my val)**, **update (my val)**, or supply (my value) (when, when, and when?)
 - depends on state of the block and the protocol
- Either get exclusive access before write via write invalidate (“write invalidate”) or update all copies on write (“write update”) (“write update” != “**update**”)

Review

- Caches contain all information on state of cached memory blocks
- Snooping cache over shared medium for smaller MP by invalidating other cached copies on write
- Sharing cached data \Rightarrow Coherence (values returned by a read), Consistency (when a written value will be returned by a read)

A Cache Coherent System Must:

- Provide set of states, state transition diagram, and actions
- Manage coherence protocol
 - (o) Determine when to invoke coherence protocol
 - (a) Find info about state of block in other caches to determine action
 - Do we need to communicate with other cached copies?
 - (b) Locate the other copies
 - (c) Communicate with those copies (invalidate/update)
- (o) is done the same way on all systems
 - state of the line is maintained in the cache
 - protocol is invoked if an “access fault” occurs on the line
- Different approaches distinguished by (a) to (c)

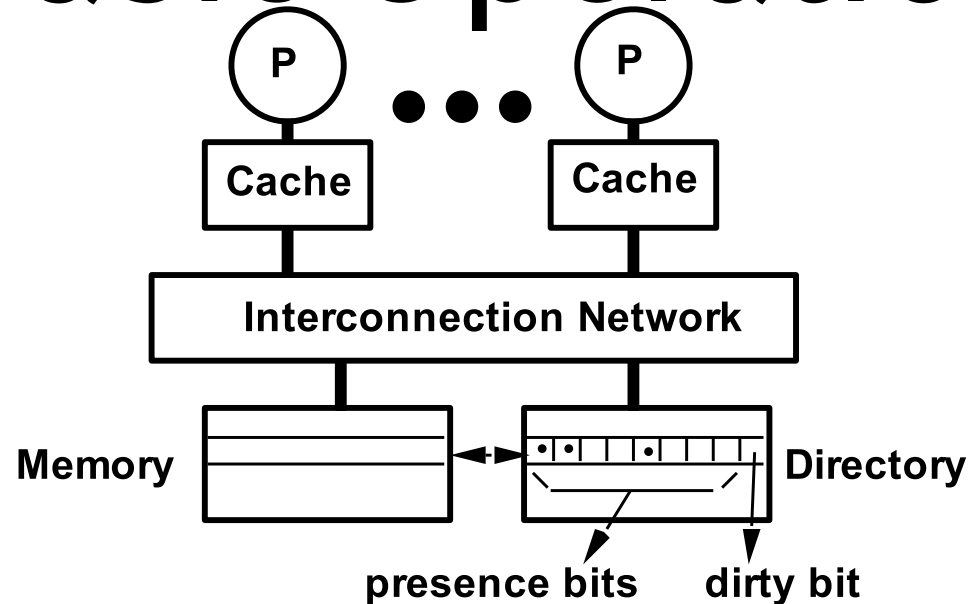
Bus-based Coherence

- All of (a), (b), (c) done through broadcast on bus
 - faulting processor sends out a “search”
 - others respond to the search probe and take necessary action
- Could do it in scalable network too
 - broadcast to all processors, and let them respond
- Conceptually simple, but **broadcast doesn't scale with p**
 - on bus, bus bandwidth doesn't scale
 - on scalable network, every fault leads to at least p network transactions
- Scalable coherence:
 - can have same cache states and state transition diagram
 - different mechanisms to manage protocol

Scalable Approach: Directories

- Every memory block has associated directory information
- keeps track of copies of cached blocks and their states
- Directory is distributed among nodes
- on a miss, find directory entry, look it up, and communicate only with the nodes that have copies if necessary
- in scalable networks, communication with directory and copies is through network transactions
- Many alternatives for organizing directory information

Basic Operation of Directory



- k processors.
- With each cache-block in memory:
 k presence-bits, 1 dirty-bit
- With each cache-block in cache:
1 valid bit, and 1 dirty (owner) bit
- Read from main memory by processor i :
 - If dirty-bit OFF then { read from main memory; turn $p[i]$ ON; }
 - If dirty-bit ON then { recall line from dirty proc (cache state to shared); update memory; turn dirty-bit OFF; turn $p[i]$ ON; supply recalled data to i ; }
- Write (miss) to main memory by processor i :
 - If dirty-bit OFF then { supply data to i ; send invalidations to all caches that have the block; turn dirty-bit ON; turn $p[i]$ ON; ... }
 - • ...

Directory Protocol

- Similar to Snoopy Protocol: Three states
 - Shared: ≥ 1 processors have data, memory up-to-date
 - Uncached (no processor has it; not valid in any cache)
 - Exclusive: 1 processor (owner) has data; memory out-of-date
- In addition to cache state, must track which processors have data when in the shared state (usually bit vector, 1 if processor has copy)
 - Why isn't this necessary for snoopy protocol?
- Keep it simple:
 - Writes to non-exclusive data \Rightarrow write miss
 - Processor blocks until access completes
 - Assume messages received and acted upon in order sent

Directory Protocol

- No bus and don't want to broadcast:
 - interconnect no longer single arbitration point
 - all messages have explicit responses
- Terms: typically 3 processors involved
 - Local node where a request originates
 - Home node where the memory location of an address resides
 - Remote node has a copy of a cache block, whether exclusive or shared
- Example messages on next slide:
P = processor number, A = address

Directory Protocol Messages (fig 5.21 in book)

Message type	Source	Destination	Msg Content
--------------	--------	-------------	-------------

Read miss	Local cache	Home directory	P, A
-----------	-------------	----------------	------

- Processor P reads data at address A;
make P a read sharer and request data

Write miss	Local cache	Home directory	P, A
------------	-------------	----------------	------

- Processor P has a write miss at address A;
make P the exclusive owner and request data

Invalidate	Home directory	Remote caches	A
------------	----------------	---------------	---

- Invalidate a shared copy at address A

Fetch	Home directory	Remote cache	A
-------	----------------	--------------	---

- Fetch the block at address A and send it to its home directory;
change the state of A in the remote cache to shared

Fetch/Invalidate	Home directory	Remote cache	A
------------------	----------------	--------------	---

- Fetch the block at address A and send it to its home directory;
invalidate the block in the cache

Data value reply	Home directory	Local cache	Data
------------------	----------------	-------------	------

- Return a data value from the home memory (read miss response)

Data write back	Remote cache	Home directory	A, Data
-----------------	--------------	----------------	---------

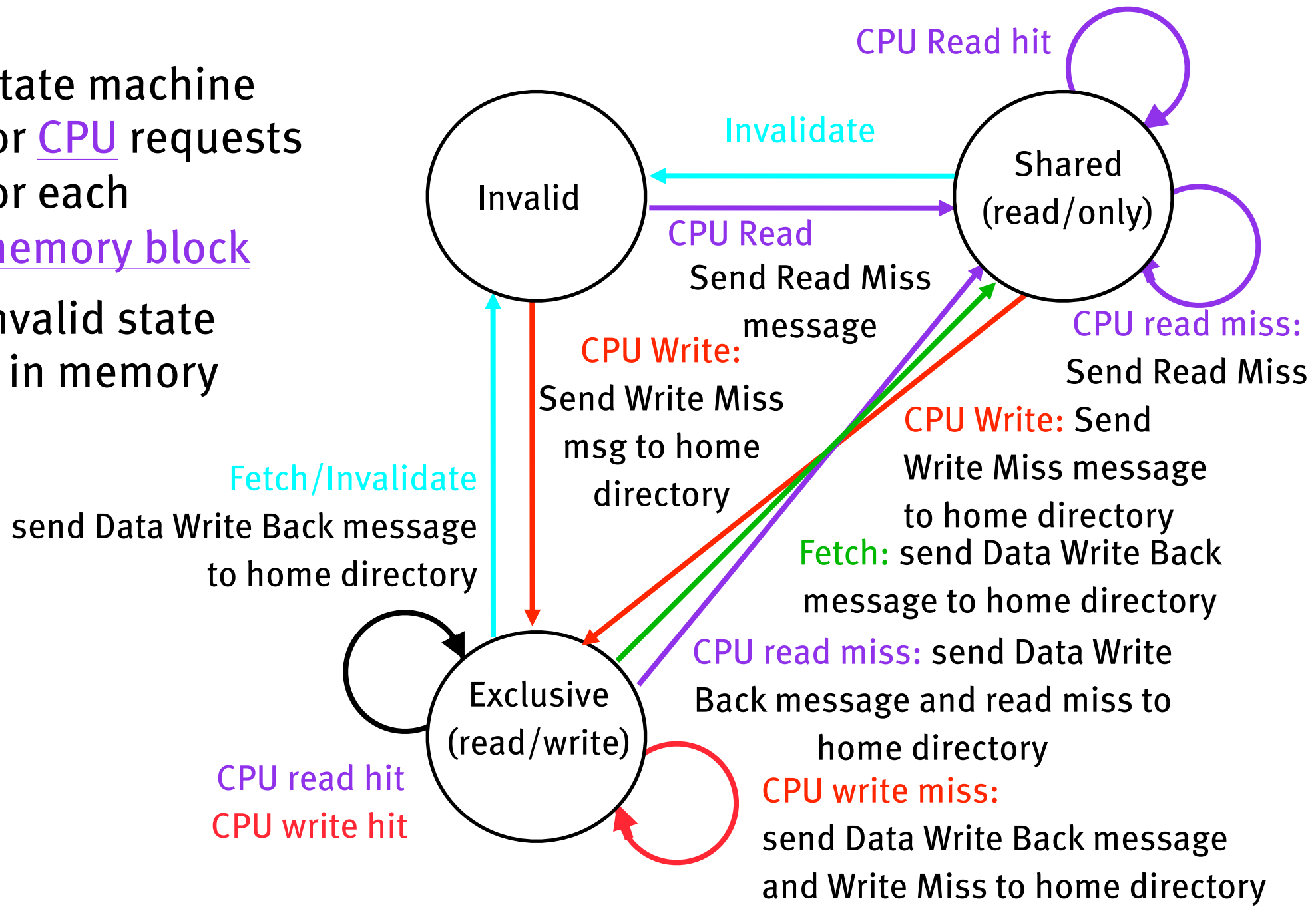
- Write back a data value for address A (invalidate response)

State Transition Diagram for One Cache Block in Directory Based System

- States identical to snoopy case; transactions very similar
- Cache block generates read miss & write miss message to home directory
- Transitions in state diagram caused by read misses, write misses, invalidates, data fetch requests
- In snooping, write misses were broadcast on the bus for snooping \Rightarrow now explicit invalidate & data fetch requests
- One-to-one communication, not broadcast
- Note: on a write (of a word), a cache block is bigger (than a word), so need to read the full cache block

CPU—Cache State Machine

- State machine for CPU requests for each memory block
- Invalid state if in memory

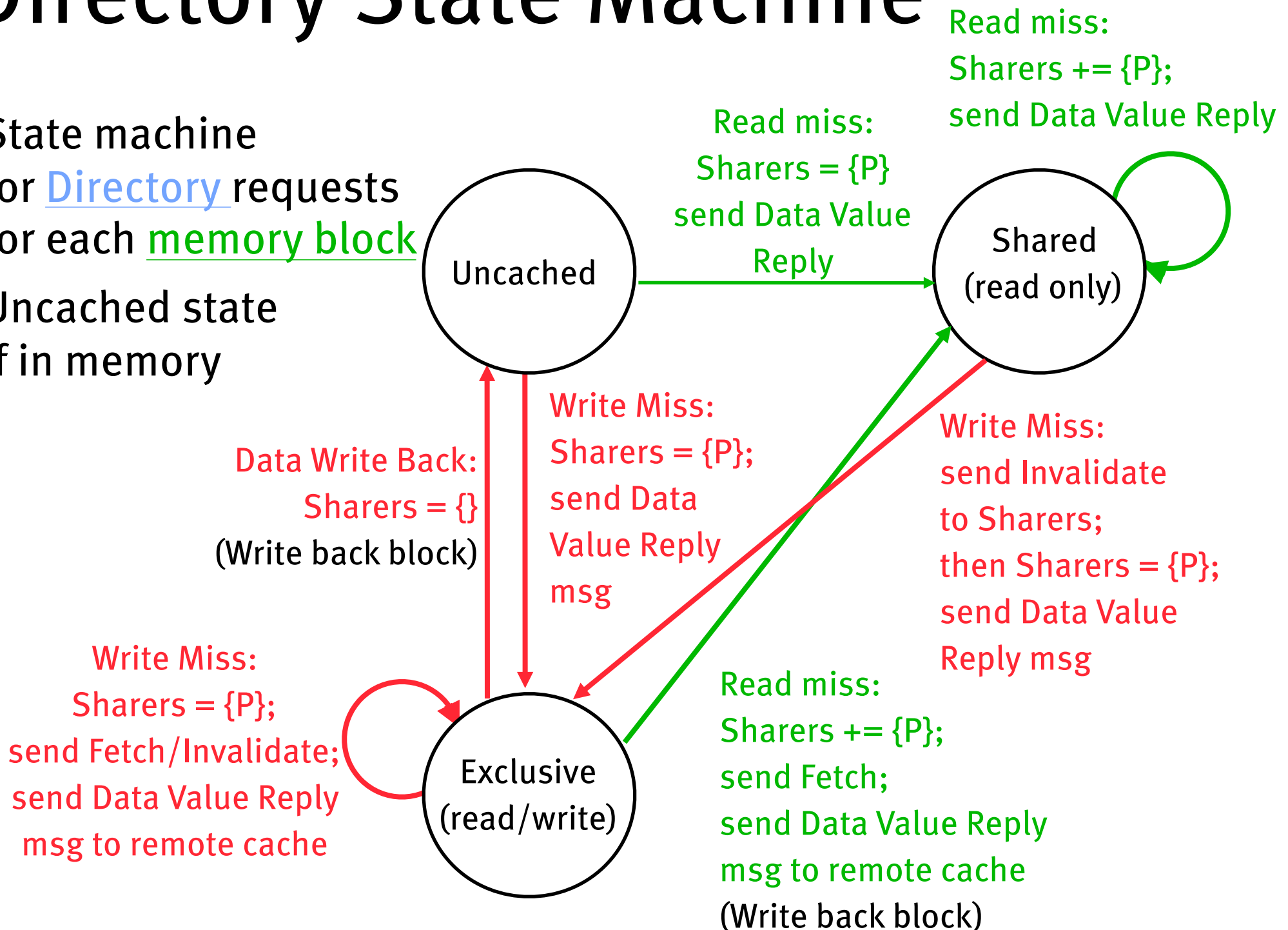


State Transition Diagram for Directory

- Same states & structure as the transition diagram for an individual cache
- 2 actions: update of directory state & send messages to satisfy requests
- Tracks all copies of memory block
- Also indicates an action that updates the sharing set, Sharers, as well as sending a message

Directory State Machine

- State machine for Directory requests for each memory block
- Uncached state if in memory



Example Directory Protocol

- Message sent to directory causes two actions:
 - Update the directory
 - More messages to satisfy request
- Block is in Uncached state: the copy in memory is the current value; only possible requests for that block are:
 - Read miss: requesting processor sent data from memory & requestor made only sharing node; state of block made Shared.
 - Write miss: requesting processor is sent the value & becomes the Sharing node. The block is made Exclusive to indicate that the only valid copy is cached. Sharers indicates the identity of the owner.

Example Directory Protocol

- Message sent to directory causes two actions:
 - Update the directory
 - More messages to satisfy request
- Block is Shared \Rightarrow the memory value is up-to-date:
 - Read miss: requesting processor is sent back the data from memory & requesting processor is added to the sharing set.
 - Write miss: requesting processor is sent the value. All processors in the set Sharers are sent invalidate messages, & Sharers is set to identity of requesting processor. The state of the block is made Exclusive.

Example Directory Protocol

- Block is Exclusive: current value of the block is held in the cache of the processor identified by the set Sharers (the owner) \Rightarrow three possible directory requests:
 - Read miss: owner processor sent data fetch message, causing state of block in owner's cache to transition to Shared and causes owner to send data to directory, where it is written to memory & sent back to requesting processor.
Identity of requesting processor is added to set Sharers, which still contains the identity of the processor that was the owner (since it still has a readable copy). State is shared.

Example Directory Protocol

- Block is Exclusive: current value of the block is held in the cache of the processor identified by the set Sharers (the owner) \Rightarrow three possible directory requests:
 - Data write-back: owner processor is replacing the block and hence must write it back, making memory copy up-to-date (the home directory essentially becomes the owner), the block is now Uncached, and the Sharer set is empty.

Example Directory Protocol

- Block is Exclusive: current value of the block is held in the cache of the processor identified by the set Sharers (the owner) \Rightarrow three possible directory requests:
 - Write miss: block has a new owner. A message is sent to old owner causing the cache to send the value of the block to the directory from which it is sent to the requesting processor, which becomes the new owner. Sharers is set to identity of new owner, and state of block is made Exclusive.

Example

Processor 1 Processor 2 Interconnect Directory Memory

	P1			P2			Bus			Directory			Memor	
step	State	Addr	Value	State	Addr	Value	Action	Proc.	Addr	Value	Addr	State	{Procs}	Value
P1: Write 10 to A1														
P1: Read A1														
P2: Read A1														
P2: Write 20 to A1														
P2: Write 40 to A2														

A1 and A2 map to the same cache block

Example

	Processor 1			Processor 2			Interconnect			Directory			Memory	
	P1			P2			Bus				Directory			Memory
step	State	Addr	Value	State	Addr	Value	Action	Proc.	Addr	Value	Addr	State	{Procs}	Value
P1: Write 10 to A1							WrMs	P1	A1		A1	Ex	{P1}	
	Excl.	A1	10				DaRp	P1	A1	0				
P1: Read A1														
P2: Read A1														
P2: Write 20 to A1														
P2: Write 40 to A2														

A1 and A2 map to the same cache block

Example

Processor 1 Processor 2 Interconnect Directory Memory

	P1			P2			Bus			Directory				Memor
step	State	Addr	Value	State	Addr	Value	Action	Proc.	Addr	Value	Addr	State	{Procs}	Value
P1: Write 10 to A1							WrMs	P1	A1		A1	Ex	{P1}	
	Excl.	A1	10				DaRp	P1	A1	0				
P1: Read A1	Excl.	A1	10											
P2: Read A1														
P2: Write 20 to A1														
P2: Write 40 to A2														

A1 and A2 map to the same cache block

Example

	Processor 1			Processor 2			Interconnect			Directory			Memory	
	P1			P2			Bus				Directory			Memory
step	State	Addr	Value	State	Addr	Value	Action	Proc.	Addr	Value	Addr	State	{Procs}	Value
P1: Write 10 to A1							WrMs	P1	A1		A1	Ex	{P1}	
	Excl.	A1	10				DaRp	P1	A1	0				
P1: Read A1	Excl.	A1	10											
P2: Read A1				Shar.	A1		RdMs	P2	A1					
	Shar.	A1	10				Ftch	P1	A1	10				10
				Shar.	A1	10	DaRp	P2	A1	10	A1	Shar.	P1,P2}	10
P2: Write 20 to A1														10
														10
P2: Write 40 to A2														10

Directory initiates Fetch, which
causes write back from P1

A1 and A2 map to the same cache block

Example

	Processor 1			Processor 2			Interconnect			Directory			Memory	
	P1			P2			Bus				Directory			Memory
step	State	Addr	Value	State	Addr	Value	Action	Proc.	Addr	Value	Addr	State	{Procs}	Value
P1: Write 10 to A1							WrMs	P1	A1		A1	Ex	{P1}	
	Excl.	A1	10				DaRp	P1	A1	0				
P1: Read A1	Excl.	A1	10											
P2: Read A1				Shar.	A1		RdMs	P2	A1					
	Shar.	A1	10				Ftch	P1	A1	10				10
				Shar.	A1	10	DaRp	P2	A1	10	A1	Shar.	P1,P2}	10
P2: Write 20 to A1				Excl.	A1	20	WrMs	P2	A1					10
	Inv.						Inval.	P1	A1		A1	Excl.	{P2}	10
P2: Write 40 to A2														10

A1 and A2 map to the same cache block

Example

	Processor 1			Processor 2			Interconnect			Directory			Memory	
	P1			P2			Bus				Directory		Memory	
step	State	Addr	Value	State	Addr	Value	Action	Proc.	Addr	Value	Addr	State	{Procs}	Value
P1: Write 10 to A1							WrMs	P1	A1		A1	Ex	{P1}	
	Excl.	A1	10				DaRp	P1	A1	0				
P1: Read A1	Excl.	A1	10											
P2: Read A1				Shar.	A1		RdMs	P2	A1					
Play goatsim	Shar.	A1	10				Ftch	P1	A1	10				10
				Shar.	A1	10	DaRp	P2	A1	10	A1	Shar.	P1,P2}	10
P2: Write 20 to A1				Excl.	A1	20	WrMs	P2	A1					10
	Inv.						Inval.	P1	A1		A1	Excl.	{P2}	10
P2: Write 40 to A2							WrMs	P2	A2		A2	Excl.	{P2}	0
							WrBk	P2	A1	20	A1	Jnca.	{}	20
				Excl.	A2	40	DaRp	P2	A2	0	A2	Excl.	{P2}	0

A1 and A2 map to the same cache block
(but different memory block addresses $A1 \neq A2$)

Example

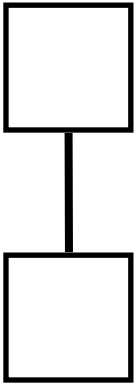
	Processor 1			Processor 2			Interconnect			Directory			Memory	
	P1			P2			Bus				Directory			Memory
step	State	Addr	Value	State	Addr	Value	Action	Proc.	Addr	Value	Addr	State	{Procs}	Value
P1: Write 10 to A1							WrMs	P1	A1		A1	Ex	{P1}	
	Excl.	A1	10				DaRp	P1	A1	0				
P1: Read A1	Excl.	A1	10											
P2: Read A1				Shar.	A1		RdMs	P2	A1					
	Shar.	A1	10				Ftch	P1	A1	10				10
				Shar.	A1	10	DaRp	P2	A1	10	A1	Shar.	P1,P2}	10
P2: Write 20 to A1				Excl.	A1	20	WrMs	P2	A1					10
	Inv.						Inval.	P1	A1		A1	Excl.	{P2}	10
P2: Write 40 to A2							WrMs	P2	A2		A2	Excl.	{P2}	0
							WrBk	P2	A1	20	A1	Jnca.	{}	20
				Excl.	A2	40	DaRp	P2	A2	0	A2	Excl.	{P2}	0

Study question: Why does the directory respond to a write miss with a data reply here and here but not here?

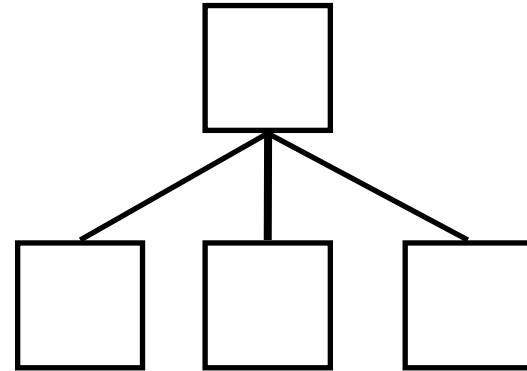
And in Conclusion ...

- Snooping and Directory Protocols similar; bus makes snooping easier because of broadcast (snooping -> uniform memory access)
- Directory has extra data structure to keep track of state of all cache blocks
- Distributing directory:
 - scalable shared address multiprocessor
 - Cache coherent, Non uniform memory access

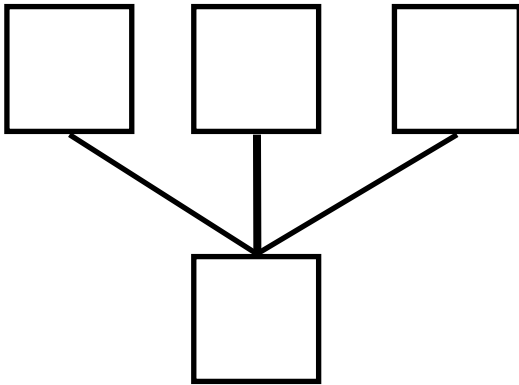
Types of Communication



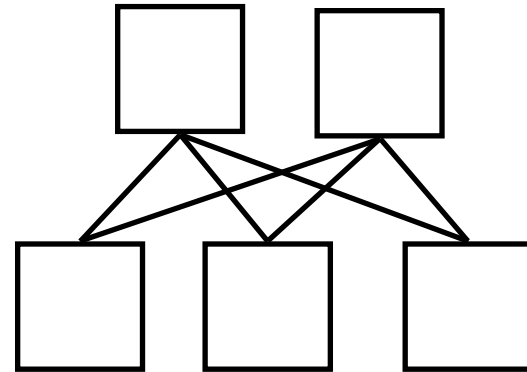
1 to 1



1 to N



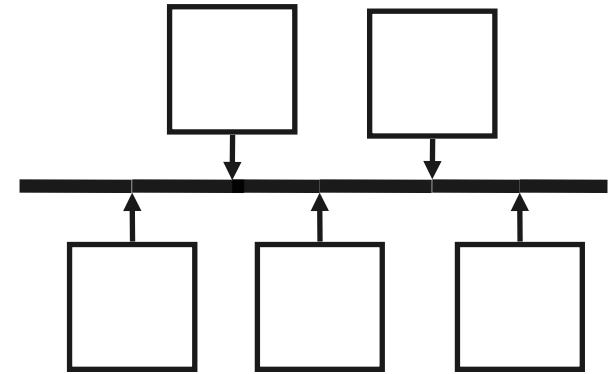
N to 1



N to M

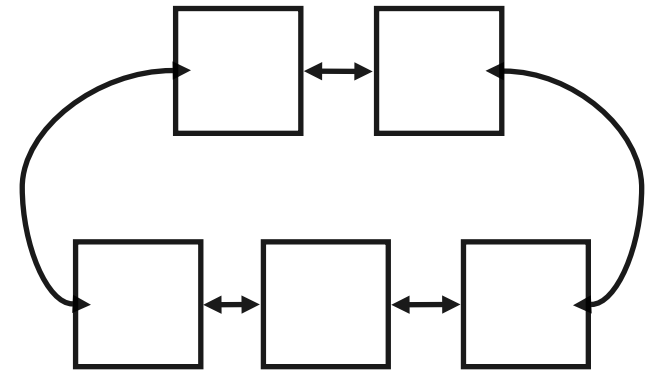
Types of Communication: Bus

- $O(n)$ per-port cost, $O(n^2)$ total
 - Not particularly scalable
 - Each server gets bus $1/n$ th of time
 - Bus width should be $O(n)$
 - Electrically difficult to scale — speed decreases with more sources/destinations
- Broadcast same cost as 1-to-1
- Can provide serialization
- In practice: target maximum degree of parallelism
 - Non-scalable bus
 - When bus not fully used, wasted cost



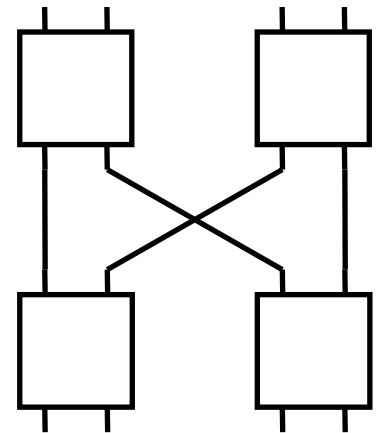
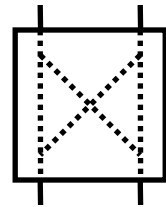
Types of Communication: Ring

- Interconnects multiple units with series of 1-to-1 neighbor communications
- $O(1)$ per-port cost, $O(n)$ total
 - Good scaling characteristic
 - Well suited for broadcast
 - Great for nearest-neighbor



Types of Communication: MIN

- Intermediate switch elements
 - Butterfly, omega, fat trees ...
 - Point-to-point communication
- $O(n \log n)$ total cost



Efficiency—Bandwidth per Cost

	Neighbor	1-to-1	Broadcast
bus	$1/n$	$1/n$	1
ring	1	$2/n$	1
MIN	$1/\log n$	$1/\log n$	$1/\log n$

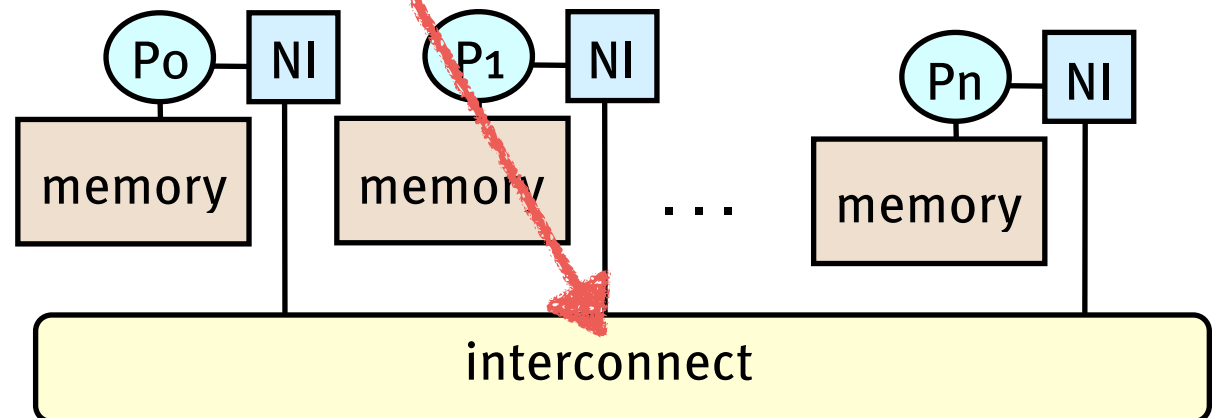
- Every node wants to XXX to another node—what's the bandwidth per cost? (What fraction of the resources are we using?)

Machine Model 2a: Distributed Memory

- Cray T3E, IBM SP2
- PC Clusters (Berkeley NOW, Beowulf)
- IBM SP-3, Millennium, CITRIS are distributed memory machines, but the nodes are SMPs.

What is in this “Interconnect”?

- Each processor has its own memory and cache but cannot directly access another processor's memory.
- Each “node” has a Network Interface (NI) for all communication and synchronization.



Interconnection Networks

Computer Architecture: A Quantitative Approach

4th Edition, Appendix E

Timothy Mark Pinkston
University of Southern California
<http://ceng.usc.edu/smart/slides/appendixE.html>

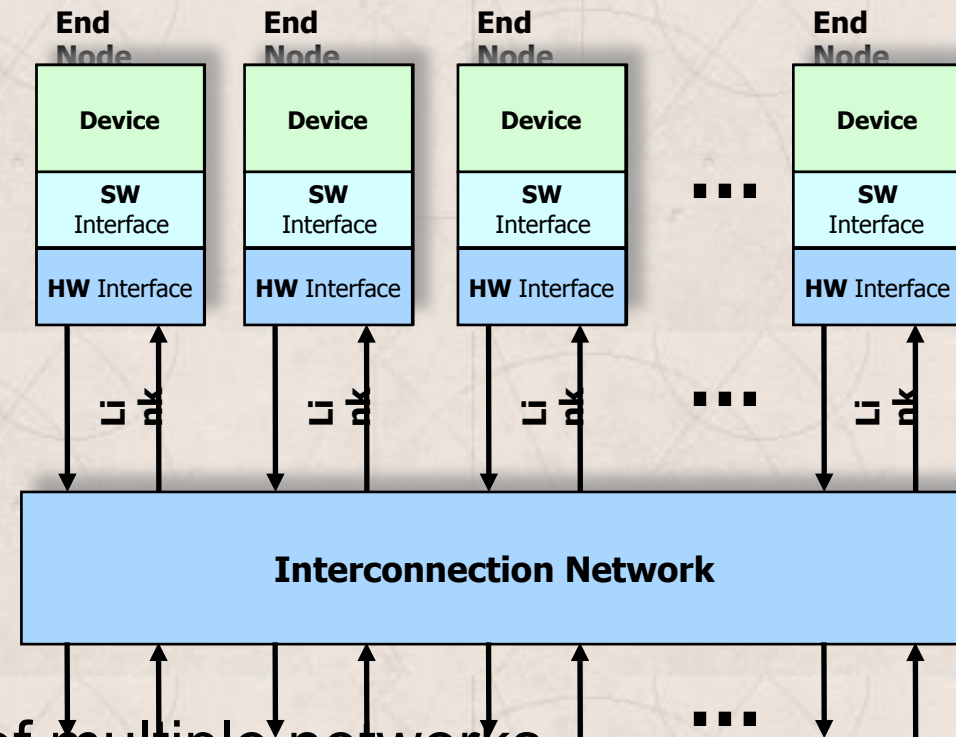
José Duato
Universidad Politécnica de Valencia
<http://www.gap.upv.es/slides/appendixE.html>

...with major presentation contribution from José Flich, UPV
(and Cell BE EIB slides by Tom Ainsworth, USC)

Introduction

How to connect individual devices together into a community of communicating devices?

- Device (definition):
 - Component within a computer
 - Single computer
 - System of computers
- Types of elements:
 - end nodes (device + interface)
 - links
 - interconnection network
- Internetworking: interconnection of multiple networks
- ***Interconnection networks** should be designed to transfer the maximum amount of information within the least amount of time (and cost, power constraints) so as not to bottleneck the system*



Introduction

Interconnection Network Domains

Interconnection networks can be grouped into *four major networking domains*, depending on the *number* and *proximity* of devices to be interconnected: OCNs, SANs, LANs, and WANs

- *On-chip networks (OCNs)*, a.k.a., network-on-chip (NoC)
 - Interconnect microarchitecture functional units, register files, caches, compute tiles, processor and IP cores
 - Chip or multichip modules
 - Tens (in future, possibly 100s) of devices interconnected
 - Maximum interconnect distance on the order of centimeters
 - Examples (custom designed)
 - › Element Interconnect Bus (Cell Broadband Engine processor chip)
 - » 2,400 Gbps (3.2 Ghz processor clock), 12 elements on the chip
 - Examples (proprietary designs)
 - › CoreConnect (IBM), AMBA (ARM), Smart Interconnect (Sonic)

Introduction

Interconnection Network Domains

- *System/storage area networks (SANs)*
 - Multiprocessor and multicomputer systems
 - › Interprocessor and processor-memory interconnections
 - Server and data center environments
 - › Storage and I/O components
 - Hundreds to thousands of devices interconnected
 - › IBM Blue Gene/L supercomputer (64K nodes, each with 2 processors)
 - Maximum interconnect distance typically on the order of tens of meters, but some with as high as a few hundred meters
 - › InfiniBand: 120 Gbps over a distance of 300 m
 - Examples (standards and proprietary)
 - › InfiniBand, Myrinet, Quadrics, Advanced Switching Interconnect

Introduction

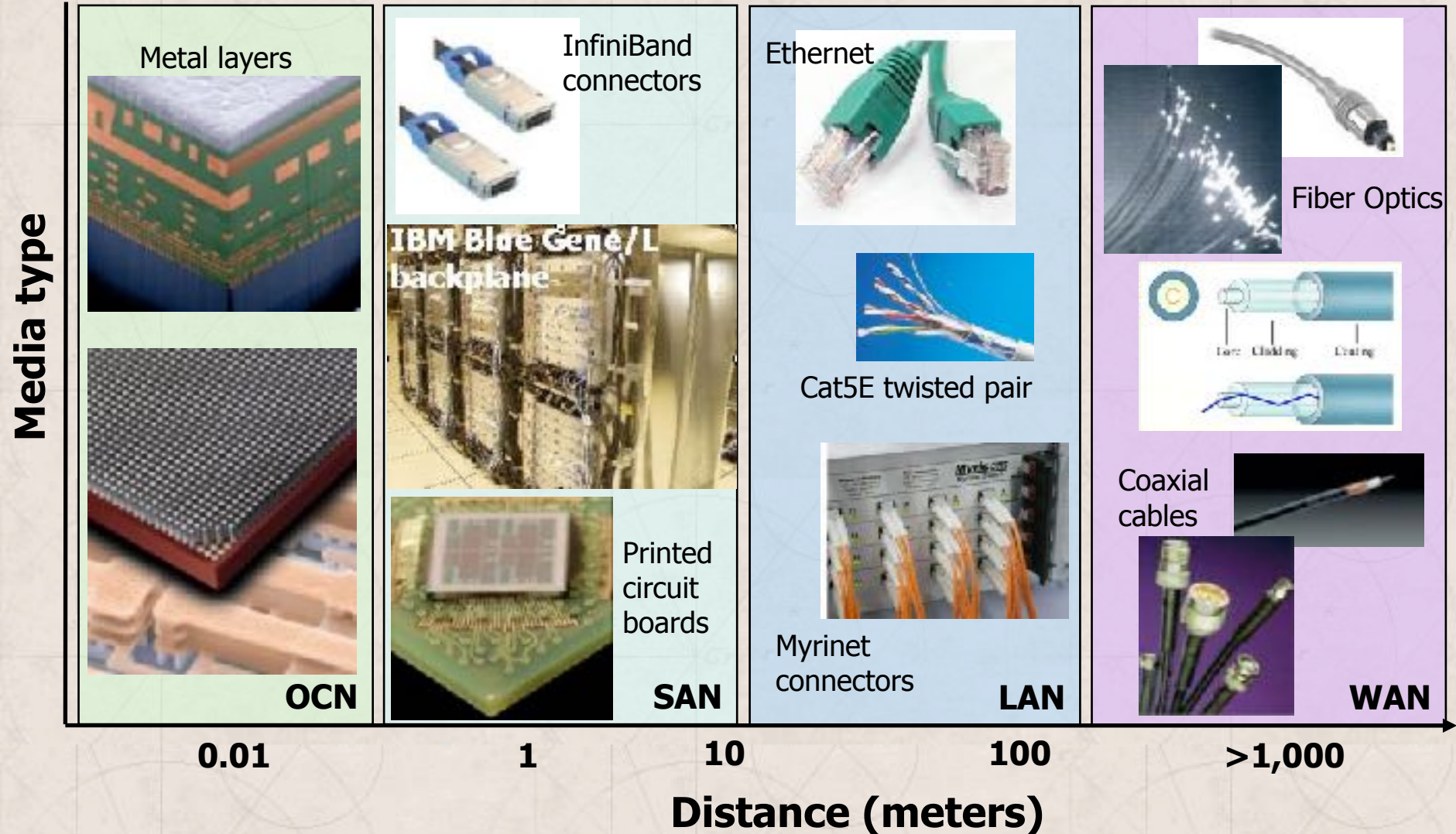
Interconnection Network Domains

- *Local area networks (LANs)*
 - Interconnect autonomous computer systems
 - Machine room or throughout a building or campus
 - Hundreds of devices interconnected (1,000s with bridging)
 - Maximum interconnect distance on the order of few kilometers, but some with distance spans of a few tens of kilometers
 - Hundred devices (thousands with bridging)
 - Example (most popular): Ethernet, with 10 Gbps over 40Km
- *Wide area networks (WANs)*
 - Interconnect systems distributed across the globe
 - Internetworking support is required
 - Many millions of devices interconnected
 - Maximum interconnect distance of many thousands of kilometers
 - Example: ATM

Interconnecting Two Devices

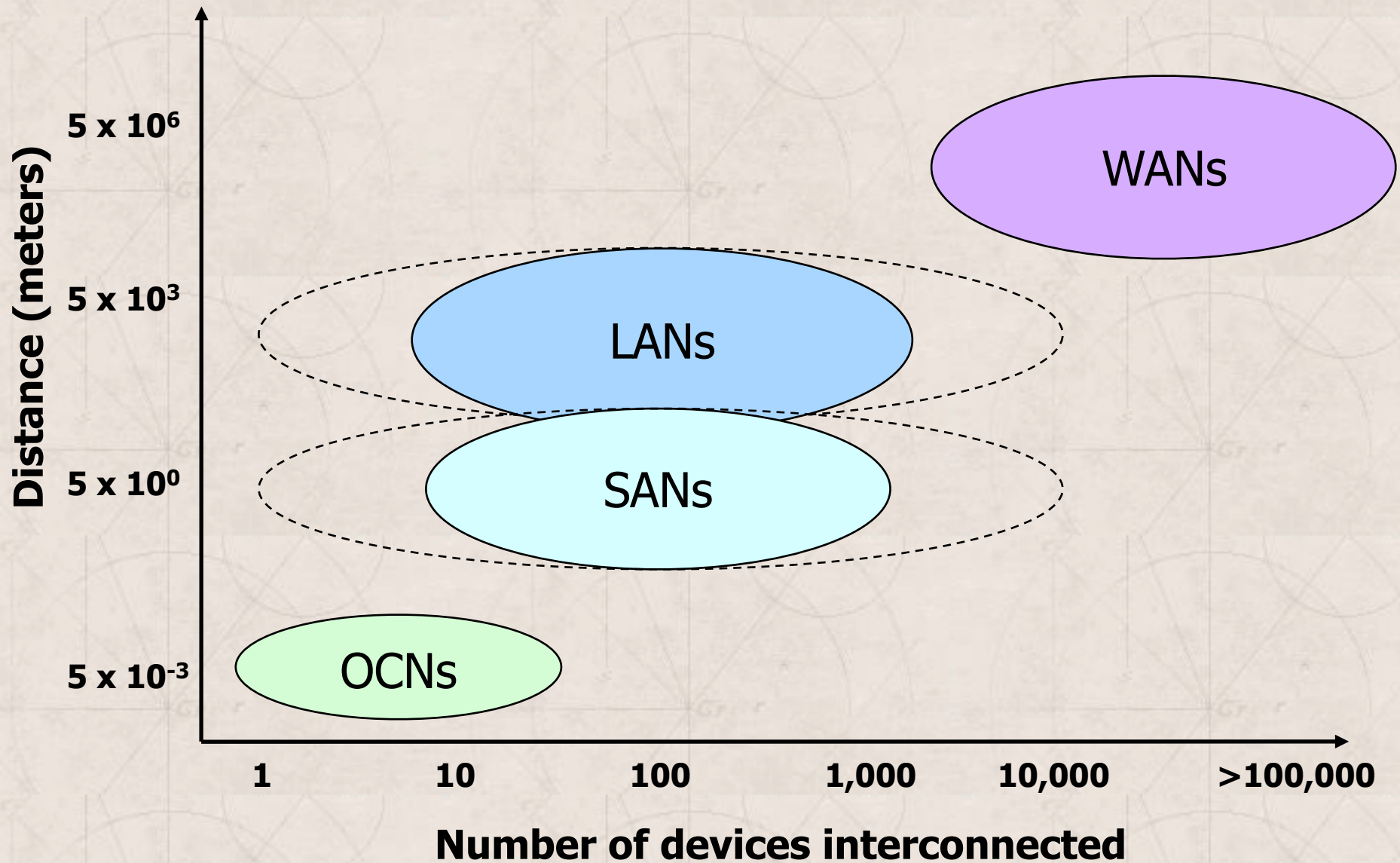
Basic Network Structure and Functions

- Media and Form Factor



Introduction

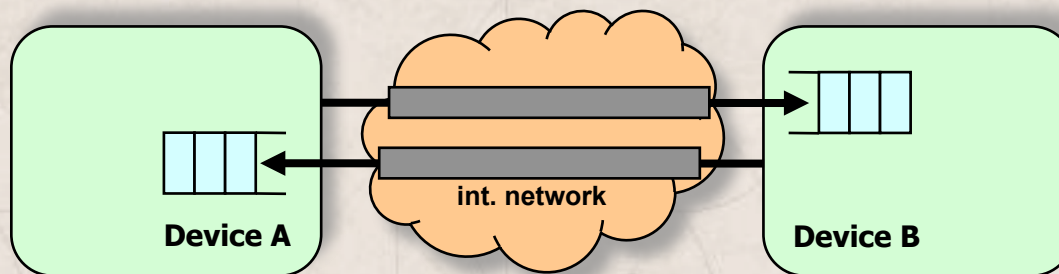
Interconnection Network Domains



Interconnecting Two Devices

An Ideal Network

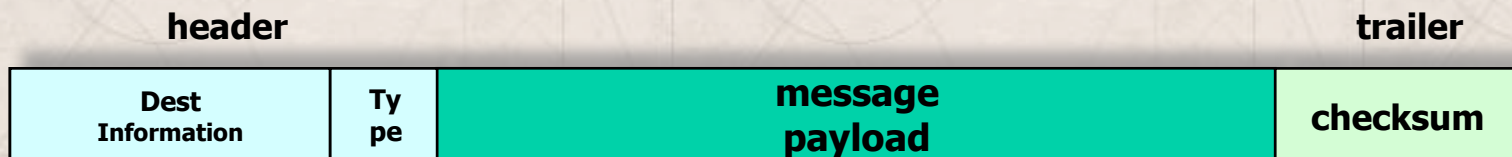
- Two end node devices (A and B)
- Device A (or B) may read an address in device B (or A)
- Interconnection network behaves as *dedicated links* between A, B
 - Unidirectional wires each way dedicated to each device
- Receiving buffers used for staging the transfers at the end nodes
- Communication protocol: *request, reply*
 - basic functions at end nodes to commence and complete comm.



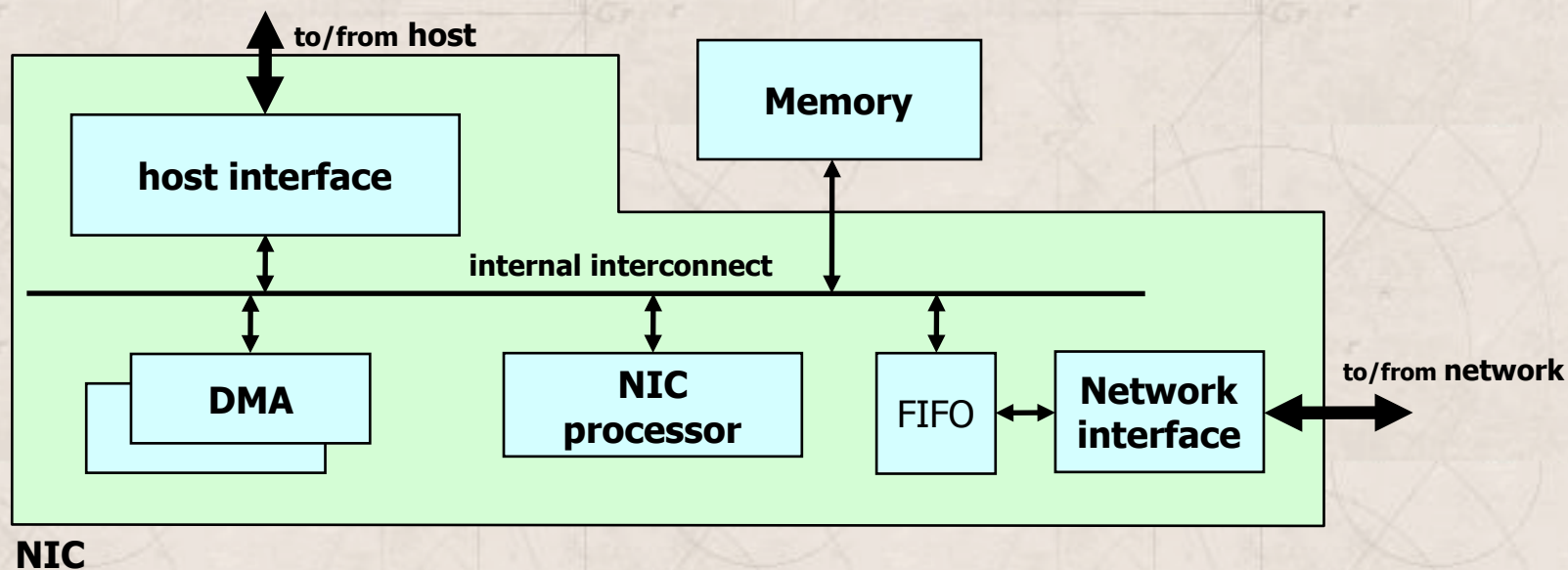
Interconnecting Two Devices

Network Interface Functions

- A *message* is the unit of information: header, payload, trailer



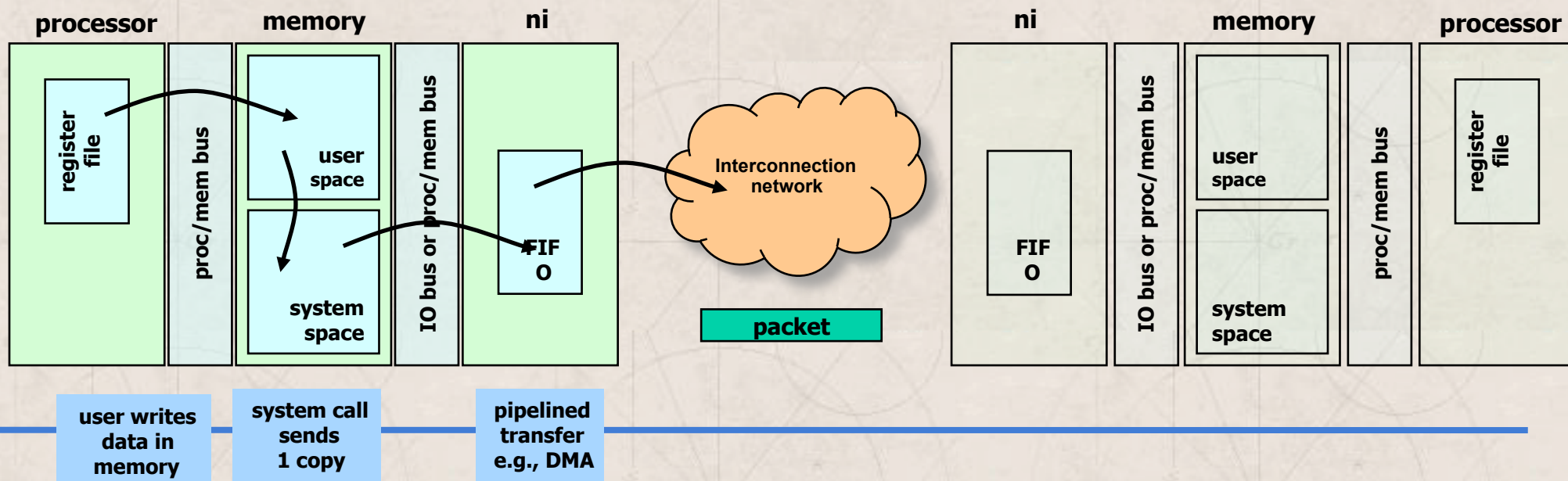
- Interfacing to the network (hardware)
 - Communication device itself (OCNs and some SANs)
 - Additional network interface card or *NIC* (SANs, LANs, WANs)
 - Embedded processor(s), DMA engine(s), RAM memory



Interconnecting Two Devices

Communication Protocol

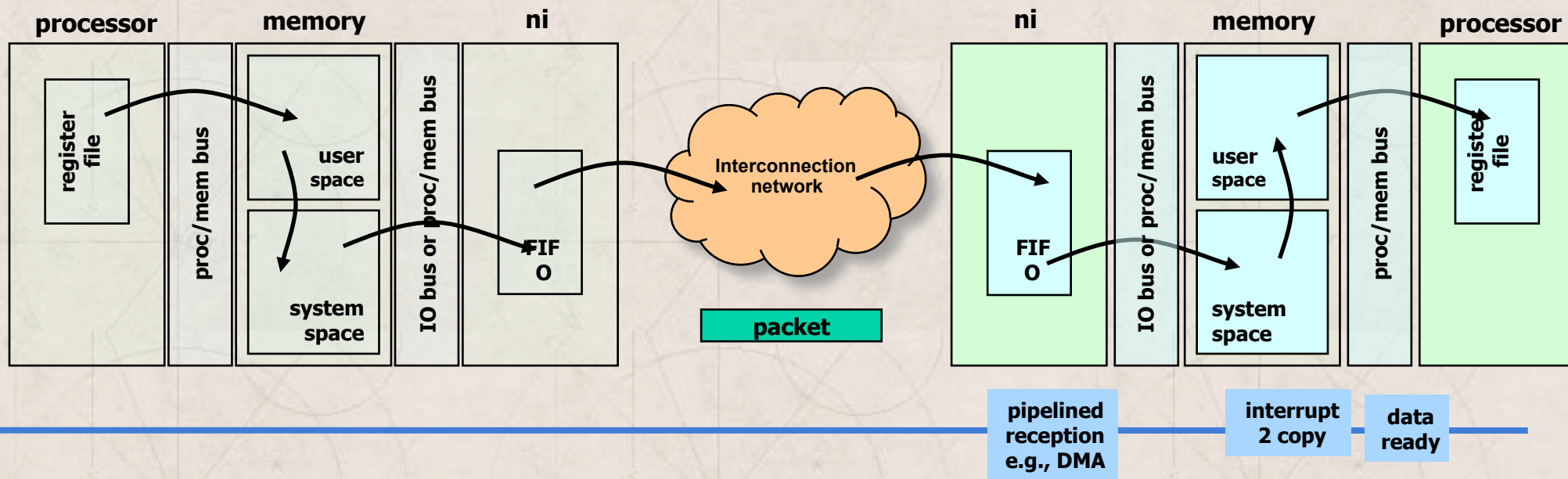
- Typical steps followed by the *sender*:
 1. System call by application
 - › Copies the data into OS and/or network interface memory
 - › Packetizes the message (if needed)
 - › Prepares headers and trailers of packets
 2. Checksum is computed and added to header/trailer
 3. Timer is started and the network interface sends the packets



Interconnecting Two Devices

Communication Protocol

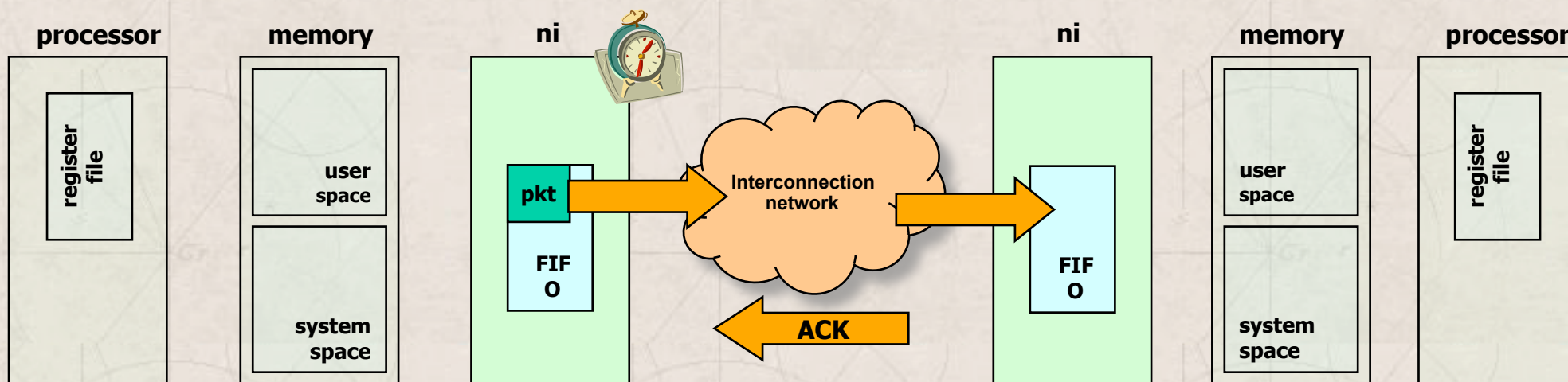
- Typical steps followed by the *receiver*:
 1. NI allocates received packets into its memory or OS memory
 2. Checksum is computed and compared for each packet
 - › If checksum matches, NI sends back an ACK packet
 3. Once all packets are correctly received
 - › The message is reassembled and copied to user's address space
 - › The corresponding application is signalled (via polling or interrupt)



Interconnecting Two Devices

Communication Protocol

- Additional steps at the sender side:
 - ACK received: the copy is released and timer is cancelled
 - Timer expires before receiving an ACK: packet is resent and the timer is started



Interconnecting Many Devices

Additional Network Structure and Functions

- Basic network structure and functions
 - Composing and processing messages, packets
 - Media and form factor
 - Packet transport
 - Reliable delivery (e.g., flow control) and error handling
- Additional structure
 - *Topology*
 - › *What paths are possible for packets?*
 - › Networks usually share paths among different pairs of devices
- Additional functions (*routing*, *arbitration*, *switching*)
 - Required in every network connecting more than two devices
 - Required to establish a valid path from source to destination
 - Complexity and applied order depends on the category of the topology: *shared-media* or *switched media*

Interconnecting Many Devices

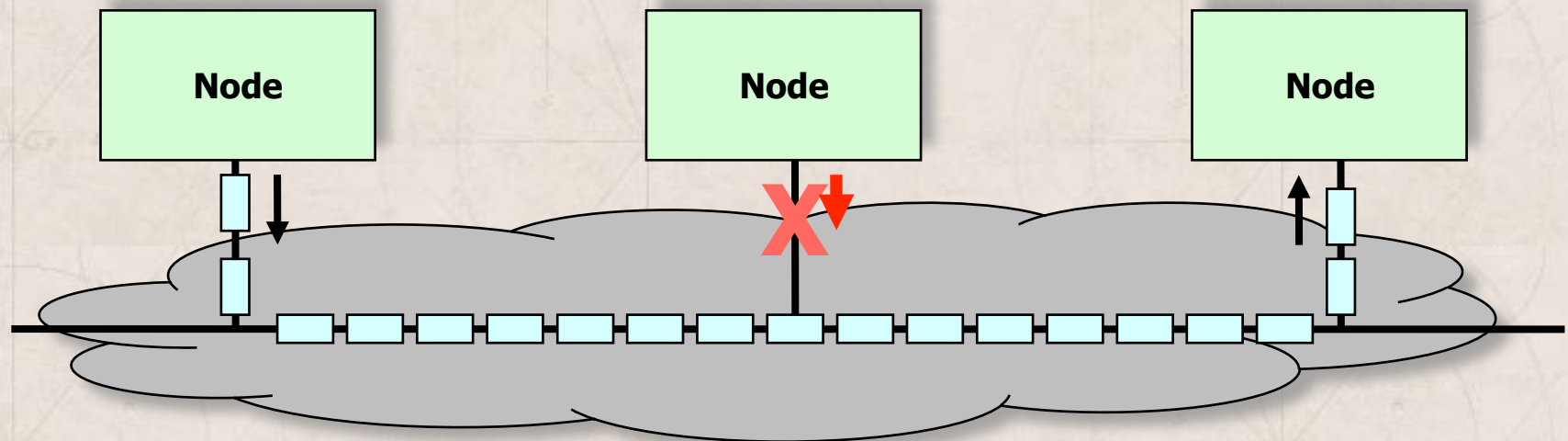
Additional Network Structure and Functions

- Additional functions (routing, arbitration, switching)
 - *Routing*
 - › *Which of the possible paths are allowable (valid) for packets?*
 - › Provides the set of operations needed to compute a valid path
 - › Executed at source, intermediate, or even at destination nodes
 - *Arbitration*
 - › *When are paths available for packets?* (along with flow control)
 - › Resolves packets requesting the same resources at the same time
 - › For every arbitration, there is a winner and possibly many losers
 - » Losers are buffered (lossless) or dropped on overflow (lossy)
 - *Switching*
 - › *How are paths allocated to packets?*
 - › The winning packet (from arbitration) proceeds towards destination
 - › Paths can be established one fragment at a time or in their entirety

Interconnecting Many Devices

Shared-media Networks

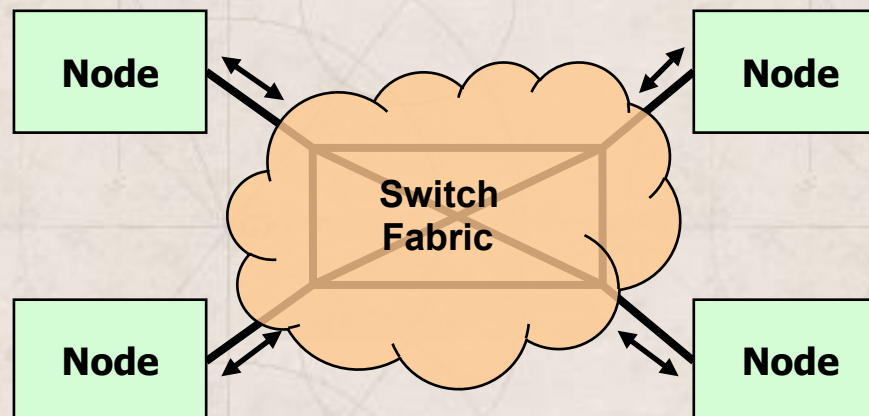
- The network media is shared by all the devices
- Operation: half-duplex or full-duplex



Interconnecting Many Devices

Switched-media Networks

- *Disjoint portions of the media are shared via switching*
- Switch fabric components
 - Passive *point-to-point links*
 - Active *switches*
 - › Dynamically establish communication between sets of source-destination pairs
- Aggregate bandwidth can be many times higher than that of shared-media networks



Interconnecting Many Devices

Comparison of Shared- versus Switched-media Networks

- Shared-media networks
 - Low cost
 - Aggregate network bandwidth does not scale with # of devices
 - Global arbitration scheme required (a possible bottleneck)
 - Time of flight increases with the number of end nodes
- Switched-media networks
 - Aggregate network bandwidth scales with number of devices
 - Concurrent communication
 - › Potentially much higher network effective bandwidth
 - **Beware:** inefficient designs are quite possible
 - › Superlinear network cost but sublinear network effective bandwidth

Network Topology

Preliminaries and Evolution

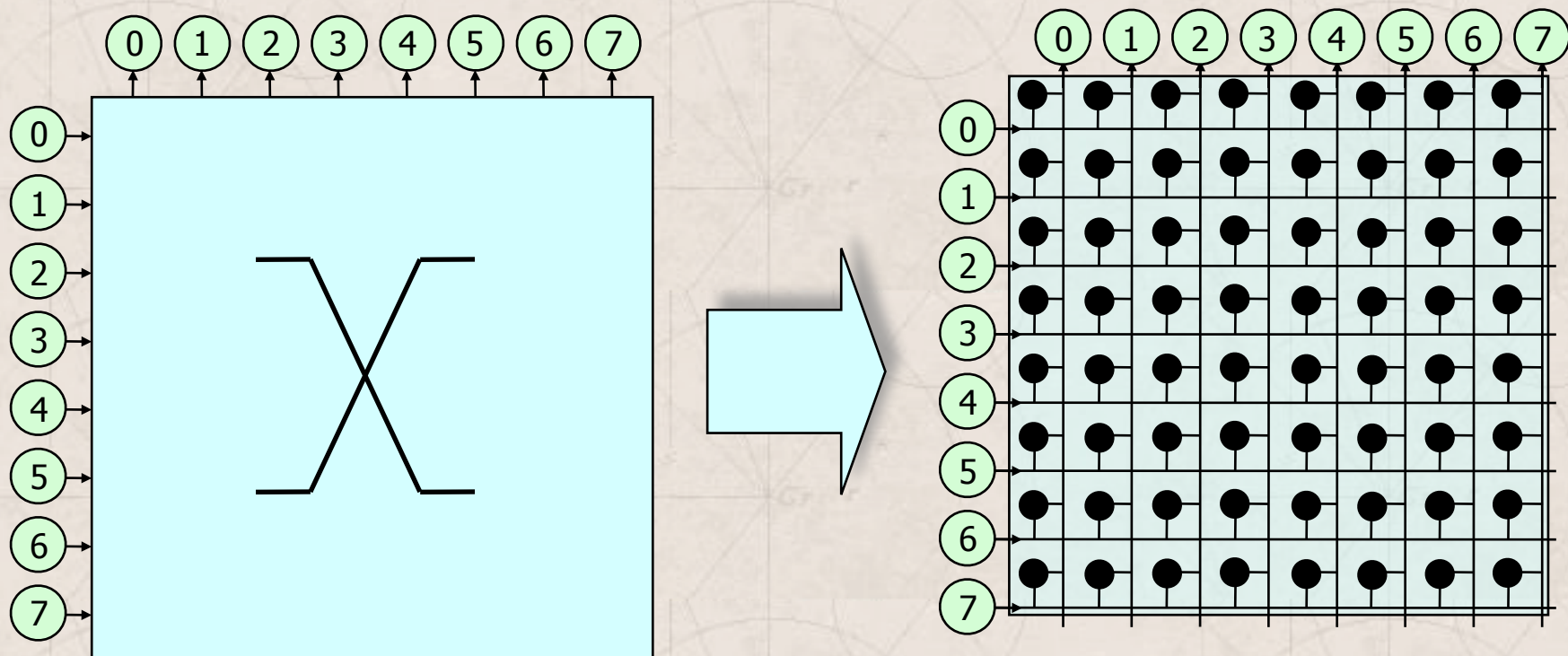
- One switch suffices to connect a small number of devices
 - Number of switch ports limited by VLSI technology, power consumption, packaging, and other such *cost* constraints
- A *fabric* of interconnected switches (i.e., *switch fabric* or *network fabric*) is needed when the number of devices is much larger
 - The *topology* must make a path(s) available for every pair of devices—property of *connectedness* or *full access* (*What paths?*)
- Topology defines the connection structure across all components
 - *Bisection bandwidth*: the *minimum* bandwidth of all links crossing a network split into two roughly equal halves
 - *Full bisection bandwidth*:
 - › Network $BW_{Bisection} = \text{Injection (or Reception) } BW_{Bisection} = N/2$
 - Bisection bandwidth mainly affects *performance*
- Topology is constrained primarily by local chip/board *pin-outs*; secondarily, (if at all) by global bisection bandwidth

Network Topology

Centralized Switched (Indirect) Networks

- **Crossbar network**

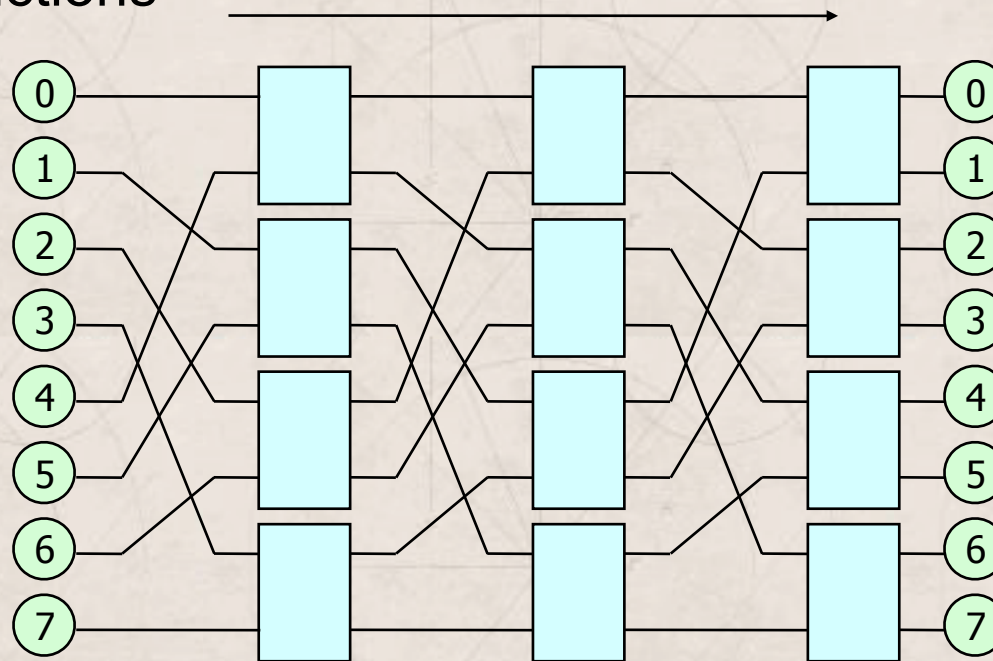
- Crosspoint switch complexity increases quadratically with the number of crossbar input/output ports, N , i.e., grows as $O(N^2)$
- Has the property of being *non-blocking*



Network Topology

Centralized Switched (Indirect) Networks

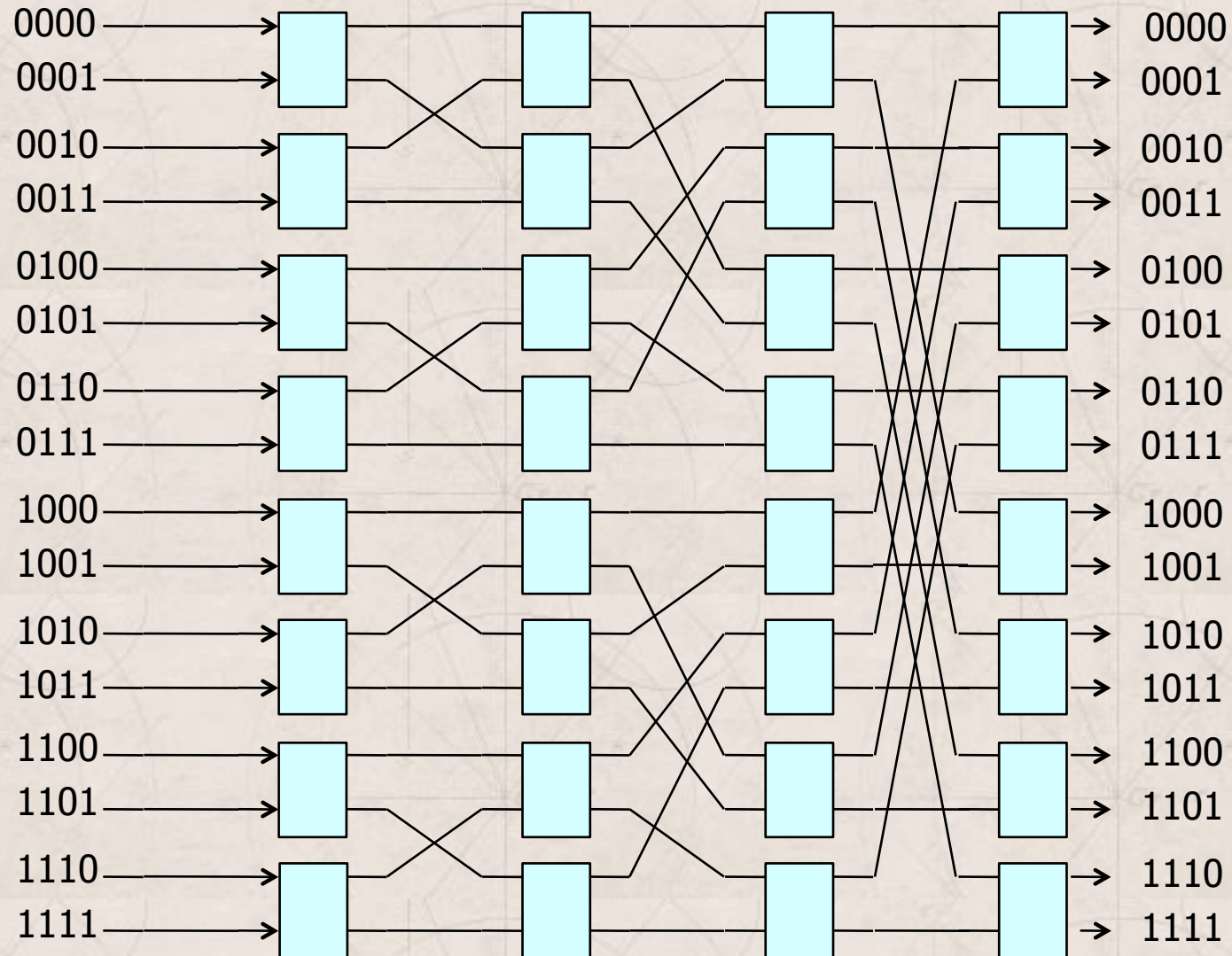
- **Multistage interconnection networks (MINs)**
 - Crossbar split into several stages consisting of smaller crossbars
 - Complexity grows as $O(N \times \log N)$, where N is # of end nodes
 - Inter-stage connections represented by a set of permutation functions



Omega topology, perfect-shuffle exchange

Network Topology

Centralized Switched (Indirect) Networks

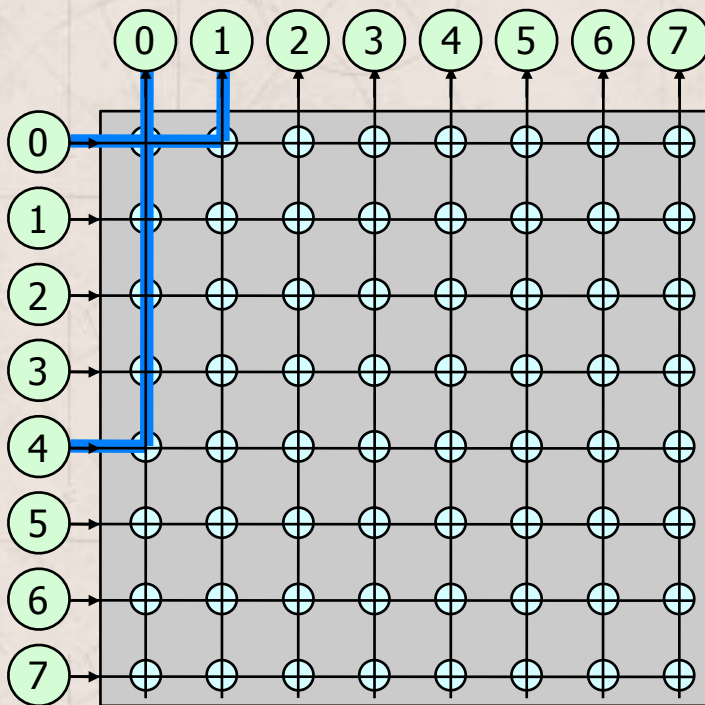


16 port, 4 stage *Butterfly* network

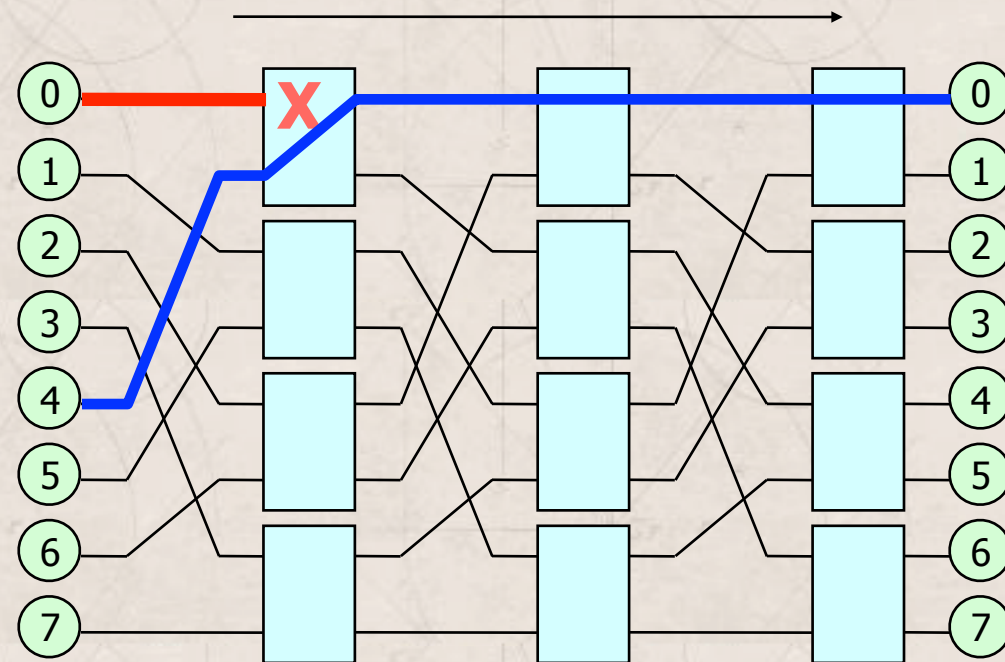
Network Topology

Centralized Switched (Indirect) Networks

- Reduction in MIN switch cost comes at the price of performance
 - Network has the property of being *blocking*
 - *Contention* is more likely to occur on network links
 - › Paths from different sources to different destinations share one or more links



non-blocking topology



blocking topology

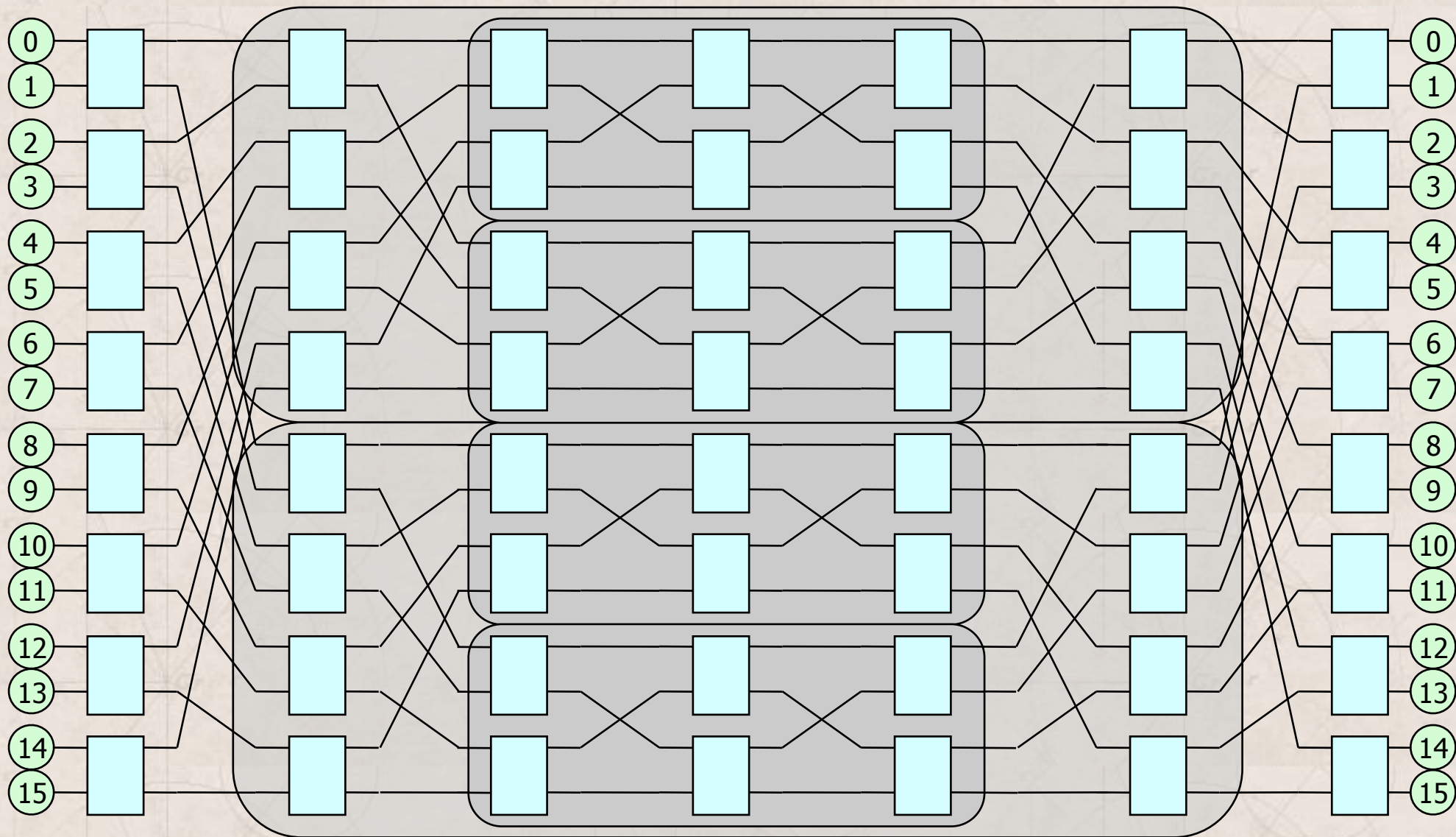
Network Topology

Centralized Switched (Indirect) Networks

- How to reduce blocking in MINs? Provide alternative paths!
 - Use larger switches (can equate to using more switches)
 - › **Clos network**: minimally three stages (non-blocking)
 - » A larger switch in the middle of two other switch stages provides enough alternative paths to avoid all conflicts
 - Use more switches
 - › Add $\log_k N - 1$ stages, mirroring the original topology
 - » *Rearrangeably non-blocking*
 - » Allows for non-conflicting paths
 - » Doubles network *hop count (distance)*, d
 - » Centralized control can rearrange established paths
 - › **Benes topology**: $2(\log_2 N) - 1$ stages (rearrangeably non-blocking)
 - » Recursively applies the three-stage Clos network concept to the middle-stage set of switches to reduce all switches to 2×2

Network Topology

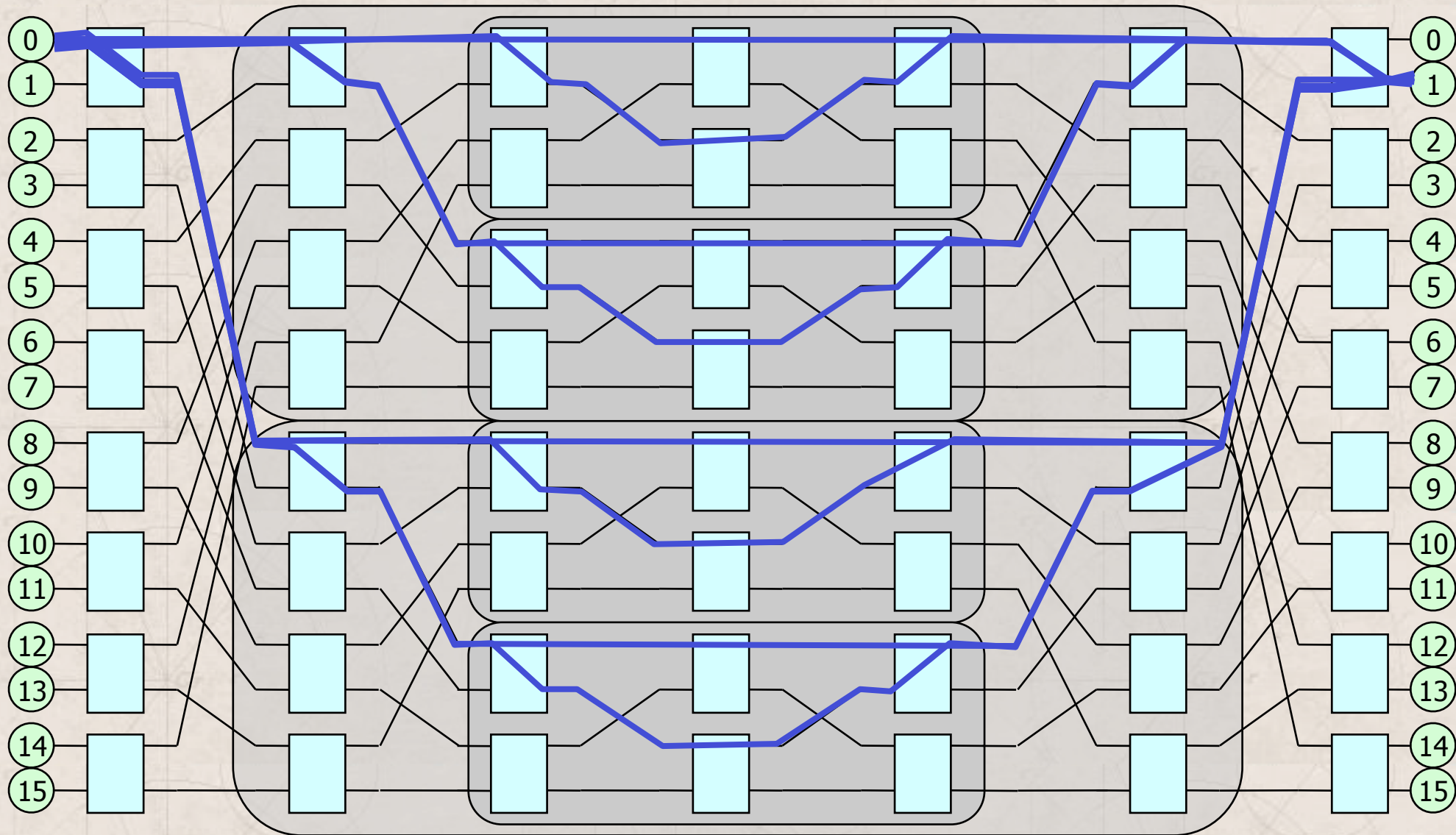
Centralized Switched (Indirect) Networks



16 port, **7 stage Clos** network = **Benes topology**

Network Topology

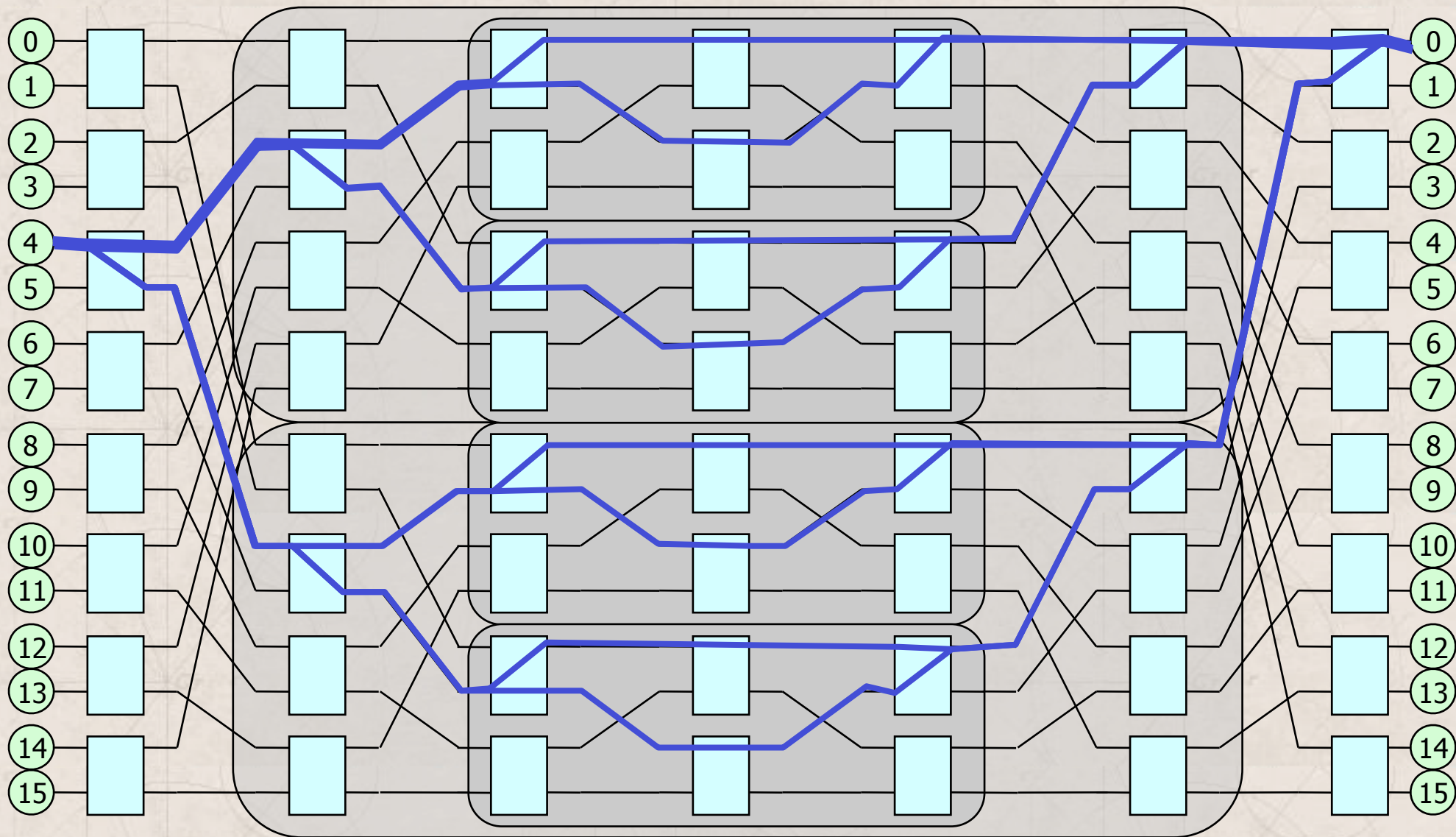
Centralized Switched (Indirect) Networks



Alternative paths from 0 to 1. 16 port, 7 stage Clos network = *Benes topology*

Network Topology

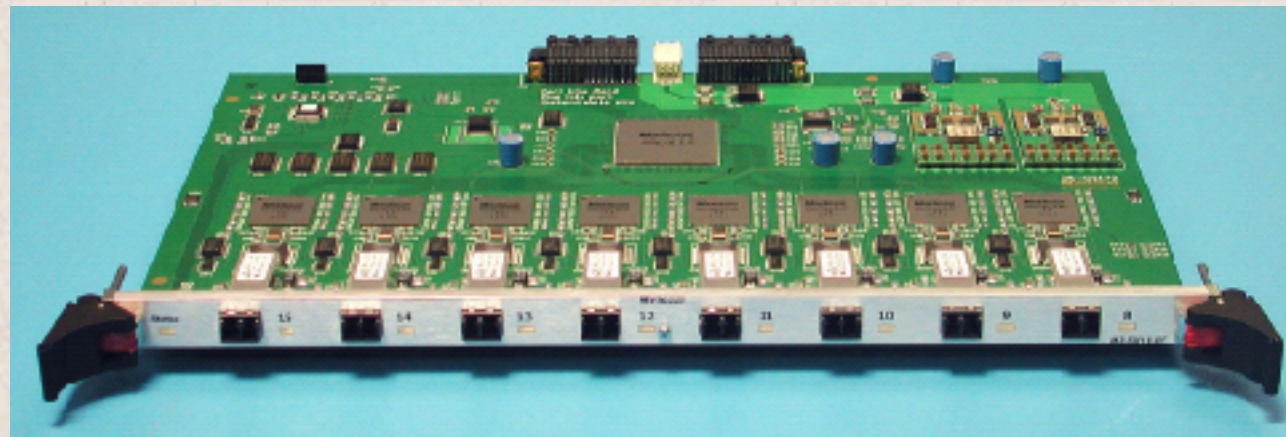
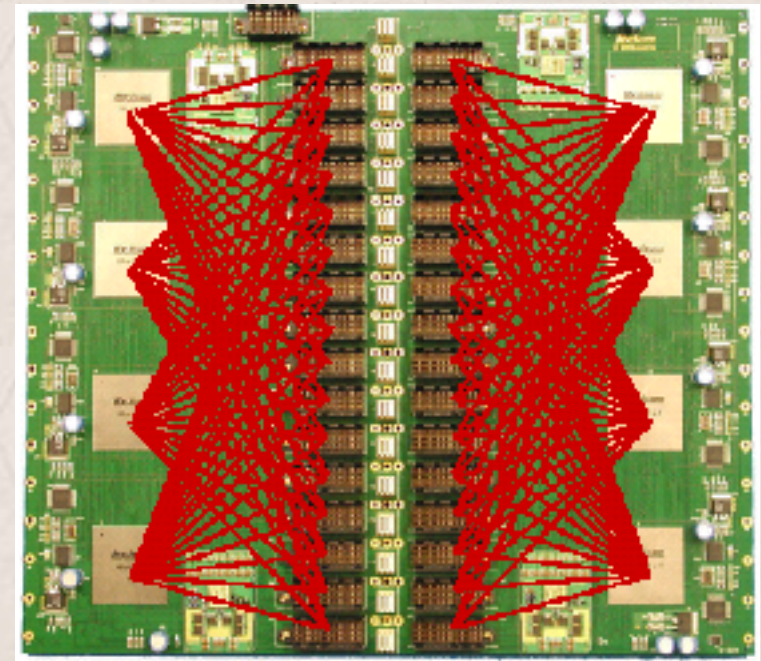
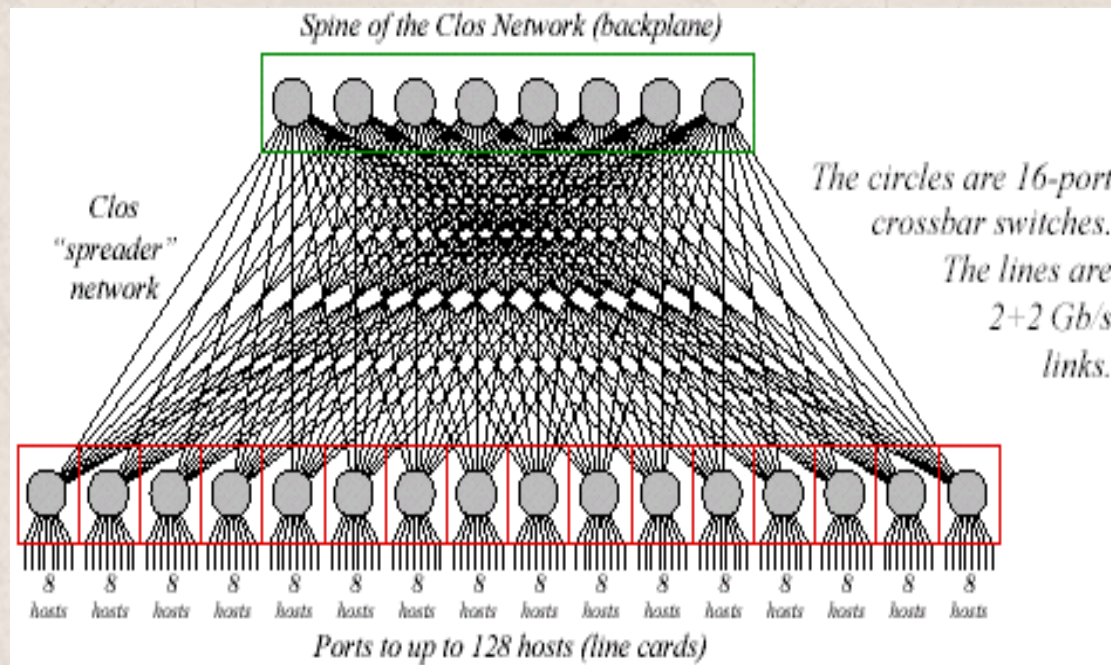
Centralized Switched (Indirect) Networks



Alternative paths from 4 to 0. 16 port, 7 stage Clos network = *Benes topology*

Network Topology

Myrinet-2000 Clos Network for 128 Hosts

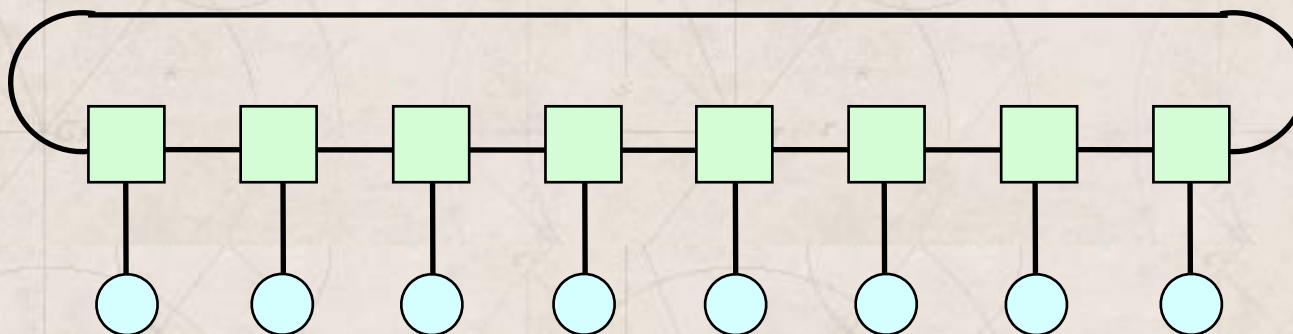


- Backplane of the M3-E128 Switch
- M3-SW16-8F fiber line card (8 ports)

Network Topology

Distributed Switched (Direct) Networks

- ***Bidirectional Ring networks***
 - N switches (3×3) and N bidirectional network links
 - Simultaneous packet transport over disjoint paths
 - Packets must hop across intermediate nodes
 - Shortest direction usually selected ($N/4$ hops, on average)



Network Topology

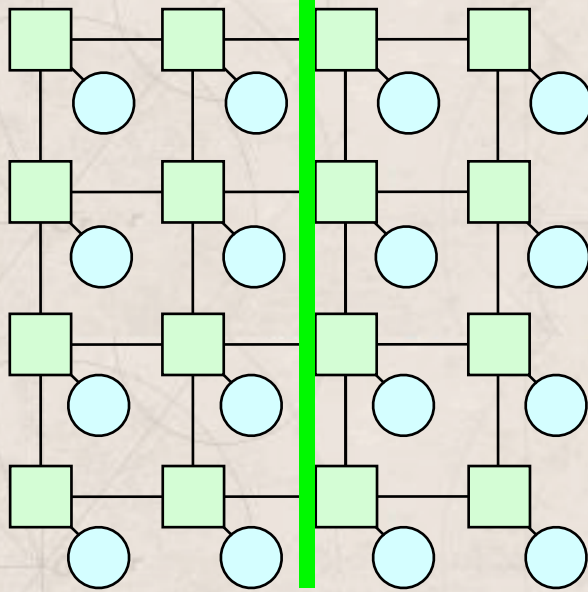
Distributed Switched (Direct) Networks:

- *Fully connected and ring topologies delimit the two extremes*
- ***The ideal topology:***
 - Cost approaching a ring
 - Performance approaching a fully connected (crossbar) topology
- More practical topologies:
 - ***k-ary n-cubes*** (*meshes*, *tori*, *hypercubes*)
 - › ***k*** nodes connected in each dimension, with ***n*** total dimensions
 - › *Symmetry and regularity*
 - » network implementation is simplified
 - » routing is simplified

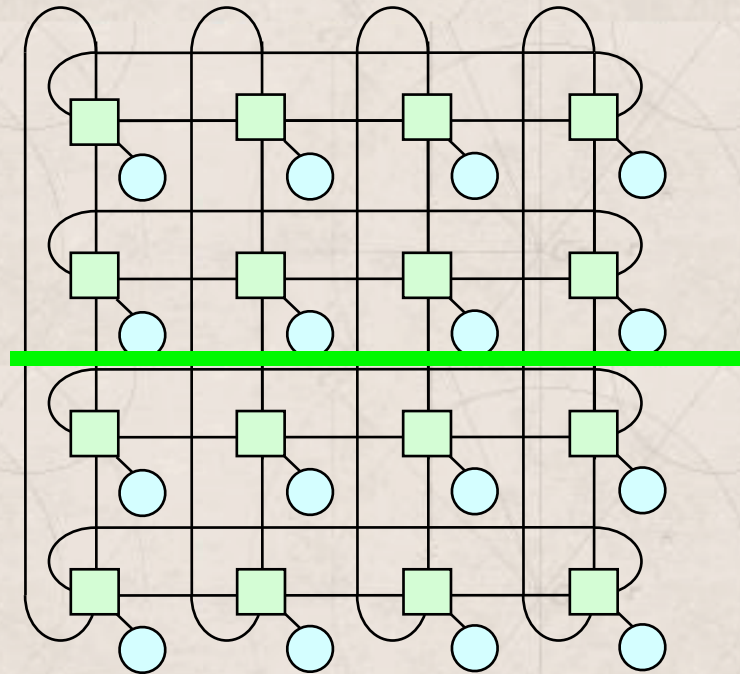
Network Topology

Distributed Switched (Direct) Networks

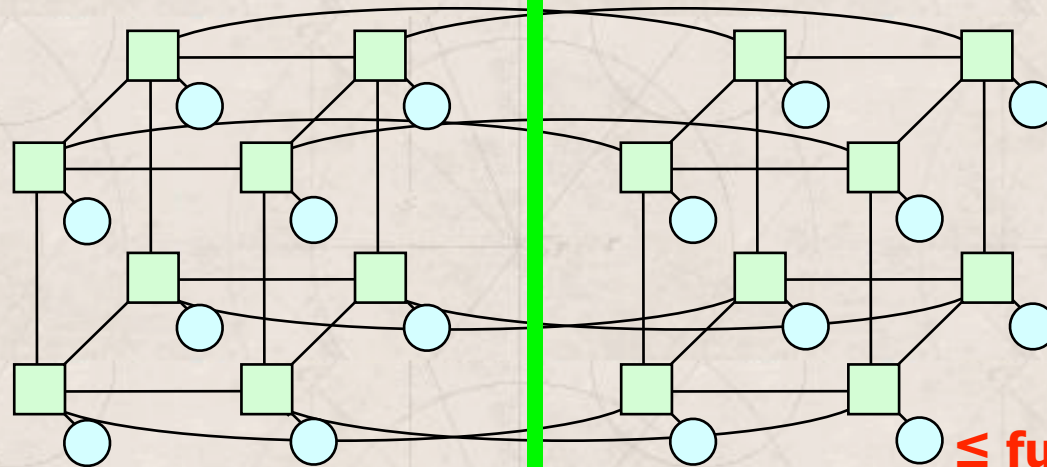
2D **mesh** or grid of 16 nodes



2D **torus** of 16 nodes



hypercube of 16 nodes
($16 = 2^4$, so $n = 4$)



**Network
Bisection**

\leq full bisection bandwidth!

Examples of Interconnection Networks

On-Chip Networks (OCNs)

Institution & Processor [Network] name	Year built	Number of network ports [cores or tiles + other ports]	Basic network topology	# of data bits per link per direction	Link bandwidth [link clock speed]	Routing; Arbitration; Switching	# of chip metal layers; flow control; # VCs
MIT Raw [General Dynamic Network]	2002	16 port [16 tiles]	2-D mesh 4 x 4	32 bits	0.9 GBps [225 MHz, clocked at proc speed]	XY DOR w/ request-reply deadlock recovery; RR arbitration; wormhole	6 layers; credit-based; no VCs
IBM POWER5	2004	7 ports [2 PE cores + 5 other ports]	Crossbar	256 b Inst fetch; 64 b for stores; 256 b LDs	[1.9 GHz, clocked at proc speed]	Shortest-path; non-blocking; circuit switch	7 layers; handshaking; no virtual channels
U.T. Austin TRIPS EDGE [Operand Network]	2005	25 ports [25 execution unit tiles]	2-D mesh 5 x 5	110 bits	5.86 GBps [533 MHz clk scaled by 80%]	XY DOR; distributed RR arbitration; wormhole	7 layers; on/off flow control; no VCs
U.T. Austin TRIPS EDGE [On-Chip Network]	2005	40 ports [16 L2 tiles + 24 network interface tile]	2-D mesh 10 x 4	128 bits	6.8 GBps [533 MHz clk scaled by 80%]	XY DOR; distributed RR arbitration; VCT switched	7 layers; credit-based flow control; 4 VCs
Sony, IBM, Toshiba Cell BE [Element Interconnect Bus]	2005	12 ports [1 PPE and 8 SPEs + 3 other ports for memory, I/O interface]	Ring 4 total, 2 in each direction	128 bits data (+16 bits tag)	25.6 GBps [1.6 GHz, clocked at half the proc speed]	Shortest-path; tree-based RR arb. (centralized); pipelined circuit switch	8 layers; credit-based flow control; no VCs
Sun UltraSPARC T1 processor	2005	Up to 13 ports [8 PE cores + 4 L2 banks + 1 shared I/O]	Crossbar	128 b both for the 8 cores and the 4 L2 banks	19.2 GBps [1.2 GHz, clocked at proc speed]	Shortest-path; age-based arbitration; VCT switched	9 layers; handshaking; no VCs

Network Topology

Topological Characteristics of Commercial Machines

Company	System [Network] Name	Max. number of nodes [x # CPUs]	Basic network topology	Injection [Recept'n] node BW in MBytes/s	# of data bits per link per direction	Raw network link BW per direction in Mbytes/sec	Raw network bisection BW (bidir) in Gbytes/s
Intel	ASCI Red Paragon	4,510 [x 2]	2-D mesh 64 x 64	400 [400]	16 bits	400	51.2
IBM	ASCI White SP Power3 [Colony]	512 [x 16]	BMIN w/8-port bidirect. switches (fat- tree or Omega)	500 [500]	8 bits (+1 bit of control)	500	256
Intel	Thunter Itanium2 Tiger4 [QsNet ^{II}]	1,024 [x 4]	fat tree w/8-port bidirectional switches	928 [928]	8 bits (+2 control for 4b/5b enc)	1,333	1,365
Cray	XT3 [SeaStar]	30,508 [x 1]	3-D torus 40 x 32 x 24	3,200 [3,200]	12 bits	3,800	5,836.8
Cray	X1E	1,024 [x 1]	4-way bristled 2-D torus (~ 23 x 11) with express links	1,600 [1,600]	16 bits	1,600	51.2
IBM	ASC Purple pSeries 575 [Federation]	>1,280 [x 8]	BMIN w/8-port bidirect. switches (fat-tree or Omega)	2,000 [2,000]	8 bits (+2 bits of control)	2,000	2,560
IBM	Blue Gene/L eServer Sol. [Torus Net]	65,536 [x 2]	3-D torus 32 x 32 x 64	612,5 [1,050]	1 bit (bit serial)	175	358.4

Routing, Arbitration, and Switching

Routing

- Performed at each switch, regardless of topology
- Defines the “allowed” path(s) for each packet (*Which paths?*)
- Needed to direct packets through network to intended destinations
- **Ideally:**
 - *Supply as many routing options to packets as there are paths provided by the topology, and evenly distribute network traffic among network links using those paths, minimizing contention*
- **Problems:** situations that cause packets never to reach their dest.
 - **Livelock**
 - › Arises from an unbounded number of allowed non-minimal hops
 - › *Solution:* restrict the number of non-minimal (mis)hops allowed
 - **Deadlock**
 - › Arises from a set of packets being blocked waiting only for network resources (i.e., links, buffers) held by other packets in the set
 - › Probability increases with increased traffic & decreased availability

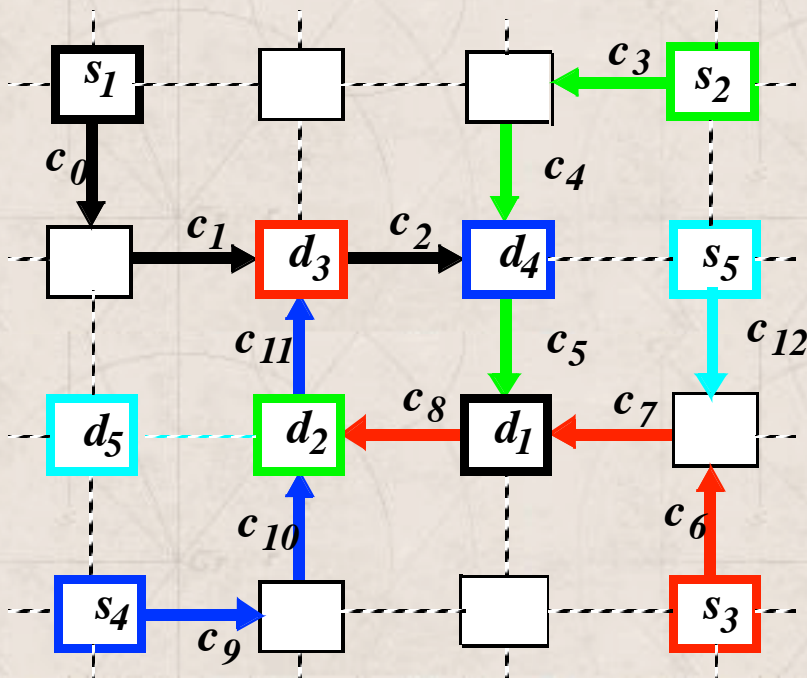
Routing, Arbitration, and Switching

Routing

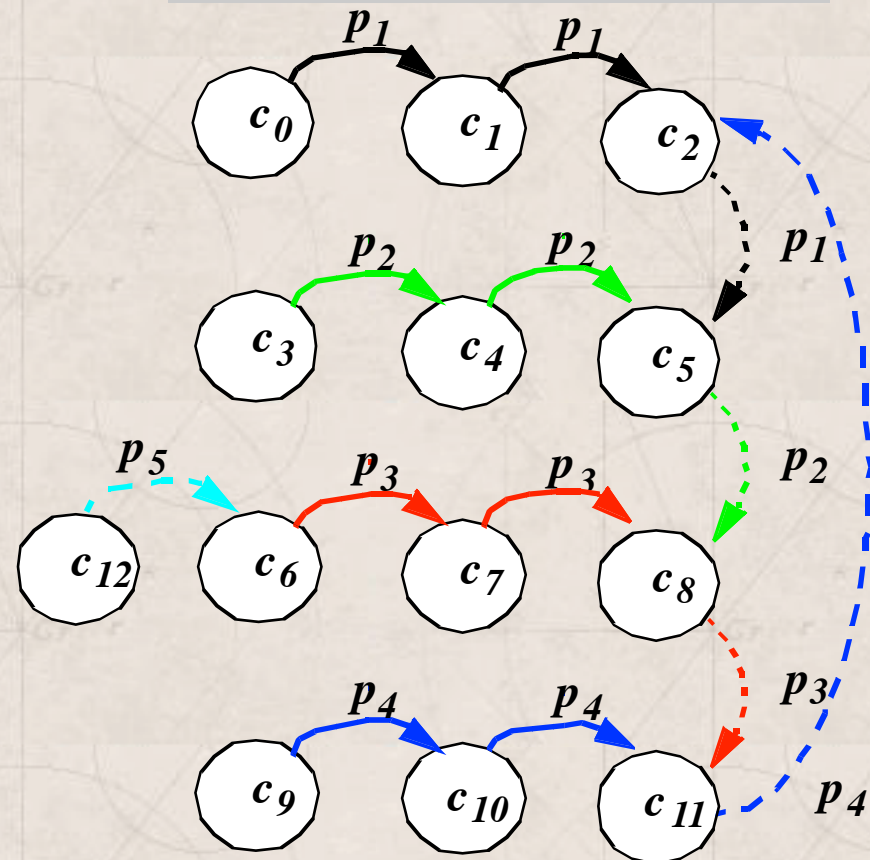
- Common forms of deadlock:
 - Routing-induced deadlock*

c_i = channel i
 s_i = source node i
 d_i = destination node i
 p_i = packet i

Routing of packets in a 2D mesh



Channel dependency graph



Examples of Interconnection Networks

Cell Broadband Engine Element Interconnect Bus

- Cell BE is successor to PlayStation 2's Emotion Engine
 - 300 MHz MIPS-based
 - Uses two vector elements
 - 6.2 GFLOPS (Single Precision)
 - 72KB Cache + 16KB Scratch Pad RAM
 - 240mm² on 0.25-micron process
- PlayStation 3 uses the Cell BE*
 - 3.2 GHz POWER-based
 - Eight SIMD (Vector) Processor Elements
 - >200 GFLOPS (Single Precision)
 - 544KB cache + 2MB Local Store RAM
 - 235mm² on 90-nanometer SOI process

*Sony has decided to use only 7 SPEs for the PlayStation 3 to improve yield.
Eight SPEs will be assumed for the purposes of this discussion.

Examples of Interconnection Networks

Cell Broadband Engine Element Interconnect Bus

- Cell Broadband Engine (Cell BE): 200 GFLOPS
 - 12 Elements (devices) interconnected by EIB:
 - › One 64-bit Power processor element (PPE) with aggregate bandwidth of 51.2 GB/s
 - › *Eight* 128-bit SIMD synergistic processor elements (SPE) with local store, each with a bandwidth of 51.2 GB/s
 - › One memory interface controller (MIC) element with memory bandwidth of 25.6 GB/s
 - › Two configurable I/O interface elements: 35 GB/s (out) and 25GB/s (in) of I/O bandwidth
 - Element Interconnect Bus (EIB):
 - › Four unidirectional rings (*two in each direction*) each connect the heterogeneous 12 elements (end node devices)
 - › Data links: 128 bits wide @ 1.6 GHz; data bandwidth: 25.6 GB/s
 - › Provides coherent and non-coherent data transfer
 - › Should optimize network traffic flow (throughput) and utilization while minimizing network latency and overhead

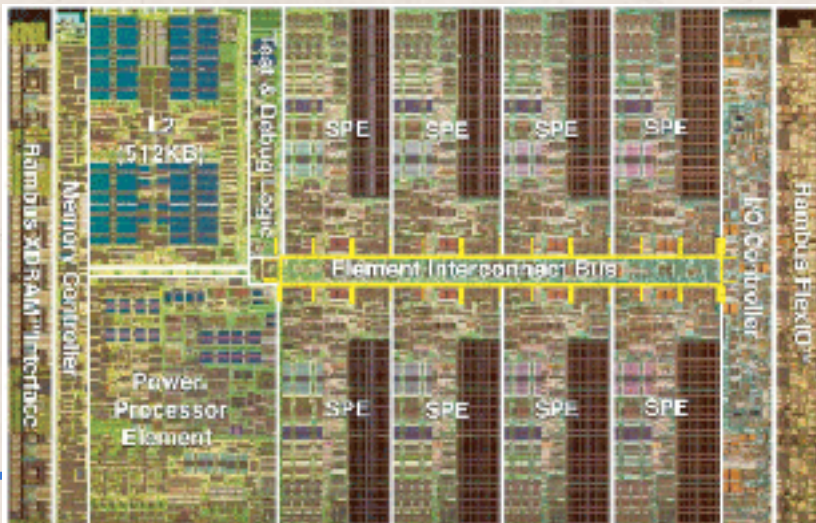
Examples of Interconnection Networks



Examples of Interconnection Networks

Cell Broadband Engine Element Interconnect Bus

- *Element Interconnect Bus (EIB)*
 - Packet size: 16B – 128B (no headers); pipelined circuit switching
 - Credit-based flow control (command bus central token manager)
 - Two-stage, dual round-robin centralized network arbiter
 - Allows up to 64 outstanding requests (DMA)
 - › 64 Request Buffers in the MIC; 16 Request Buffers per SPE
 - Latency: 1 cycle/hop, transmission time (largest packet) 8 cycles
 - Effective bandwidth: peak 307.2 GB/s, max. sustainable 204.8 GB/s



Examples of Interconnection Networks

Blue Gene/L 3D Torus Network

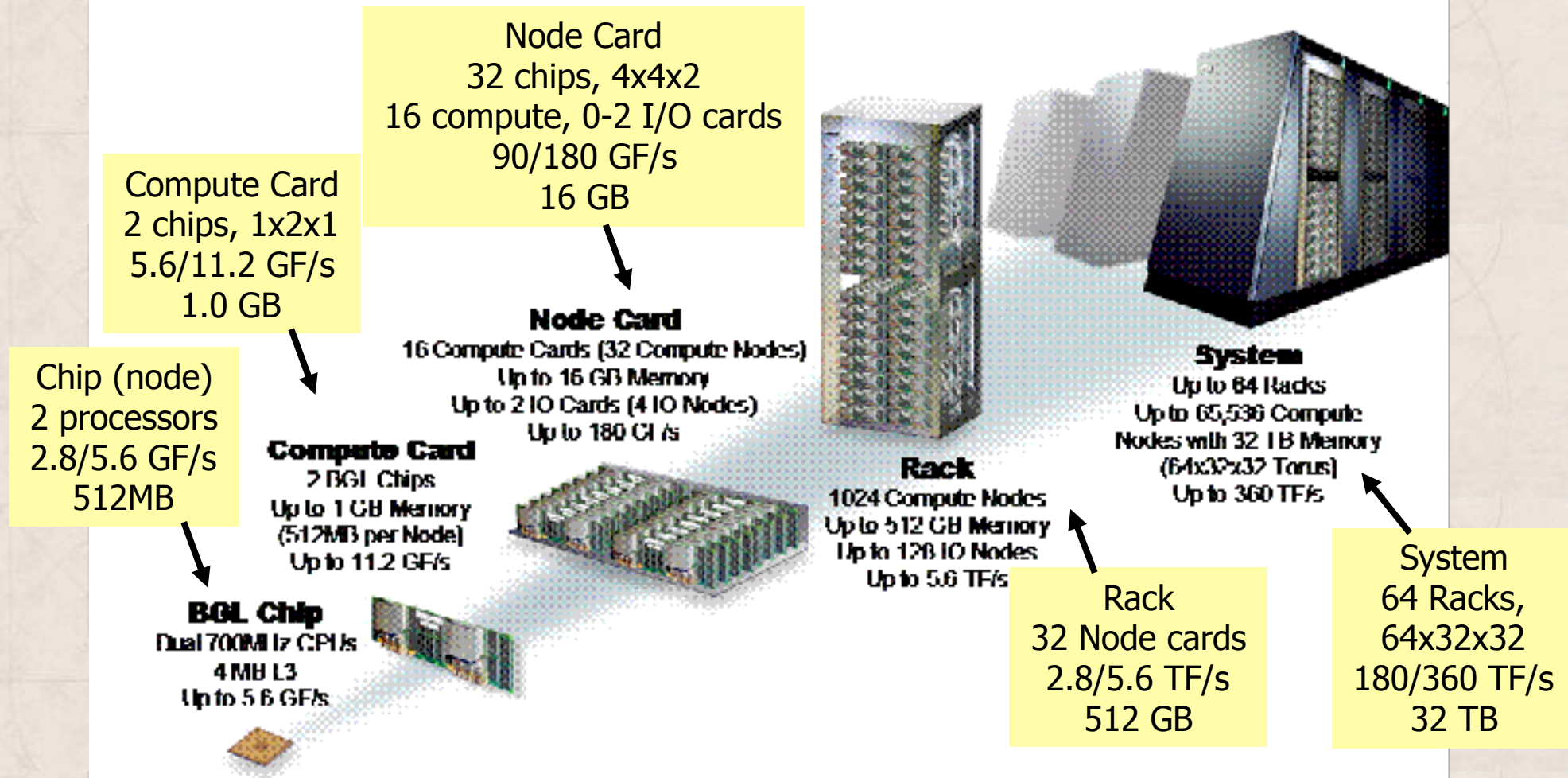
- 360 TFLOPS (peak)
- 2,500 square feet
- Connects 65,536 dual-processor nodes and 1,024 I/O nodes
 - One processor for computation; other meant for communication



Examples of Interconnection Networks

Blue Gene/L 3D Torus Network

www.ibm.com



Node distribution: Two nodes on a 2 x 1 x 1 compute card, 16 compute cards + 2 I/O cards on a 4 x 4 x 2 node board, 16 node boards on an 8 x 8 x 8 midplane, 2 midplanes on a 1,024 node rack, 8.6 meters maximum physical link length

Examples of Interconnection Networks

Blue Gene/L 3D Torus Network

- Main network: 32 x 32 x 64 3-D torus
 - Each node connects to six other nodes
 - Full routing in hardware
- Links and Bandwidth
 - 12 bit-serial links per node (6 in, 6 out)
 - Torus clock speed runs at 1/4th of processor rate
 - Each link is 1.4 Gb/s at target 700-MHz clock rate (175 MB/s)
 - High internal switch connectivity to keep all links busy
 - › External switch input links: 6 at 175 MB/s each (1,050 MB/s aggregate)
 - › External switch output links: 6 at 175 MB/s each (1,050 MB/s aggregate)
 - › Internal datapath crossbar input links: 12 at 175 MB/s each
 - › Internal datapath crossbar output links: 6 at 175 MB/s each
 - › Switch injection links: 7 at 175 MBps each (2 cores, each with 4 FIFOs)
 - › Switch reception links: 12 at 175 MBps each (2 cores, each with 7 FIFOs)

