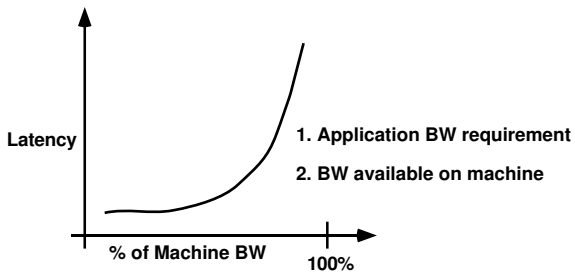
Scalable Communication Networks

- Perspective on bandwidth and latency
- Design of Interconnection Networks
- Assessing Trade-offs
- Analysis of k-ary n-cubes

© Culler 1995

Bandwidth and Latency



- Latencies remain reasonable only as long as 1. < 2.
- Goal is to both reduce 1. and increase 2. without making cost too high.

© Culler 1995

Observations

- In uniprocessors caches act as both BW and Latency reducers.
- In multiprocessors BW can be measured at two points:
  - **local bandwidth**
  - **bisection bandwidth**
- Multiprocessor applications also have two kinds of BW requirements:
  - **local bandwidth**
  - **global bandwidth (true commn., false sharing, large working set)**
- Since network costs may not scale linearly, solutions for small-scale and large-scale machines may be different.

© Culler 1995

Reducing Application BW Demand

- Should be our first focus
- Solution techniques:
  - **Algorithmic techniques to reduce working set size and to increase cache locality**
  - **Data/Task placement so that private data in local memory**
  - **Data/Task placement so shared data in local or close-by memory**
- Use of dynamic data replication/migration to achieve above

© Culler 1995

Providing High Bandwidth

- Cost is a primary driving factor
- Solution techniques:
  - **Interleaved memory systems**
  - **new memory technologies (SDRAMS, RDRAMS, ...)**
  - **distributed memory architecture**
  - **network design (topology, link width and speeds, contention management, ...)**

© Culler 1995

Design Space for Interconnection Networks

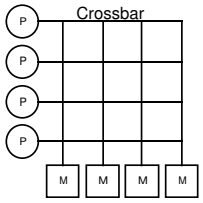
- Design Options:
  - **Topology**
  - **Routing**
  - **Direct vs. Indirect**
  - **Physical implementation**

- Evaluation Criteria:
  - **Latency**
    - » **number of hops**
    - » **wire length & switch delay**
  - **Bisection Bandwidth**
    - » **BW across cut that divides nodes into two halves**
  - **Contention and hot-spot behavior**
  - **Partitionability**
  - **Cost and scalability**
  - **Fault tolerance**

© Culler 1995

Basics

- Buses
  - **Simple and cost-effective for small-scale multiprocessors**
  - **Not scalable (limited bandwidth; electrical complications)**



- Cross-bars

  Each port has link to every other port

  + Low latency and high throughput

  - Cost grows as O(N^2) so not very scalable.

  - Difficult to arbitrate and to get all data lines into and out of a centralized crossbar.

  Used in small-scale MPs (e.g., C.mmp)

  & building block for other networks

© Culler 1995

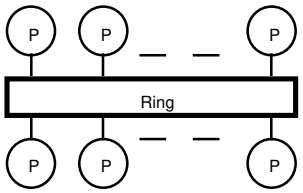
Rings

- Cheap: Cost is O(N).
- Short Point-to-point wires, simple switches and pipelining can be used to make them very fast.

+ High overall bandwidth

- Low bisection BW
- High latency O(N)
- Poor partitioning
- Very poor fault tolerance

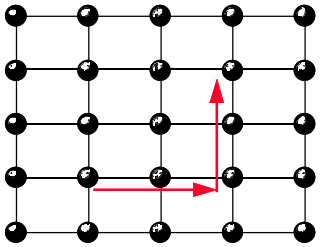
• Examples: KSR machine



© Culler 1995

Meshes

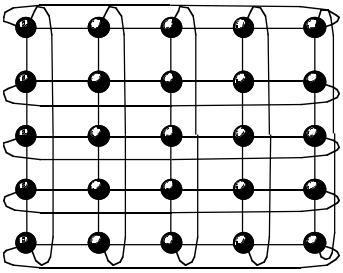
- Latency
  - **2 $\sqrt{n}$ hops average distance**
  - **Short wires**
- $\sqrt{n}$ bisection bandwidth
- Typical route: Δx, Δy
- Poor fault tolerance
- Serious Contention Problems
  - **ex: transpose**
- Requires locality for good performance
- Easy to partition
- Easy to build

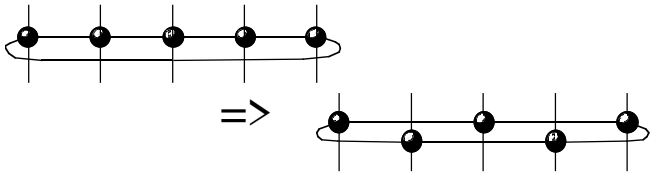


© Culler 1995

2D Torus

- Latency
  - **About half as many hops**
  - **Short wires**
- Twice the bisection bandwidth
- Slightly better fault tolerance
- Hard to partition!
- Still easy to build

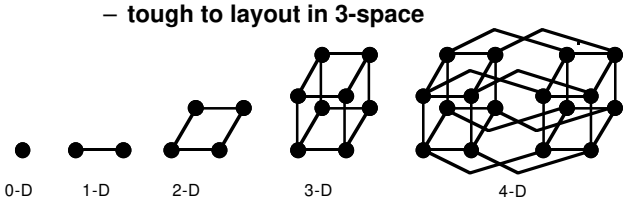




© Culler 1995

Hypercubes

- Also called binary n-cubes. # of nodes = N = 2^n.
- Latency
  - **O(logN) Hops**
- Good bisection BW
- Complexity
  - **Out degree of PE is O(logN)**
  - **tough to layout in 3-space**

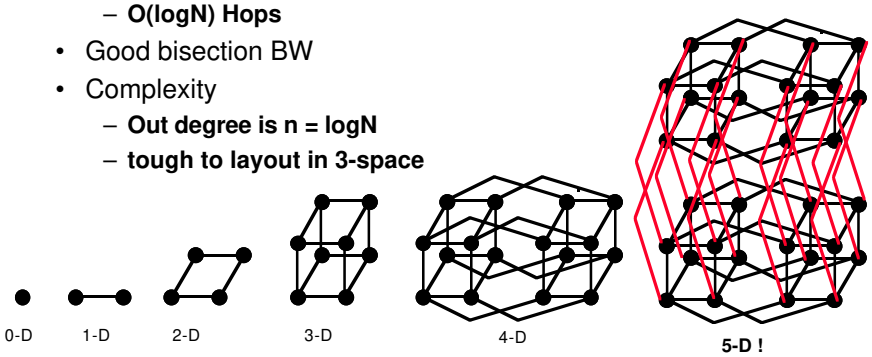


- Popular in early message-passing computers
  - **(e.g., intel iPSC, NCUBE)**
- Used as direct network ==> emphasizes locality

© Culler 1995

Hypercubes

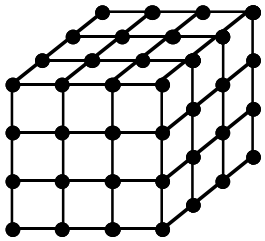
- Also called binary n-cubes. # of nodes = N = 2^n.
- Latency
  - **O(logN) Hops**
- Good bisection BW
- Complexity
  - **Out degree is n = logN**
  - **tough to layout in 3-space**



- Route from A to B by crossing dimension A xor B
  - **correct dimensions in order**
  - **with random comm. 2 ports per processor**

© Culler 1995

k-ary n-cubes



4-ary 3-cube

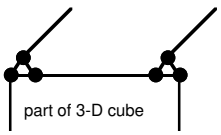
- Generalization of hypercubes
  - **k nodes in each dimension)**
- Total # of nodes = N = k^n.
- out degree = n
- k > 2 reduces # of channels at bisection, thus allowing for wider channels but more hops.

=> Will discuss advantages in greater detail later.

© Culler 1995

Cube Connected Cycles

- Another variation of hypercubes
- Addresses node out-degree problem without increasing latency much
- Each node in 2-ary k-cube replaced by k-ring
- Latency (# of hops) is less than (2.5 • k).



© Culler 1995

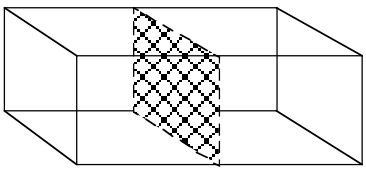
How Many Dimensions?

- 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**

© Culler 1995

In the 3D world

- For n nodes, bisection area is $O(n^{2/3})$



- For large n, bisection bandwidth is limited to $O(n^{2/3})$
  - **Bill Dally, IEEE TPDS, [Dal90a]**
  - **For fixed bisection bandwidth, low-dimensional k-ary n-cubes are better (otherwise higher is better)**
  - **i.e., a few short fat wires are better than many long thin wires**
  - **What about many long fat wires?**

© Culler 1995

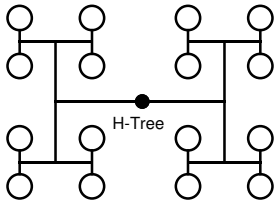
Multistage Logarithmic Networks

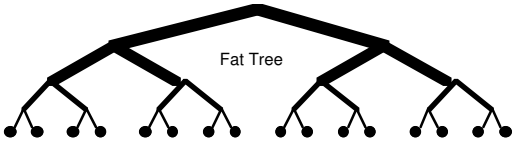
- Cost is O(NlogN); latency is O(logN); throughput is O(N).
- Generally indirect networks.
- Many variations exist (Omega, Butterfly, Benes, ...).
- Used in many machines: BBN Butterfly, IBM RP3, ...

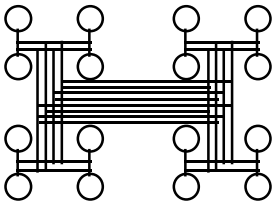
© Culler 1995

Trees

- Cheap: Cost is O(N).
- Latency is O(logN).
- Easy to layout as planar graphs
  - **(e.g., H-Trees)**
- Up/Down routing
- Good partitioning into subtrees
- For random permutations, root can become bottleneck.
- To avoid root being bottleneck => Fat-Trees (used in CM-5)
  - **channels are wider as you move towards root.**
  - **actually more of them**



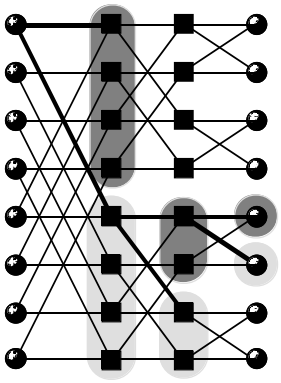




© Culler 1995

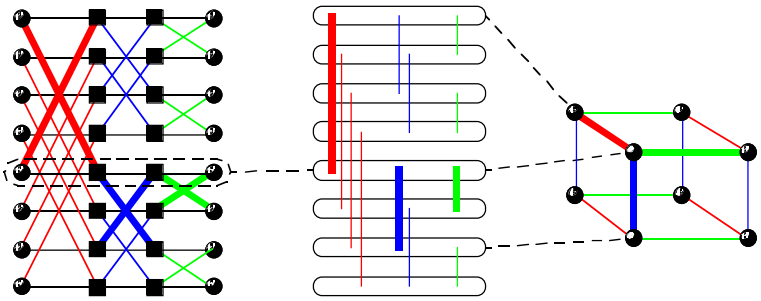
Butterflies

- Split into up & down at each stage
  - **A xor B in order**
- Log n hops
- Bisection bandwidth: O(n)
- Some long wires
- No fault tolerance
- No locality
- Easy to partition



© Culler 1995

Relationship to Hypercubes



- Wiring is isomorphic
- Except that Butterfly always takes log n steps

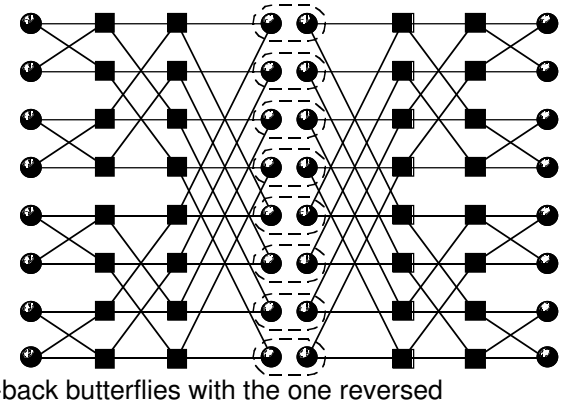
© Culler 1995

Butterflies vs. 3D Torus

- Butterfly has:
  - **More parts: O(n log n) routers**
  - **More edges: unidirectional vs. bidirectional**
  - **Worse fault tolerance**
  + **Lower latency unless significant locality**
  + **Much better bisection bandwidth for fixed cable size**
- Fixed bisection bandwidth implies worse performance
  - **more on this later**
- Both have significant hot-spot problems
- Simple partitioning into sub-machines
- Strictly more expensive, but potentially better performance

© Culler 1995

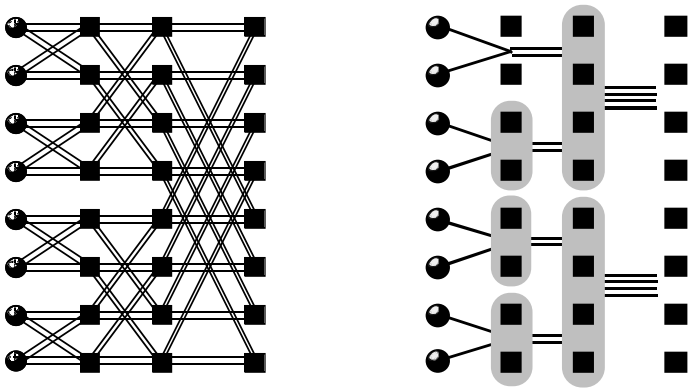
Benes Network



- Back-to-back butterflies with the one reversed
- Can route any permutation without blocking
- First half randomizes, second half routes
-
- Randomization is key recent innovation in networks

© Culler 1995

Fat Trees Really



- Folded Benes Network
  - **except route only to lowest common ancestor**
- Randomize on the way up!

© Culler 1995

Quick Comparison

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)

© Culler 1995

Analysis of k-ary n-cubes

- What can be built in VLSI is often wire limited:
  - **limited by wire density**
  - **speed by power delay**
  - **majority of power consumed in driving wires**

==> Need Wire Efficient Networks

- k-ary n-cubes represent a very wide family of networks
  - **hypercubes, n-D grids, Omega networks are all special cases**

© Culler 1995

Evaluation Criteria

- Traditionally networks compared under constant-channel-BW assumption:
  - **e.g., width = W = 1 bit; delay = Tc = 1 unit.**
- Traditional model favors higher dimensionality (HD) networks (e.g., hypercubes) over lower dimensionality (LD) networks (e.g., grids)
  - **W=1 not reasonable because HDNs require more wires**
  - **Tc = 1 not reasonable because HDNs require longer wires**

=> To account for more wires in HDNs:

  - **compare different topologies under constant bisection width**
  - **bisection width: # of wires cut when network is split in two halves**

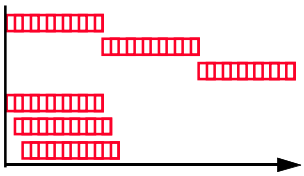
=> To account for longer wires, consider 3 delay models:

  - **Tc $\propto$ log(len): for small wires; delay due to charging capacitance**
  - **Tc $\propto$ len: for long wires; delay due to speed of light**
  - **Tc = 1: for short wires; delay due to switcing time**

© Culler 1995

Routing Strategies and Latency

- Store-and-Forward routing:
  - **Tsf = Tc • (D • L / W) ,**
    where L = msg length, D = # of hops
- Wormhole routing:
  - **Twh = Tc • (D + L / W)**
  - **# of hops is an additive**
    rather than multiplicative factor



=> *nota bena,* fewer hops vs. fatter wires !!!!
routing delay per hop also decreases with width

- Virtual Cut-Through routing:
  - **Older and similar to wormhole.**
  - **When blockage occurs, however, message is removed from network and buffered.**
- Deadlocks are avoided through use of *virtual channels* and by using a routing strategy that does not allow channel-dependency cycles.

© Culler 1995

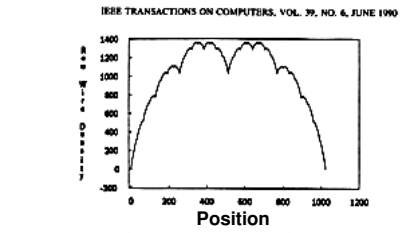
Comparison of k-ary n-cubes

- Assumptions:
  - **wormhole routing;**
  - **bisection normalized to hypercube with W=1.**
    B(2,n) = 2^n = N;
  - **network embedded in 2-D plane.**
- Channel Width:
  - **For k-ary n-cubes,**
    B(k,n) = 2•W•N / k or W(k,n) = k / 2.
- Example: Consider N = 2^16
  - **2-ary 16-cube ==> W = 1 bit;**
  - **16-ary 4-cube ==> W = 8 bits;**
  - **256-ary 2-cube ==> W = 128 bits.**
- This increase in channel width for LDNs is the major factor in its performing better.

© Culler 1995

Engineering Trade-offs

- Maximum Wire Density:
  - **Can be higher than bisection width for HDNs.**
  - **Thus layout complexity is even more than that predicted by bisection width.**
  - **k-ary 2-cubes are the simplest**



- Pin Requirements per Node:
  - **# of pins = 2n • k/2 = n • k.**
  - **n inputs, n outputs, W = k/2**
- Thus for N = 2^16
  - **2-ary 16 -cube: 32 pins**
  - **16-ary 4-cube: 64 pins**
  - **256-ary 2-cube: 512 pins**

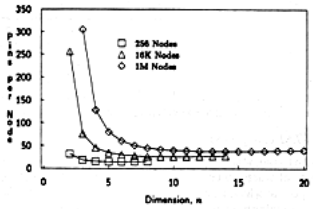


Fig. 7. Pin density versus dimension for 256, 16K, and 1M nodes.

© Culler 1995

Latency

- Latency:
  - **T = T-net + T-node**
  - **T-net = Tc (D + L / W)**
  - **D-avg = (k-1) / 2 • n;**
    dist of (k-1) / 2 in each dimension
  - **T-net minimized when D ~ L/W**
- Length of longest wire in 2-D:
  - **len = k^(n /2 -1) ==> bad for HDNs**

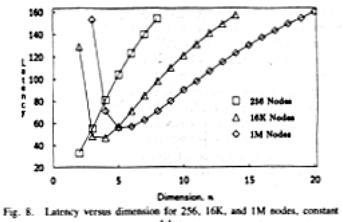


Fig. 8. Latency versus dimension for 256, 16K, and 1M nodes, constant delay.

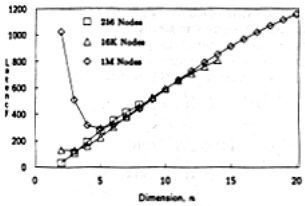


Fig. 9. Latency versus dimension for 256, 16K, and 1M nodes, logarithmic delay.

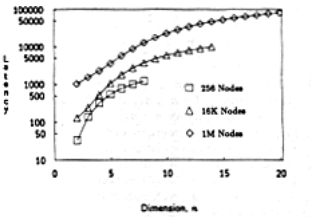


Fig. 10. Latency versus dimension for 256, 16K, and 1M nodes, linear delay.

© Culler 1995

Tradeoffs between LDNs and HDNs

- LDN latency:
  - **primarily determined by # of hops**
- HDN latency:
  - **primarily determined by length of message going across W=1 channel**
- In applications that can exploit locality, LDNs are better because:
  - **Distance between PEs is smaller as no long wires**
  - **Wider channels available between neighbors:**
  - **In hypercubes, although total number of channels out of each node is large, most of them cannot be used for a single transfer.**

© Culler 1995

Throughput

- Number of messages that can be handled per unit time
- Network capacity per node = $\frac{\text{Total BW out of each Node}}{\text{Avg \# Channels per Msg}}$

  $= n \cdot W / ((k-1) / 2) \cdot n) = (n \cdot k/2) / (n \cdot k/2) = 1$
- Thus for constant B(n,k) networks, capacity is independent of "n".

- Throughput is less than capacity because some channels will block.
- Contention also increases latency.
- Throughput can be increased
- and latency reduced by using
- virtual channels.
-
- Hot-Spot throughput: LDNs do better as more pins per node than HDNs

© Culler 1995

k-ary n-cube summary

- Contrary to conventional wisdom, LDNs using wormhole routing offer better latency, throughput, and hot-spot performance.
  - **LDNs are easier to layout:**
  - **more uniform wiring density**
  - **mostly local connections (e.g., Grids)**
- Example: Delay for sending 10-byte message.
  - **N = 1024, W = 8 bits, D = 32 hops, L = 10 bytes, Tc = 10ns**
  - **T-net is only 0.410 microseconds**
- Interesting extensions: virtual channels; express channels; adaptive routing, ...

© Culler 1995