What characterizes 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 msgs may follow
  - many algorithms with different properties
    - deadlock avoidance?
- **Switching Strategy** (how)
  - how data in a msg traverses a route
  - circuit switching vs. packet switching
- **Flow Control Mechanism** (when)
  - when a msg or portions of it traverse a route
  - what happens when traffic is encountered?

Formalism

- network is a graph \( V = \{ \text{switches and nodes} \} \) connected by communication channels \( C \subseteq V \times V \)
- Channel has width \( w \) and signaling rate \( f = \frac{1}{\tau} \)
  - channel bandwidth \( b = wf \)
  - phit (physical unit) data transferred per cycle
  - flit - basic unit of flow-control
- Number of input (output) channels is switch degree
- Sequence of switches and links followed by a message is a route

Topological Properties

- **Routing Distance** - number of links on route
- **Diameter** - maximum routing distance
- **Average Distance**
- A network is partitioned by a set of links if their removal disconnects the graph

Think streets and intersections
Interconnection Topologies

- Class of networks scaling with N
- Logical Properties:
  - distance, degree
- Physical properties
  - length, width

- Fully connected network
  - diameter = 1
  - degree = N
  - cost?
    - bus => $O(N)$, but BW is $O(1)$ - actually worse
    - crossbar => $O(N^2)$ for BW $O(N)$
- VLSI technology determines switch degree

Example: Linear Arrays and Rings

- 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

Example: Multidimensional Meshes and Tori

- $n$-dimensional array
  - $N = k_{i_1} \times \ldots \times k_{i_d}$ nodes
  - described by $n$-vector of coordinates $(i_1, \ldots, i_d)$
- $n$-dimensional $k$-ary mesh: $N = k^n$
  - $k = \sqrt[n]{N}$
  - described by $n$-vector of radix $k$ coordinate
- $n$-dimensional $k$-ary torus (or $k$-ary $n$-cube)?

On Chip: Embeddings in two dimensions

- Embed multiple logical dimension in one physical dimension using long wires
- When embedding higher-dimension in lower one, either some wires longer than others, or all wires long
Trees

- Diameter and ave distance logarithmic
  - k-ary tree, height \( n = \log_k N \)
  - address specified n-vector of radix k coordinates describing path down from root
- Fixed degree
- Route up to common ancestor and down
  - \( R = B \text{xor} A \)
  - 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 \( O(\sqrt{N}) \) long wires
- Bisection BW?

Fat-Trees

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

Butterflies

- Tree with lots of roots!
- \( N \log N \) (actually \( N/2 \times \log N \))
- Exactly one route from any source to any dest
- \( R = A \text{xor} B \), at level \( i \) use ‘straight’ edge if \( r_i=0 \), otherwise cross edge
- Bisection \( N/2 \) vs \( N^{(n-1)/n} \) (for n-cube)

k-ary n-cubes vs k-ary n-flies

- degree \( n \) vs degree \( k \)
- \( N \) switches vs \( N \log N \) switches
- diminishing BW per node vs constant
- requires locality vs little benefit to locality
- Can you route all permutations?
**Benes network and Fat Tree**

- Back-to-back butterfly can route all permutations
- What if you just pick a random mid point?

**Hypercubes**
- Also called binary n-cubes. \( \# \text{ of nodes } = N = 2^n \).
- \( O(\log N) \) Hops
- Good bisection BW
- Complexity
  - Out degree is \( n = \log N \)
  - Correct dimensions in order
  - With random comm. 2 ports per processor

**Some Properties**
- Routing
  - Relative distance: \( R = (b_{n-1} - a_{n-1}, \ldots, b_0 - a_0) \)
  - Traverse \( r_i = b_i - a_i \) hops in each dimension
  - Dimension-order routing? Adaptive routing?
- Average Distance
  - \( n \times 2k/3 \) for mesh
  - \( nk/2 \) for cube
- Degree?
- Bisection bandwidth?
- Physical layout?
  - \( k^{n-1} \) bidirectional links
  - 2D in \( O(N) \) space
  - Higher dimension?

**The Routing problem: Local decisions**
- Routing at each hop: Pick next output port!
How do you build a crossbar?

Input buffered switch

- Independent routing logic per input
  - FSM
- Scheduler logic arbitrates each output
  - priority, FIFO, random
- Head-of-line blocking problem
  - Message at head of queue blocks messages behind it

Output Buffered Switch

- How would you build a shared pool?

Properties of Routing Algorithms

- Routing algorithm:
  - $R: \mathbb{N} \times \mathbb{N} \rightarrow C$, which at each switch maps the destination node $n_d$ to the next channel on the route
  - which of the possible paths are used as routes?
  - how is the next hop determined?
    - arithmetic
    - source-based port select
    - table driven
    - general computation
- 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 packets are deadlocked and never move forward
Example: Simple Routing Mechanism

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

Administrative

- Exam: This Wednesday (3/17)
  Location: 310 Soda
  TIME: 6:00-9:00
  - This info is on the Lecture page (has been)
  - Get on 8 ½ by 11 sheet of notes (both sides)
  - Meet at LaVal’s afterwards for Pizza and Beverages
- Assume that major papers we have discussed may show up on exam

Communication Performance

- Typical Packet includes data + encapsulation bytes
  - Unfragmented packet size $S = S_{data} + S_{encapsulation}$
- Routing Time:
  - $\text{Time}(S)_{adv} = \text{overhead} + \text{routing delay} + \text{channel occupancy} + \text{contention delay}$
  - Channel occupancy = $S/b = (S_{data} + S_{encapsulation})/b$
  - Routing delay in cycles ($\Delta$):
    - Time to get head of packet to next hop
  - Contention?

Store & Forward vs Cut-Through Routing

- Time: $h(S/b + \Delta/\tau)$ vs $S/b + h\Delta/\tau$
- OR(cycles): $h(S/w + \Delta)$ vs $S/w + h\Delta$
  - what if message is fragmented?
  - wormhole vs virtual cut-through
**Contention**

- Two packets trying to use the same link at the 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
  - Source Squelching

**Bandwidth**

- What affects local bandwidth?
  - packet density: \( b \times \frac{S_{data}}{S} \)
  - routing delay: \( b \times \frac{S_{data}}{(S + w)} \)
  - 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: \( C_b \)
  - suppose \( N \) hosts issue packet every \( M \) cycles with ave dist
    - each msg occupies \( h \) channels for \( = \frac{S}{w} \) cycles each
    - \( C/N \) channels available per node
    - link utilization for store-and-forward:
      \[ \rho = \frac{h}{M} \frac{C/N}{channel \ cycles/node} = \frac{Nh}{MC} < 1! \]
    - link utilization for wormhole routing?

**Saturation**

- How Many Dimensions?
  - \( n = 2 \) or \( n = 3 \)
    - Short wires, easy to build
    - Many hops, low bisection bandwidth
    - Requires traffic locality
  - \( n \geq 4 \)
    - Harder to build, more wires, longer average length
    - Fewer hops, better bisection bandwidth
    - Can handle non-local traffic
  - \( k \)-ary \( n \)-cubes provide a consistent framework for comparison
    - \( N = k^n \)
    - scale dimension (\( n \)) or nodes per dimension (\( k \))
    - assume cut-through
Traditional Scaling: Latency scaling with N

- Assumes equal channel width
  - independent of node count or dimension
  - dominated by average distance

Average Distance

- but, equal channel width is not equal cost!
- Higher dimension => more channels

Dally Paper: In the 3D world

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

Dally paper (con’t)

- Equal Bisection, $W=1$ for hypercube $\Rightarrow W = \frac{1}{2}k$
- Three wire models:
  - Constant delay, independent of length
  - Logarithmic delay with length (exponential driver tree)
  - Linear delay (speed of light/optimal repeaters)
Equal cost in k-ary n-cubes

- Equal number of nodes?
- Equal number of pins/wires?
- Equal bisection bandwidth?
- Equal area?
- Equal wire length?

What do we know?
- switch degree: n
diameter = n(k-1)
- total links = Nn
- pins per node = 2wn
- bisection = kn-1 = N/k links in each directions
- 2Nw/k wires cross the middle

Latency for Equal Width Channels

- total links(N) = Nn

Latency with Equal Pin Count

- Baseline n=2, has w = 32  (128 wires per node)
- fix 2nw pins => w(n) = 64/n
- distance up with n, but channel time down
Larger Routing Delay (w/ equal pin)

- Dally’s conclusions strongly influenced by assumption of small routing delay
  - Here, Routing delay $\Delta = 20$

Saturation

- Fatter links shorten queuing delays

Reducing routing delay: Express Cubes

- Problem: Low-dimensional networks have high $k$
  - Consequence: may have to travel many hops in single dimension
  - Routing latency can dominate long-distance traffic patterns
- Solution: Provide one or more “express” links

  ![Express Cubes Diagram](image)

- Like express trains, express elevators, etc
  - Delay linear with distance, lower constant
  - Closer to “speed of light” in medium
  - Lower power, since no router cost
- Another Idea: route with pass transistors through links

Reducing Contention with Virtual Channels

- Problem: A blocked message can prevent others from using physical channels:

  ![Virtual Channels Diagram](image)

- Idea: add channels!
  - provide multiple “virtual channels” to break the dependence cycle
  - good for BW too!

  ![Virtual Channels Construction](image)

- Do not need to add links, or xbar, only buffer resources
Paper Discussion: Bill Dally
“Virtual Channel Flow Control”

• Basic Idea: Use of virtual channels to reduce contention
  – Provided a model of k-ary, n-flies
  – Also provided simulation
• Tradeoff: Better to split buffers into virtual channels
  – Example (constant total storage for 2-ary 8-fly):

[Graph showing latency versus throughput for different scenarios]

When are virtual channels allocated?

• Two separate processes:
  – Virtual channel allocation
  – Switch/connection allocation
• Virtual Channel Allocation
  – Choose route and free output virtual channel
  – Really means: Source of link tracks channels at destination
• Switch Allocation
  – For incoming virtual channel, negotiate switch on outgoing pin

Deadlock Freedom

• How can deadlock arise?
  – necessary conditions:
    » shared resource
    » incrementally allocated
    » non-preemptible
  – channel is a shared resource that is acquired incrementally
    » source buffer then dest. buffer
    » channels along a route
• How do you avoid it?
  – constrain how channel resources are allocated
  – ex: dimension order
• Important assumption:
  – Destination of messages must always remove messages
• How do you prove that a routing algorithm is deadlock free?
  – Show that channel dependency graph has no cycles!

Consider Trees

• Why is the obvious routing on X deadlock free?
  – butterfly?
  – tree?
  – fat tree?
• Any assumptions about routing mechanism?
  amount of buffering?
Up*-Down* routing for general topology

- Given any bidirectional network
- Construct a spanning tree
- Number of the nodes increasing from leaves to roots
- UP increase node numbers
- Any Source -> Dest by UP*-DOWN* route
  - up edges, single turn, down edges
  - Proof of deadlock freedom?
- Performance?
  - Some numberings and routes much better than others
  - interacts with topology in strange ways

Turn Restrictions in X,Y

- XY routing forbids 4 of 8 turns and leaves no room for adaptive routing
- Can you allow more turns and still be deadlock free?

Minimal turn restrictions in 2D

- West-first
- north-last negative first

Example legal west-first routes

- Can route around failures or congestion
- Can combine turn restrictions with virtual channels
**General Proof Technique**

- resources are logically associated with channels
- messages introduce dependences between resources as they move forward
- need to articulate the possible dependences that can arise between channels
- show that there are no cycles in Channel Dependence Graph
  - find a numbering of channel resources such that every legal route follows a monotonic sequence
  - no traffic pattern can lead to deadlock
- network need not be acyclic, just channel dependence graph

**Example: k-ary 2D array**

- Thm: Dimension-ordered (x,y) routing is deadlock free
- Numbering
  - +x channel (i,y) \(\rightarrow (i+1,y)\) gets \(i\)
  - similarly for -x with 0 as most positive edge
  - +y channel (x,j) \(\rightarrow (x,j+1)\) gets \(N+j\)
  - similarly for -y channels
- any routing sequence: x direction, turn, y direction is increasing
- Generalization:
  - “e-cube routing” on 3-D: X then Y then Z

**Channel Dependence Graph**

**More examples:**

- What about wormhole routing on a ring?

- Or: *Unidirectional* Torus of higher dimension?
Breaking deadlock with virtual channels

• Basic idea: Use virtual channels to break cycles
  – Whenever wrap around, switch to different set of channels
  – Can produce numbering that avoids deadlock

General Adaptive Routing

• \( R: C \times N \times \Sigma \rightarrow C \)
• Essential for fault tolerance
  – At least multipath
• Can improve utilization of the network
• Simple deterministic algorithms easily run into bad permutations
• Fully/partially adaptive, minimal/non-minimal
• Can introduce complexity or anomalies
• Little adaptation goes a long way!

Paper Discussion: Linder and Harden
“An Adaptive and Fault Tolerant Wormhole”

• General virtual-channel scheme for k-ary n-cubes
  – With wrap-around paths
• Properties of result for uni-directional k-ary n-cube:
  – 1 virtual interconnection network
  – n+1 levels
• Properties of result for bi-directional k-ary n-cube:
  – \(2^n-1\) virtual interconnection networks
  – n+1 levels per network

Example: Unidirectional 4-ary 2-cube

Physical Network
• Wrap-around channels necessary but can cause deadlock

Virtual Network
• Use VCs to avoid deadlock
• 1 level for each wrap-around
Bi-directional 4-ary 2-cube: 2 virtual networks

Use of virtual channels for adaptation

- Want to route around hotspots/faults while avoiding deadlock
- Linder and Harden, 1991
  - General technique for k-ary n-cubes
    - Requires: $2^{n-k}$ virtual channels/lane!!
- Alternative: Planar adaptive routing
  - Chien and Kim, 1995
  - Divide dimensions into “planes”,
    - i.e. in 3-cube, use X-Y and Y-Z
  - Route planes adaptively in order: first X-Y, then Y-Z
    - Never go back to plane once have left it
    - Can’t leave plane until have routed lowest coordinate
  - Use Linder-Harden technique for series of 2-dim planes
    - Now, need only 3 $\times$ number of planes virtual channels
- Alternative: two phase routing
  - Provide set of virtual channels that can be used arbitrarily for routing
  - When blocked, use unrelated virtual channels for dimension-order (deterministic) routing
  - Never progress from deterministic routing back to adaptive routing

Summary #1

- Network Topologies:
  - Fair metrics of comparison
    - Equal cost: area, bisection bandwidth, etc

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

Summary #2

- Routing Algorithms restrict the set of routes within the topology
  - simple mechanism selects turn at each hop
  - arithmetic, selection, lookup
- Virtual Channels
  - Adds complexity to router
  - Can be used for performance
  - Can be used for deadlock avoidance
- Deadlock-free if channel dependence graph is acyclic
  - limit turns to eliminate dependences
  - add separate channel resources to break dependences
  - combination of topology, algorithm, and switch design
- Deterministic vs adaptive routing