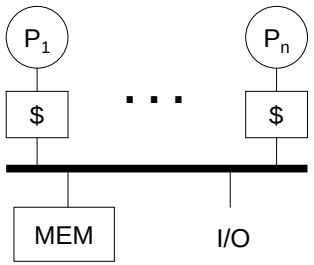
Scalable Multiprocessors

Topics

- Scaling issues
- Supporting programming models
- Network interface
- Interconnection network
- Considerations in Bluegene/L design

Limited Scaling of a Bus

Characteristic	Bus
Physical Length	~ 1 ft
Number of Connections	fixed
Maximum Bandwidth	fixed
Interface to Comm. medium	memory
Global Order	arbitration
Protection	Virtual ⇒ physical
Trust	total
OS	single
comm. abstraction	HW



- Scaling limit
- Close coupling among components

Comparing with a LAN

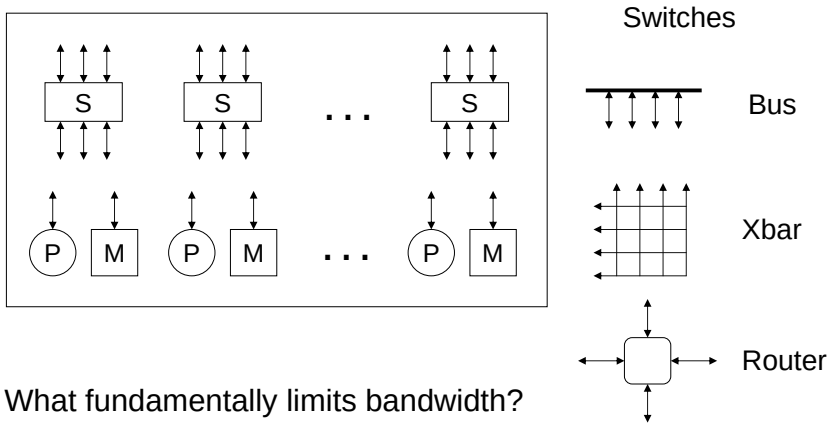
Characteristic	Bus	LAN
Physical Length	~ 1 ft	KM
Number of Connections	fixed	many
Maximum Bandwidth	fixed	???
Interface to Comm. medium	memory	peripheral
Global Order	arbitration	???
Protection	Virtual ⇒ physical	OS
Trust	total	little
OS	single	independent
comm. abstraction	HW	SW

- No clear limit to physical scaling, little trust, no global order, consensus difficult to achieve.
- Independent failure and restart

Scalable Computers

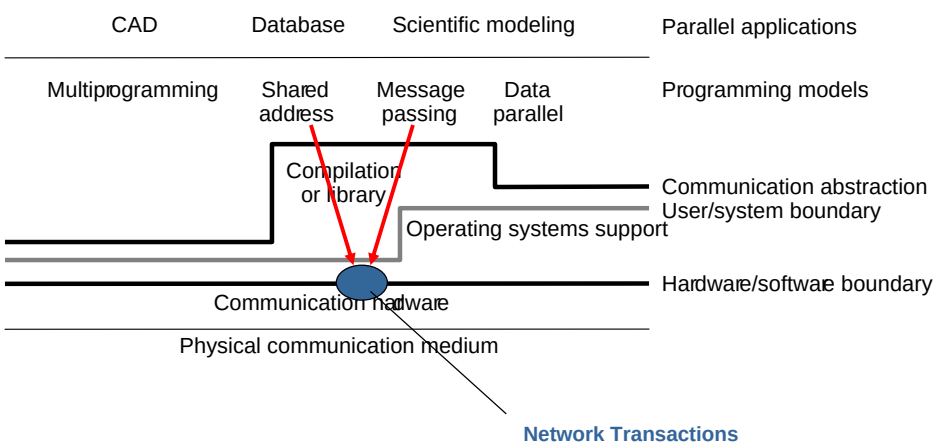
- What are the design trade-offs for the spectrum of machines between?
  - Specialize or commodity nodes?
  - Capability of node-to-network interface
  - Supporting programming models?
- What does scalability mean?
  - Avoid inherent design limits on resources
  - Bandwidth increases with n
  - Latency does not increase with n
  - Cost increases slowly with n

Bandwidth Scalability

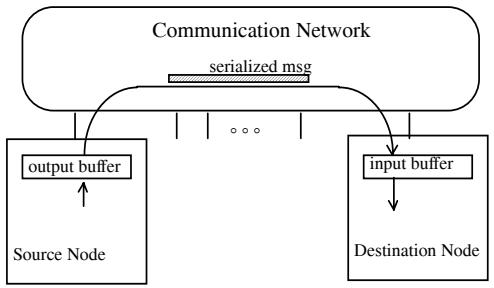


- What fundamentally limits bandwidth?
  - single set of wires
- Must have many independent wires
- Connect modules through switches

Programming Models Realized by Protocols

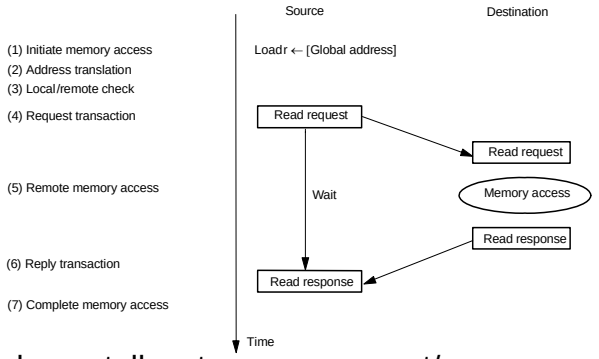


Network Transaction Primitive



- one-way transfer of information from a source output buffer to a dest. input buffer
  - causes some action at the destination
  - occurrence is not directly visible at source
- deposit data, state change, reply

Shared Address Space Abstraction



- Fundamentally a two-way request/response protocol
  - writes have an acknowledgement
- Issues
  - fixed or variable length (bulk) transfers
  - remote virtual or physical address, where is action performed?
  - deadlock avoidance and input buffer full

The Fetch Deadlock Problem

- Even if a node cannot issue a request, it must sink network transactions.
- Incoming transaction may be a request, which will generate a response.
- Closed system (finite buffering)

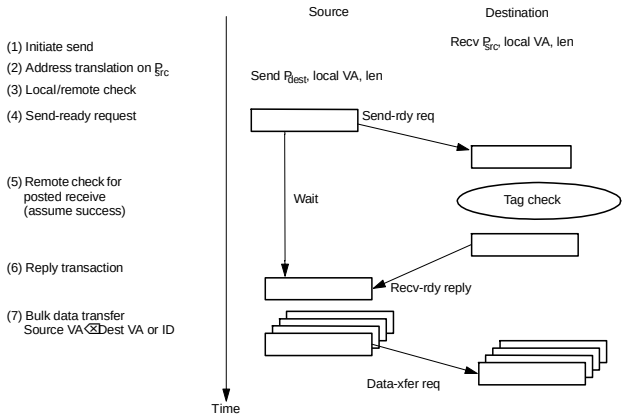
Key Properties of SAS Abstraction

- Source and destination data addresses are specified by the source of the request
  - a degree of logical coupling and trust
- no storage logically “outside the application address space(s)”
  - But it may employ temporary buffers for transport
- Operations are fundamentally request / response
- Remote operation can be performed on remote memory
  - logically does not require intervention of the remote processor

Message passing

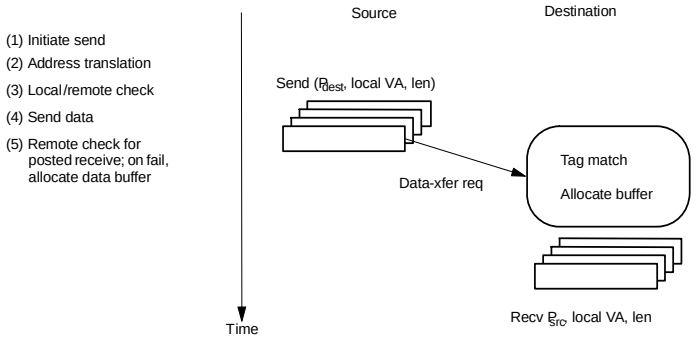
- Bulk transfers
- Complex synchronization semantics
  - more complex protocols
  - More complex action
- Synchronous
  - Send completes after matching recv and source data sent
  - Receive completes after data transfer complete from matching send
- Asynchronous
  - Send completes after send buffer may be reused

Synchronous Message Passing



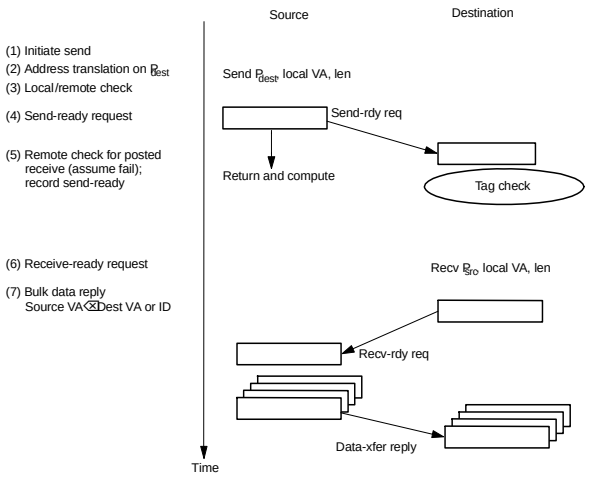
- Constrained programming model.
- Deterministic! What happens when threads added?
- Destination contention very limited.

Asynchronous Message Passing: Optimistic



- More powerful programming model
- Wildcard receive => non-deterministic
- Storage required within msg layer?

Asynchronous MSG Passing: Conservative



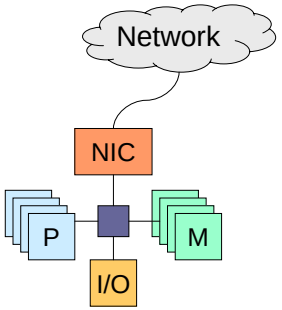
- Where is the buffering?
- Contention control? Receiver initiated protocol?
- Short message optimizations

Key Features of Msg Passing Abstraction

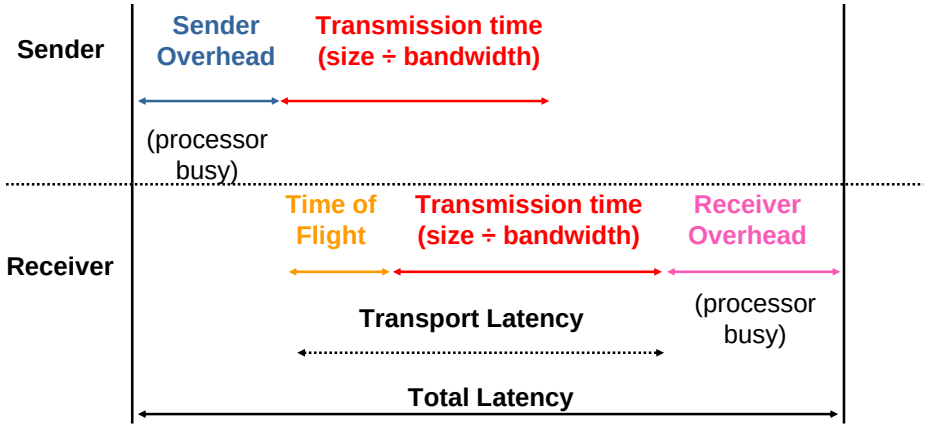
- Source knows send data address, destination knows receive data address
  - after handshake they both know
- Arbitrary storage “outside the local address spaces”
  - may post many sends before any receives
  - non-blocking asynchronous sends reduces the requirement to an arbitrary number of descriptors
    - fine print says these are limited too
- Fundamentally a 3-phase transaction
  - includes a request / response
  - can use optimistic 1-phase in limited “Safe” cases

Network Interface

- Transfer between local memory and NIC buffers
  - SW translates VA ⇔ PA
  - SW initiate DMA
  - SW does buffer management
  - NIC initiates interrupts on receive
  - Provides protection
- Transfer between NIC buffers and the network
  - Generate packets
  - Flow control with the network



Network Performance Metrics



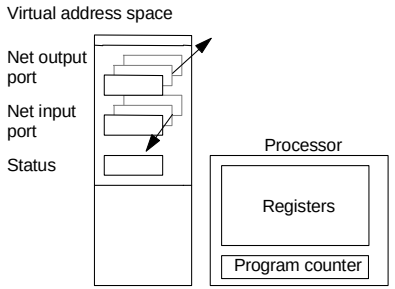
Total Latency = Sender Overhead + Time of Flight + Message Size ÷ BW + Receiver Overhead

Includes header/trailer in BW calculation?

Protected User-Level Communication

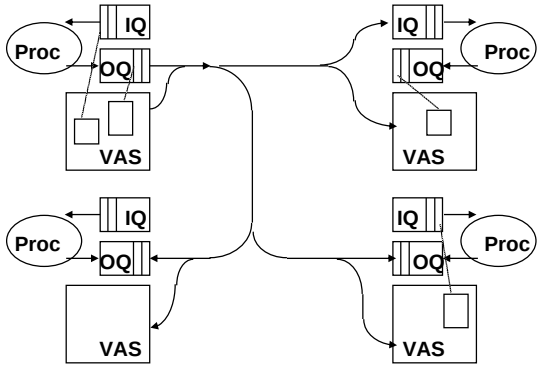
- Traditional NIC (e.g. Ethernet) requires OS kernel to initiate DMA and to manage buffers
  - Prevent apps from crashing OS or other apps
  - Overhead is high (how high?)
- Multicomputer or multiprocessor NICs
  - OS maps VA to PA buffers
  - Apps initiate DMAs using VA addresses or handles of descriptors
  - NIC use mapped PA buffers to perform DMAs
- Examples
  - Research: Active message, UDMA
  - Industry: VIA and RDMA

User Level Network ports



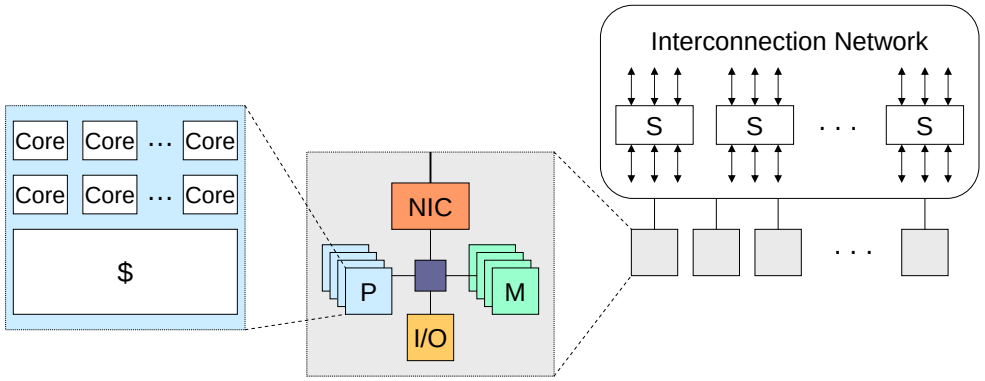
- Appears to user as logical message queues plus status
- What happens if no user pop?

User Level Abstraction



- Any user process can post a transaction for any other in protection domain
  - communication layer moves OQ_{src} –> IQ_{dest}
  - may involve indirection: VAS_{src} –> VAS_{dest}

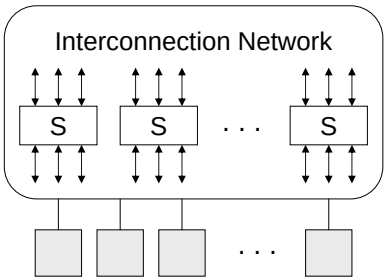
Generic Multiprocessor Architecture



Network characteristics
- Network bandwidth: on-chip and off-chip interconnection network
- Bandwidth demands: independent and communicating threads/processes
- Latency: local and remote

Scalable Interconnection Network

- At core of parallel computer architecture
- Requirements and trade-offs at many levels
  - Elegant mathematical structure
  - Deep relationships to algorithm structure
  - Managing many traffic flows
  - Electrical / optical link properties
- Little consensus
  - interactions across levels
  - Performance metrics
  - Cost metrics
  - Workload
- Need holistic understanding



Requirements from Above

- Communication-to-computation ratio
  - ⇒ bandwidth that must be sustained for given computational rate
  - Traffic localized or dispersed?
  - Bursty or uniform?
- Programming Model
  - Protocol
  - Granularity of transfer
  - Degree of overlap (slackness)
- The job of a parallel machine's interconnection network is to transfer information from source node to destination node in support of network transactions that realize the programming model

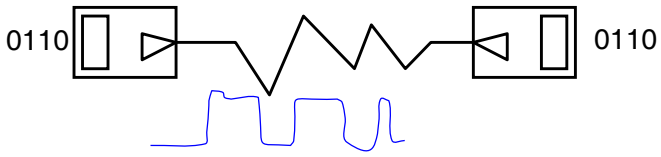
Characteristics of A Network

- Topology (what)
  - Physical interconnection structure of the network graph
  - Direct: node connected to every switch
  - Indirect: nodes connected to specific subset of switches
- Routing Algorithm (which)
  - Restricts the set of paths that messages may follow
  - Many algorithms with different properties
- Switching Strategy (how)
  - How data in a message traverses a route
  - Store and forward vs. cut through
- Flow Control Mechanism (when)
  - When a message or portions of it traverse a route
  - What happens when traffic is encountered?

Basic Definitions

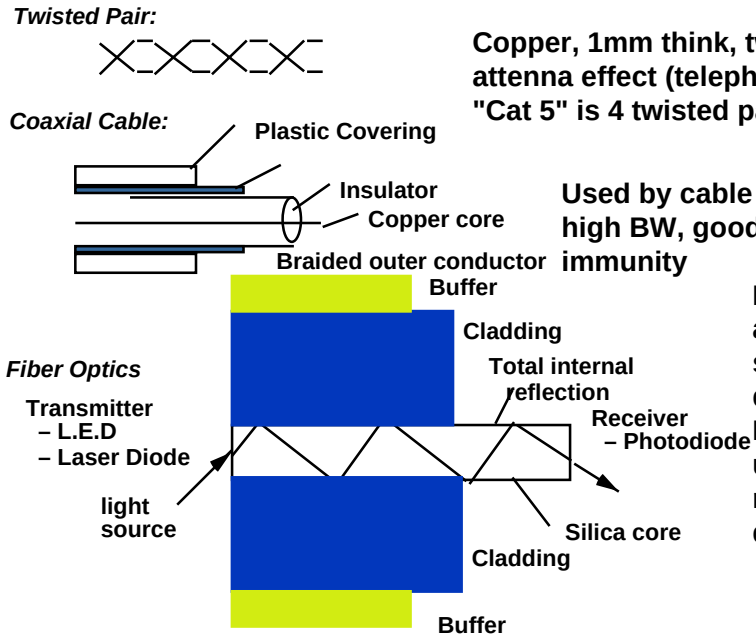
- Network interface
  - Communication between a node and the network
- Links
  - Bundle of wires or fibers that carries signals
- Switches
  - Connects fixed number of input channels to fixed number of output channels

Network Basics



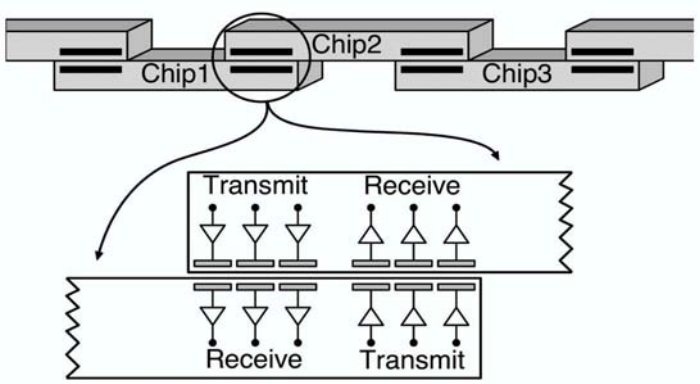
- Link made of some physical media
  - wire, fiber, air
- with a transmitter (tx) on one end
  - converts digital symbols to analog signals and drives them down the link
- and a receiver (rx) on the other
  - captures analog signals and converts them back to digital signals
- tx+rx called a transceiver

Traditional Network Media



Emerging Media

- Proximity project (Sun Microsystems)
  - Potentially deliver TB/sec between chips
  - Microscopic metal pads coated with a micron-thin layer of insulator to protect the chip from static electricity
  - Two chips contact each other



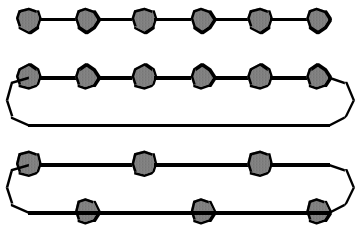
Networks in Parallel Machines

- Some old machines

Machine	Topology	Cycle Time (ns)	Channel Width (bits)	Routing Delay (cycles)	Flit (data bits)
nCUBE/2	Hypercube	25	1	40	32
TMC CM-5	Fat-Tree	25	4	10	4
IBM SP-2	Banyan	25	8	5	16
Intel Paragon	2D Mesh	11.5	16	2	16
Meiko CS-2	Fat-Tree	20	8	7	8
CRAY T3D	3D Torus	6.67	16	2	16
DASH	Torus	30	16	2	16
J-Machine	3D Mesh	31	8	2	8
Monsoon	Butterfly	20	16	2	16
SGI Origin	Hypercube	2.5	20	16	160
Myricom	Arbitrary	6.25	16	50	16

- New machines
  - Cray XT3 and XT4: 3D torus, 7GB/sec each link
  - IBM Bluegene/L: 3D torus, 1.4Gb/sec each link

Linear Arrays and Rings



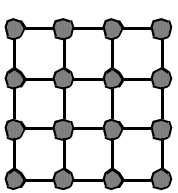
Linear Array

Torus

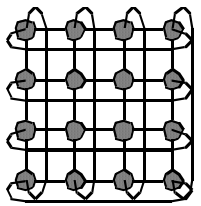
Torus arranged to use short wires

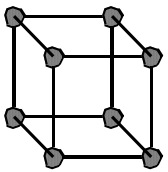
- Linear Array
  - Diameter?
  - Average Distance?
  - Bisection bandwidth?
  - Route A -> B given by relative address R = B-A
- Torus?
- Examples: FDDI, SCI, FiberChannel Arbitrated Loop, KSR1

Multidimensional Meshes and Tori



2D Grid





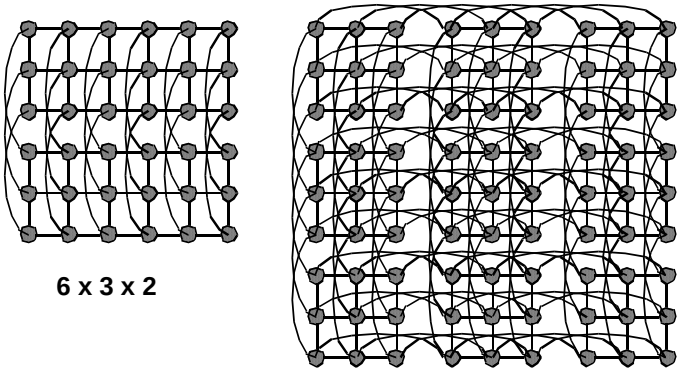
3D Cube

- *d*-dimensional array
  - $n = k_{d-1} X ... X k_0$ nodes
  - described by *d*-vector of coordinates $(i_{d-1}, ..., i_0)$
- *d*-dimensional *k*-ary mesh: $N = k^d$
  - $k = \sqrt[d]{N}$
  - described by *d*-vector of radix k coordinate
- *d*-dimensional *k*-ary torus (or *k*-ary *d*-cube)?

Properties

- Routing
  - relative distance: R = ($b_{d-1} - a_{d-1}$, ... , $b_0 - a_0$)
  - traverse ri = $b_i - a_i$ hops in each dimension
  - *dimension-order routing*
- Average Distance — Wire Length?
  - d x 2k/3 for mesh
  - dk/2 for cube
- Degree?
- Bisection bandwidth? — Partitioning?
  - k^{d-1} bidirectional links
- Physical layout?
  - 2D in O(N) space — Short wires
  - higher dimension?

Embeddings in Two Dimensions

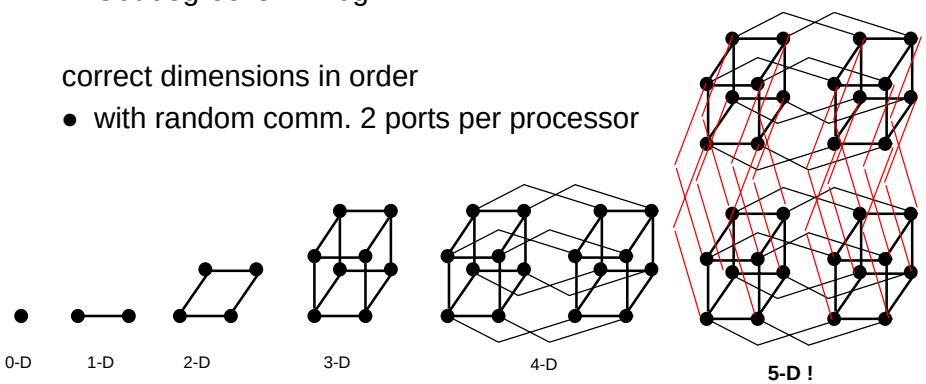


- Embed multiple logical dimension in one physical dimension using long wires

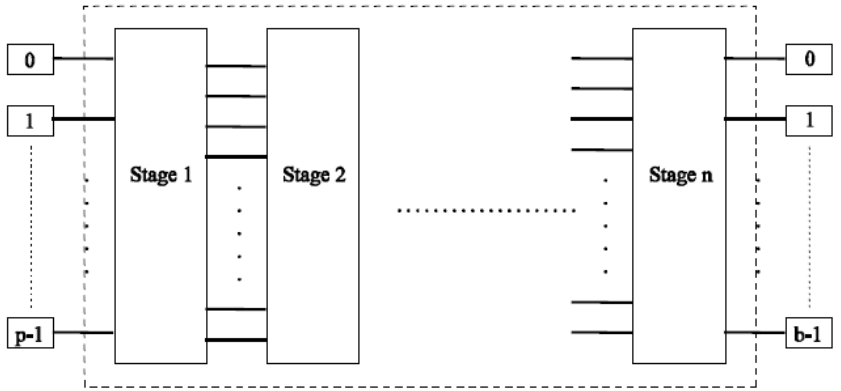
Hypercubes

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

correct dimensions in order
- with random comm. 2 ports per processor

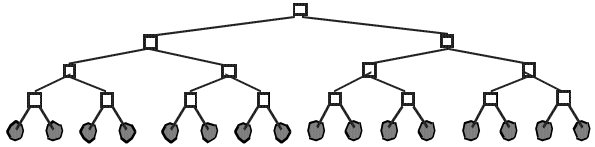


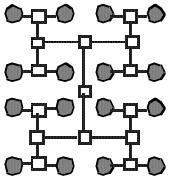
Multistage Network



- Routing from left to right
- Typically n = log(p)

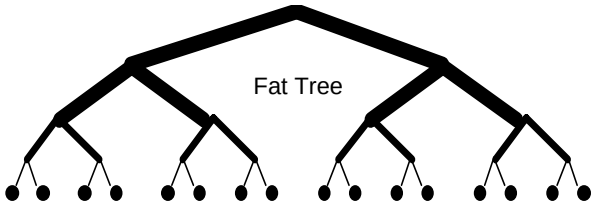
Trees

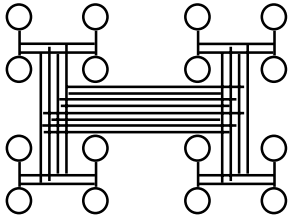




- Diameter and ave distance logarithmic
  - k-ary tree, height d = $\log_k N$
  - address specified d-vector of radix k coordinates describing path down from root
- Fixed degree
- Route up to common ancestor and down
  - R = B xor A
  - let i be position of most significant 1 in R, route up i+1 levels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

Fat-Trees





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

Topology Summary

Topology	Degree	Diameter	Ave Dist	Bisection	D (D ave) @ P=1024	
1D Array	2	N-1	N/3	1	huge	
1D Ring	2	N/2	N/4	2		
2D Mesh	4	2 ($N^{1/2}$ - 1)	2/3 $N^{1/2}$	$N^{1/2}$	63 (21)	
2D Torus	4	$N^{1/2}$	1/2 $N^{1/2}$	$2N^{1/2}$	32 (16)	
k-ary n-cube	2n	nk/2	nk/4	nk/4	15 (7.5)	@n=3
Hypercube	n =log N		n	n/2	N/2	10 (5)

- All have some “bad permutations”
  - many popular permutations are very bad for meshes (transpose)
  - randomness in wiring or routing makes it hard to find a bad one!

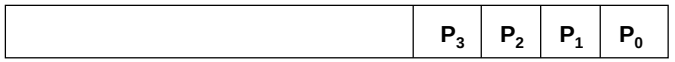
How Many Dimensions?

- n = 2 or n = 3
  - Short wires, easy to build
  - Many hops, low bisection bandwidth
  - Requires traffic locality
- n >= 4
  - Harder to build, more wires, longer average length
  - Fewer hops, better bisection bandwidth
  - Can handle non-local traffic
- k-ary d-cubes provide a consistent framework for comparison
  - N = kd
  - scale dimension (d) or nodes per dimension (k)
  - assume cut-through

Routing Mechanism

- Need to select output port for each input packet
  - in a few cycles
- Simple arithmetic in regular topologies
  - ex: Dx, Dy routing in a grid
    - west (-x) Dx < 0
    - east (+x) Dx > 0
    - south (-y) Dx = 0, Dy < 0
    - north (+y) Dx = 0, Dy > 0
    - processor Dx = 0, Dy = 0
- Reduce relative address of each dimension in order
  - Dimension-order routing in k-ary d-cubes
  - e-cube routing in n-cube

Routing Mechanism (cont)



- Source-based
  - message header carries series of port selects
  - used and stripped en route
  - *CRC? Packet Format?*
- Table-driven
  - message header carried index for next port at next switch
    - o = R[i]
  - table also gives index for following hop
    - o, I' = R[i]
  - ATM, HPPI

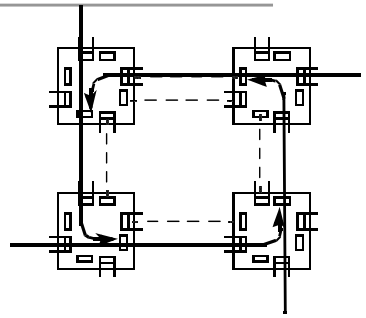
Properties of Routing Algorithms

- Deterministic
  - route determined by (source, dest), not intermediate state (i.e. traffic)
- Adaptive
  - route influenced by traffic along the way
- Minimal
  - only selects shortest paths
- Deadlock free
  - no traffic pattern can lead to a situation where no packets mover forward

Routing Messages

- Shared Media
  - Broadcast to everyone
- Options:
  - Source-based routing: message specifies path to the destination (changes of direction)
  - Destination-based routing: message specifies destination, switch must pick the path
    - deterministic: always follow same path
    - adaptive: pick different paths to avoid congestion, failures
    - Randomized routing: pick between several good paths to balance network load

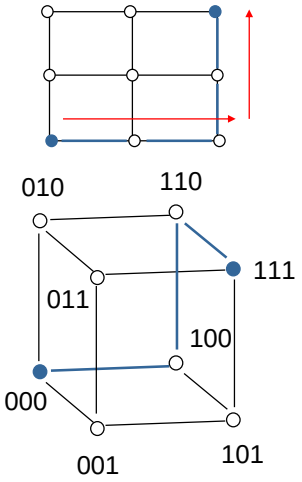
Deadlock Freedom



- How can it arise?
  - necessary conditions:
    - shared resource
    - incrementally allocated
    - non-preemptible
  - think of a channel as a shared resource that is acquired incrementally
    - source buffer then destination buffer
    - channels along a route
- How do you avoid it?
  - constrain how channel resources are allocated
- How to prove that a routing algorithm is deadlock free

Deterministic Routing Examples

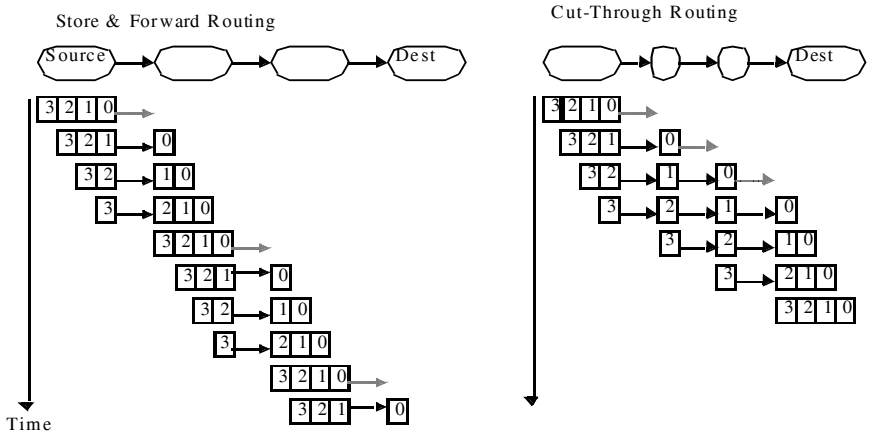
- Mesh: dimension-order routing
  - (x1, y1) -> (x2, y2)
  - first x = x2 - x1,
  - then y = y2 - y1,
- Hypercube: edge-cube routing
  - X = xox1x2 . . .xn -> Y = yoy1y2 . . .yn
  - R = X xor Y
  - Traverse dimensions of differing address in order
- Tree: common ancestor
- Deadlock free?



Store and Forward vs. Cut-Through

- Store-and-forward
  - each switch waits for the full packet to arrive in switch before sending to the next switch
  - Applications: LAN or WAN
- Cut-through routing
  - switch examines the header, decides where to send the message
  - starts forwarding it immediately

Store&Forward vs Cut-Through Routing



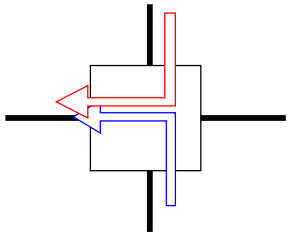
- h(n/b + D) vs n/b + h D
- what if message is fragmented?

Cut-Through vs. Wormhole Routing

- In wormhole routing, when head of message is blocked, message stays strung out over the network, potentially blocking other messages (needs only buffer the piece of the packet that is sent between switches).
- Cut through routing lets the tail continue when head is blocked, accordioning the whole message into a single switch. (Requires a buffer large enough to hold the largest packet).

- References
  - P. Kermani and L. Kleinrock, Virtual cut-through: A new computer communication switching technique. Computer Networks, vol. 3, pp. 267-286, 1979.
  - W.J. Dally and C.L. Seitz, "Deadlock-Free Message Routing in Multiprocessor Interconnection Networks," *IEEE Trans. Computers,* Vol. C-36, No. 5, May 1987, pp. 547-553

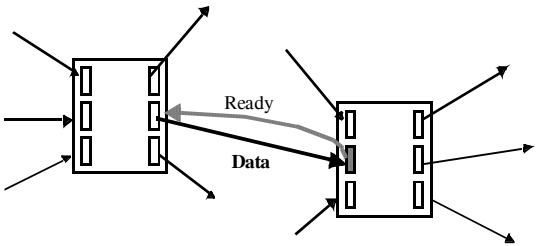
Contention



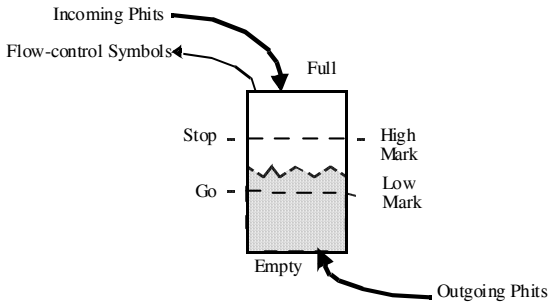
- Two packets trying to use the same link at same time
  - limited buffering
  - drop?
- Most parallel mach. networks block in place
  - link-level flow control
  - tree saturation
- Closed system - offered load depends on delivered

Flow Control

- What do you do when push comes to shove?
  - ethernet: collision detection and retry after delay
  - FDDI, token ring: arbitration token
  - TCP/WAN: buffer, drop, adjust rate
  - any solution must adjust to output rate
- Link-level flow control

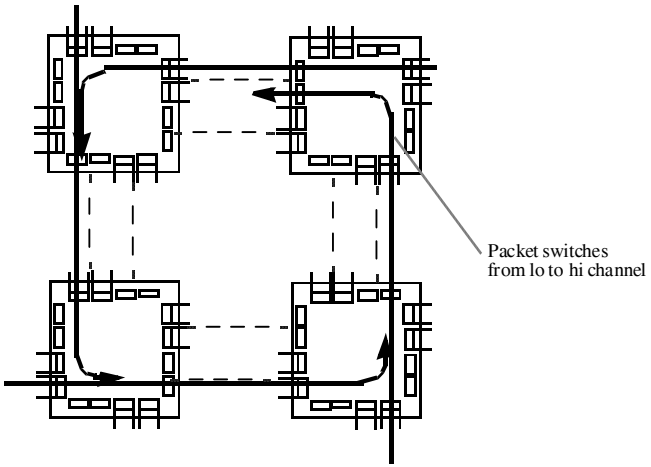


Smoothing the flow



- How much slack do you need to maximize bandwidth?

Virtual Channels



- W.J. Dally, "Virtual-Channel Flow Control," Proceedings of the 17th annual international symposium on Computer Architecture, p.60-68, May 28-31, 1990

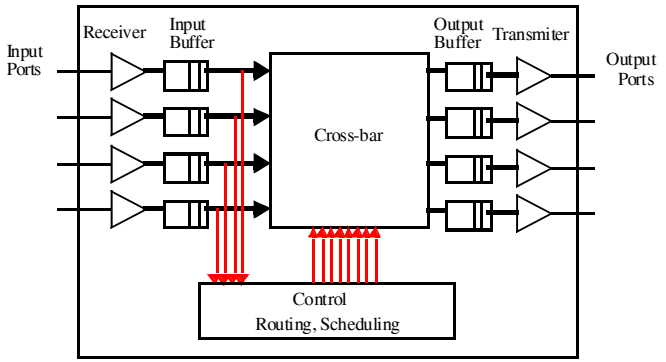
Bandwidth

- What affects local bandwidth?
  - packet density b x n/(n + ne)
  - routing delay b x n / (n + ne + wD)
  - contention
    - endpoints
    - within the network
- Aggregate bandwidth
  - bisection bandwidth
    - sum of bandwidth of smallest set of links that partition the network
  - total bandwidth of all the channels: Cb
  - suppose N hosts issue packet every M cycles with ave dist
    - each msg occupies h channels for l = n/w cycles each
    - C/N channels available per node
    - link utilization r = MC/Nhl < 1

Some Examples

- T3D: Short, Wide, Synchronous (300 MB/s)
  - 24 bits
    - 16 data, 4 control, 4 reverse direction flow control
  - single 150 MHz clock (including processor)
  - flit = phit = 16 bits
  - two control bits identify flit type (idle and framing)
    - no-info, routing tag, packet, end-of-packet
- T3E: long, wide, asynchronous (500 MB/s)
  - 14 bits, 375 MHz
  - flit = 5 phits = 70 bits
    - 64 bits data + 6 control
  - switches operate at 75 MHz
  - framed into 1-word and 8-word read/write request packets
- Cost = f(length, width) ?

Switches

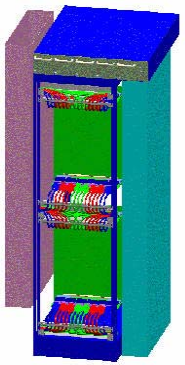


- With virtual channels, a buffer becomes multiple buffers
- Who selects a virtual channel?

Switch Components

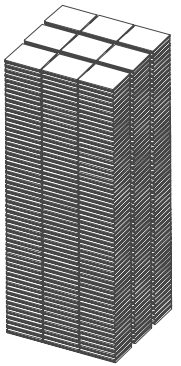
- Output ports
  - transmitter (typically drives clock and data)
- Input ports
  - synchronizer aligns data signal with local clock domain
  - essentially FIFO buffer
- Crossbar
  - connects each input to any output
  - degree limited by area or pinout
- Buffering
- Control logic
  - complexity depends on routing logic and scheduling algorithm
  - determine output port for each incoming packet
  - arbitrate among inputs directed at same output

Bluegene/L: A Low Power Design



BG/L
2048 processors

20.1 kW



450 Thinkpads

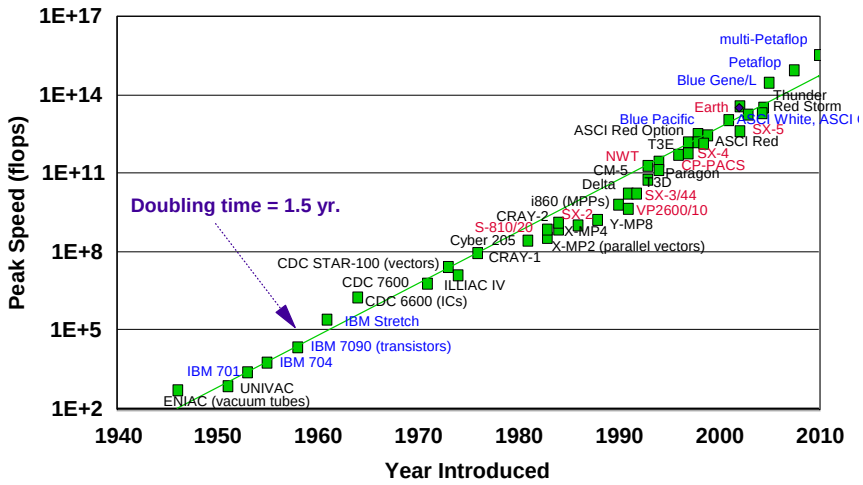
20.3 kW

(LS Mok,4/2002)

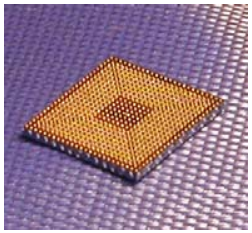
Comparing Systems

	ASCI White	ASCI Q	Earth Simulator	Blue Gene/L
Machine Peak (TF/s)	12.3	30	40.96	367
Total Mem. (TBytes)	8	33	10	32
Footprint (sq ft)	10,000	20,000	34,000	2,500
Power (MW)	1	3.8	6-8.5	1.5
Cost ($M)	100	200	400	100
# Nodes	512	4096	640	65,536
MHz	375	1000	500	700

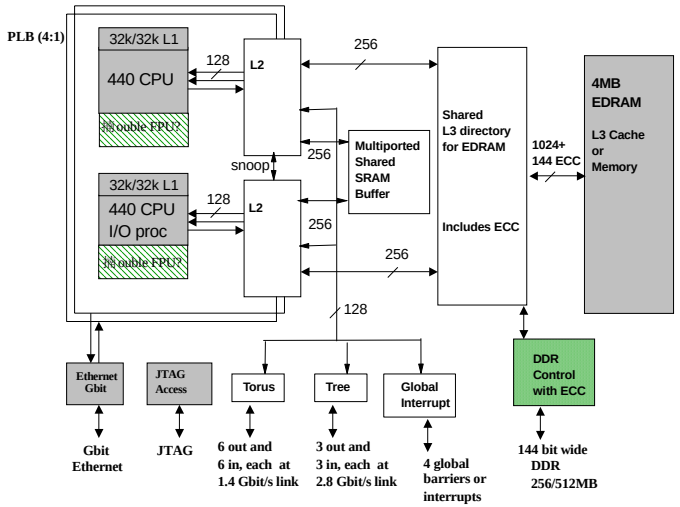
Supercomputer Peak Performance



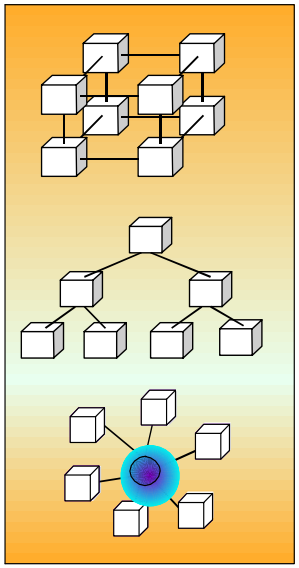
BlueGene/L Compute ASIC



- IBM CU-11, 0.13 µm
- 11 x 11 mm die size
- 25 x 32 mm CBGA
- 474 pins, 328 signal
- 1.5/2.5 Volt



BlueGene/L Interconnection Networks



3 Dimensional Torus

- Interconnects all compute nodes (65,536)
- Virtual cut-through hardware routing
- 1.4Gb/s on all 12 node links (2.1 GB/s per node)
- 1 µs latency between neighbors, 5 µs to the farthest
- 4 µs latency for one hop with MPI, 10 µs to the farthest
- Communications backbone for computations
- 0.7/1.4 TB/s bisection bandwidth, 68TB/s total bandwidth

Global Tree

- One-to-all broadcast functionality
- Reduction operations functionality
- 2.8 Gb/s of bandwidth per link
- Latency of one way tree traversal 2.5 µs
- ~23TB/s total binary tree bandwidth (64k machine)
- Interconnects all compute and I/O nodes (1024)

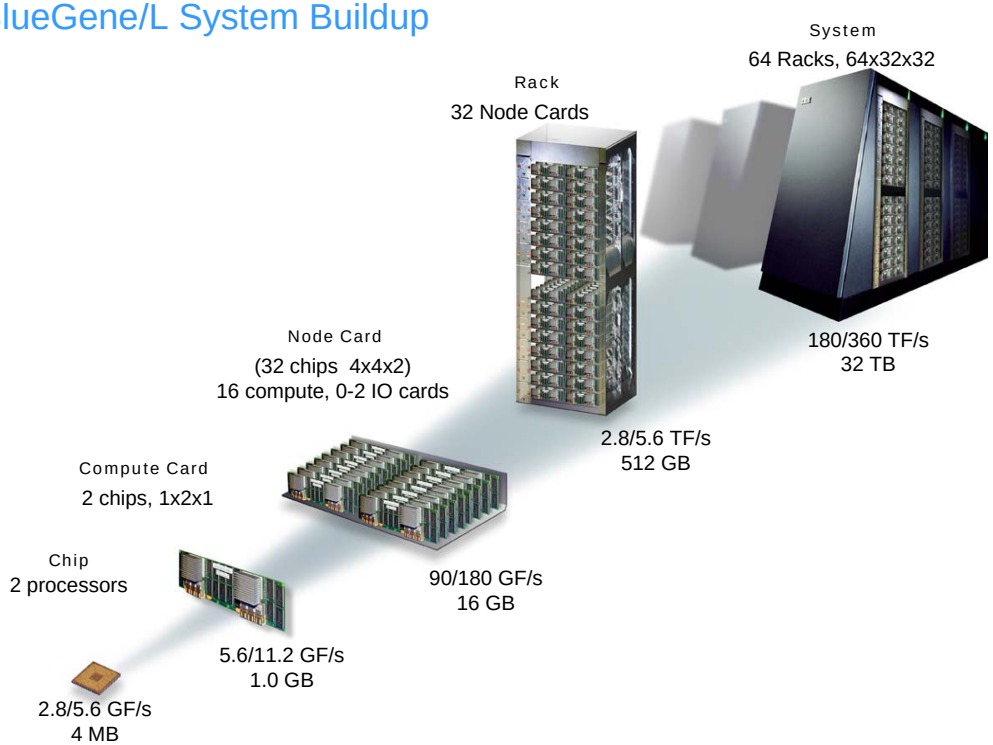
Low Latency Global Barrier and Interrupt

- Latency of round trip 1.3 µs

Ethernet

- Incorporated into every node ASIC
- Active in the I/O nodes (1:64)
- All external comm. (file I/O, control, user interaction, etc.)

BlueGene/L System Buildup



512 Way Bluegene/L

Bluegene/L: 16384 nodes (IBM Rochester)

Summary

- Scalable multicomputers must consider scaling issues in bandwidth, latency, power and cost
- Network interface design
  - Substantially reduce the send and receive overheads
- Networks need to support programming models and applications well
- Many network topologies have been studied, the most common ones are meshes, tori, tree and multi-stage
- Current network routers use virtual cut through, wormhole routing with virtual channels
- New-generation scalable computers must consider power scaling