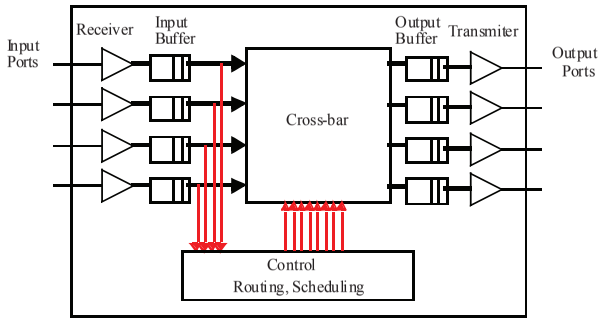
CS252
Graduate Computer Architecture
Lecture 16

Multiprocessor Networks (con't)
March 14th, 2012

John Kubiatowicz
Electrical Engineering and Computer Sciences
University of California, Berkeley

http://www.eecs.berkeley.edu/~kubitron/cs252

Recall: The Routing problem: Local decisions

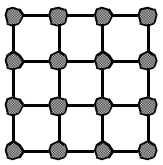


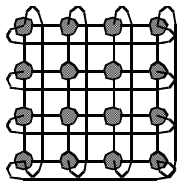
- Routing at each hop: Pick next output port!

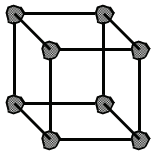
Properties of Routing Algorithms

- Routing algorithm:
  - R: N x N -> 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

Recall: Multidimensional Meshes and Tori


2D Grid

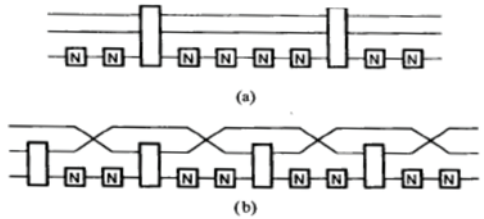

2D Torus


3D Cube

- *n*-dimensional array
  - $N = k_{n-1} X ... X k_0$ nodes
  - described by *n*-vector of coordinates $(i_{n-1}, ..., i_0)$
- *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)?

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

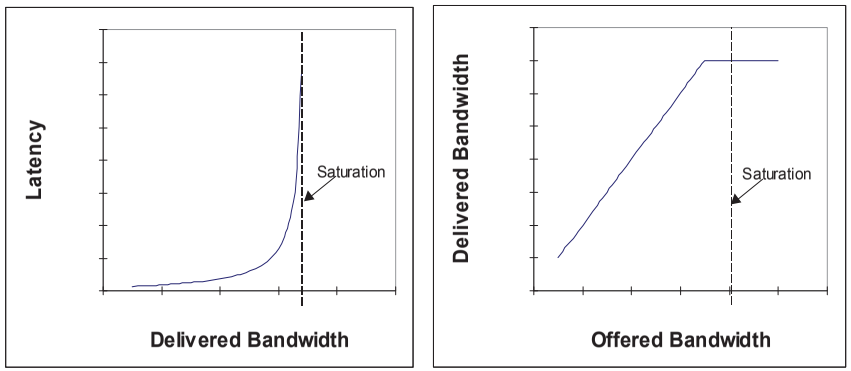


  - Like express trains, express elevators, etc
    - » Delay linear with distance, lower constant
    - » Closer to "speed of light" in medium
    - » Lower power, since no router cost
  - "Express Cubes: Improving performance of k-ary n-cube interconnection networks," Bill Dally 1991
- Another Idea: route with pass transistors through links

Bandwidth

- What affects local bandwidth?
  - packet density: $b \times S_{data}/S$
  - routing delay: $b \times S_{data}/(S + w\Delta)$
  - 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 = S/w cycles each
    - » C/N channels available per node
    - » link utilization for store-and-forward: $\rho = (hl/M \text{ channel cycles/node})/(C/N) = Nhl/MC < 1!$
    - » link utilization for wormhole routing?

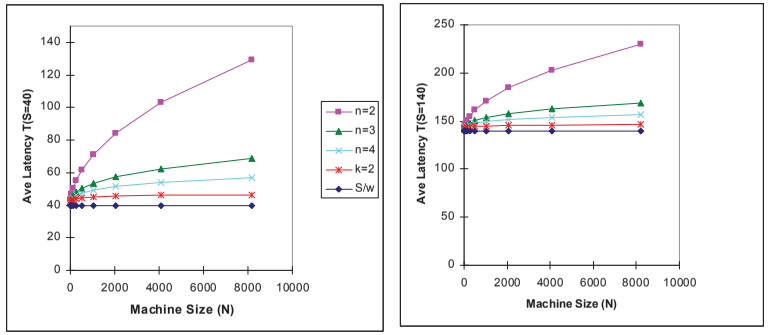
Saturation



How Many Dimensions?

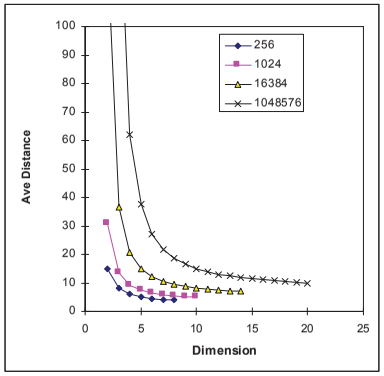
- n = 2 or n = 3
  - Short wires, easy to build
  - Many hops, low bisection bandwidth
  - Requires traffic locality
- n >= 4
  - Harder to build, more wires, longer average length
  - Fewer hops, better bisection bandwidth
  - Can handle non-local traffic
- k-ary n-cubes provide a consistent framework 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

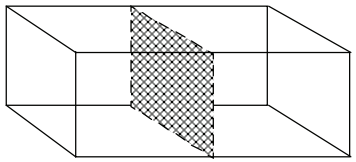


ave dist = n(k-1)/2

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

Dally Paper: In the 3D world

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



- 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?

Dally Paper (con't)

- Equal Bisection,W=1 for hypercube ⇒ W= ½k
- Three wire models:
  - Constant delay, independent of length
  - Logarithmic delay with length (exponential driver tree)
  - Linear delay (speed of light/optimal repeaters)

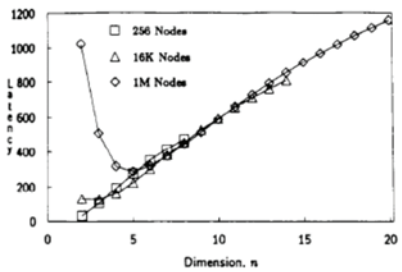


Fig. 9. Latency versus dimension for 256, 16K, and 1M nodes, logarithmic delay.

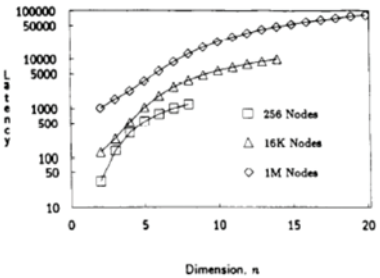


Fig. 10. Latency versus dimension for 256, 16K, and 1M nodes, linear delay.

Logarithmic Delay

Linear Delay

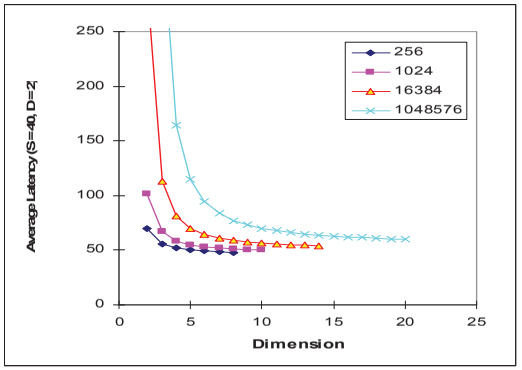
Equal cost in k-ary n-cubes

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

What do we know?

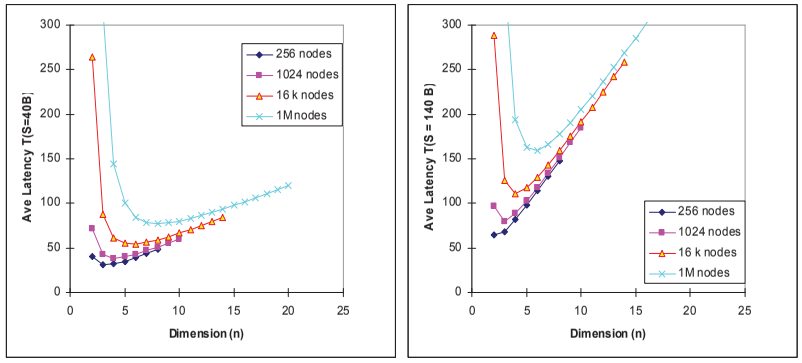
- switch degree: n diameter = n(k-1)
- total links = Nn
- pins per node = 2wn
- bisection = k^{n-1} = N/k links in each directions
- 2Nw/k wires cross the middle

Latency for Equal Width Channels



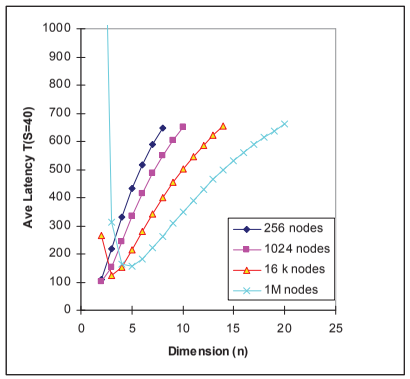
- total links(N) = Nn

Latency with Equal Pin Count



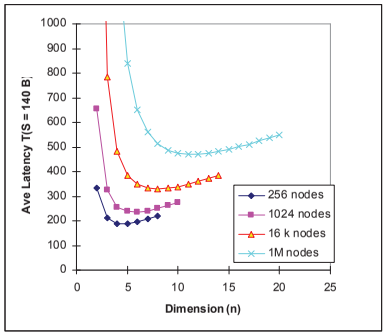
- Baseline n=2, has w = 32 (128 wires per node)
- fix 2nw pins => w(n) = 64/n
- distance up with n, but channel time down

Latency with Equal Bisection Width



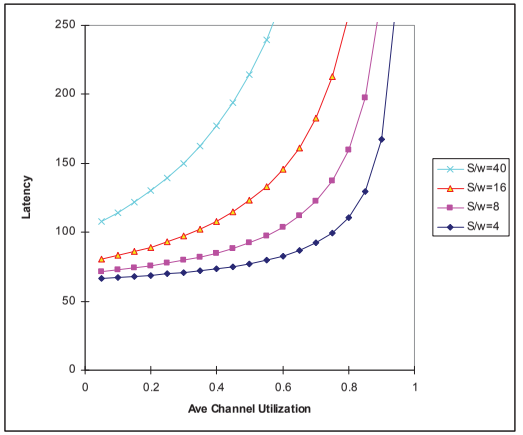
- N-node hypercube has N bisection links
- 2d torus has $2N^{1/2}$
- Fixed bisection ⇒ $w(n) = N^{1/n} / 2 = k/2$
- 1 M nodes, n=2 has w=512!

Larger Routing Delay (w/ equal pin)



- Dally's conclusions strongly influenced by assumption of small routing delay
  - Here, Routing delay Δ=20

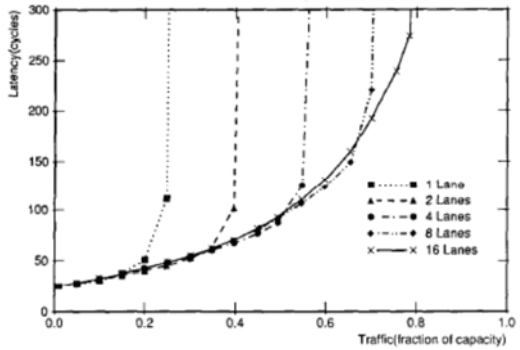
Saturation



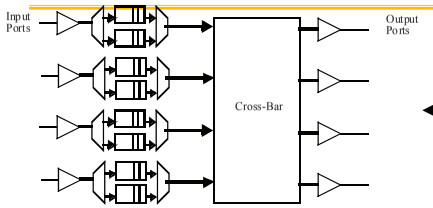
- Fatter links shorten queuing delays

Discuss of paper: 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):



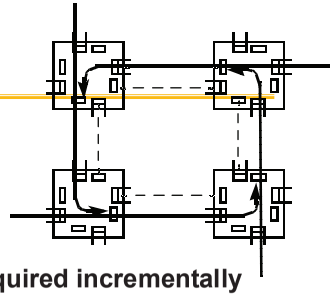
When are virtual channels allocated?



Hardware efficient design For crossbar

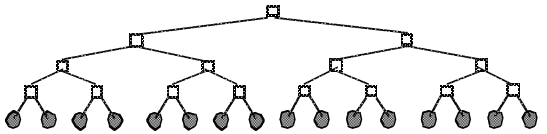
- Two *separate processes:*
  - Virtual channel allocation
  - Switch/connection allocation
- Virtual Channel Allocation
  - Choose route and free output virtual channel
  - Really means: Source of link tracks channels at destination
- Switch Allocation
  - For incoming virtual channel, negotiate switch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

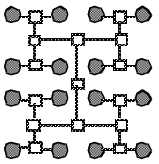
Deadlock Freedom



- 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



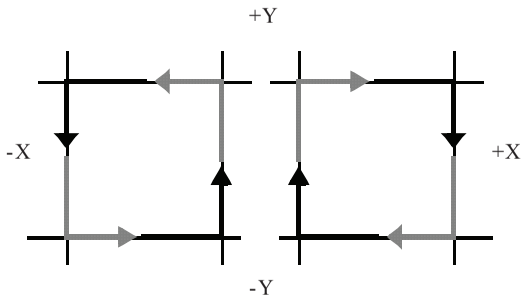


- Why is the obvious routing on X deadlock free?
  - butterfly?
  - tree?
  - fat tree?
- Any assumptions about routing mechanism? amount of buffering?

Up*-Down* routing for general topology

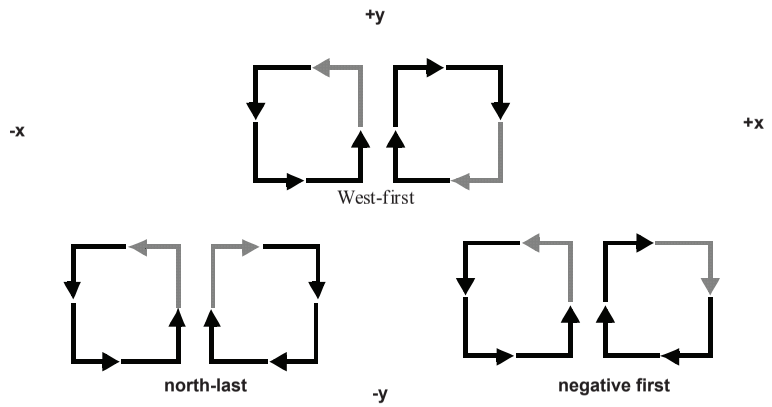
- Given any bidirectional network
- Construct a spanning tree
- Number of the nodes increasing from leaves to roots
- UP increase node numbers
- Any Source -> Dest by UP*-DOWN* route
  - up edges, single turn, down edges
  - Proof of deadlock freedom?
- Performance?
  - Some numberings and routes much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

Turn Restrictions in X,Y

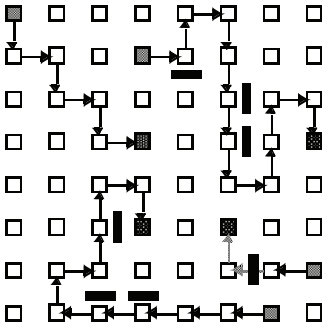


- XY routing forbids 4 of 8 turns and leaves no room for adaptive routing
- Can you allow more turns and still be deadlock free?

Minimal turn restrictions in 2D



Example legal west-first routes



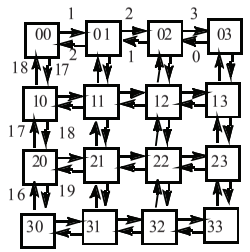
- Can route around failures or congestion
- Can combine turn restrictions with virtual channels

General Proof Technique

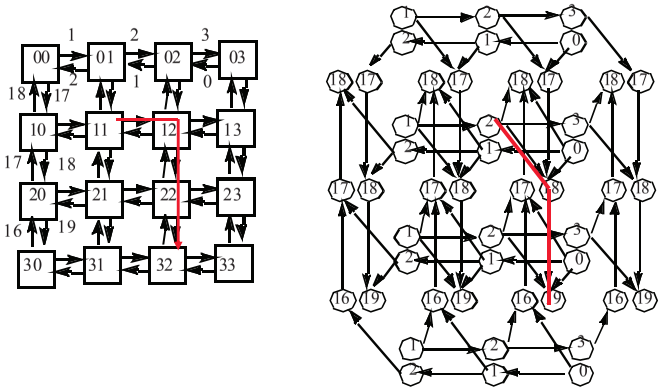
- resources are logically associated with channels
- messages introduce dependences between resources as they move forward
- need to articulate the possible dependences that can arise between channels
- show that there are no cycles in Channel Dependence Graph
  - find a numbering of channel resources 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) -> (i+1,y) gets i
  - similarly for -x with 0 as most positive edge
  - +y channel (x,j) -> (x,j+1) gets N+j
  - similary 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

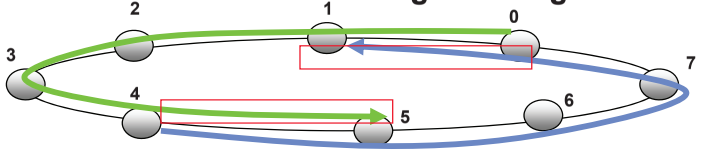


Channel Dependence Graph

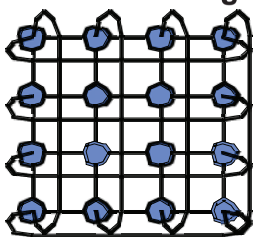


More examples:

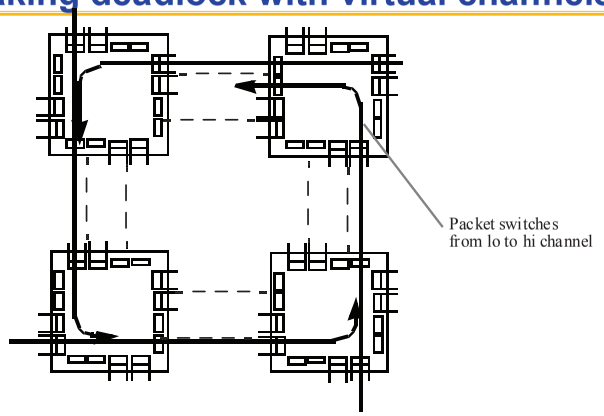
- What about wormhole routing on a ring?



- Or: *Unidirectional* Torus of higher dimension?



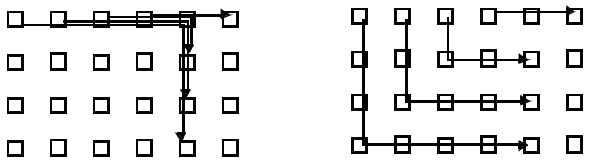
Breaking deadlock with virtual channels



- 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 x N x Σ -> C
- Essential for fault tolerance
  - at least multipath
- Can improve utilization of the network
- Simple deterministic algorithms easily run into bad permutations

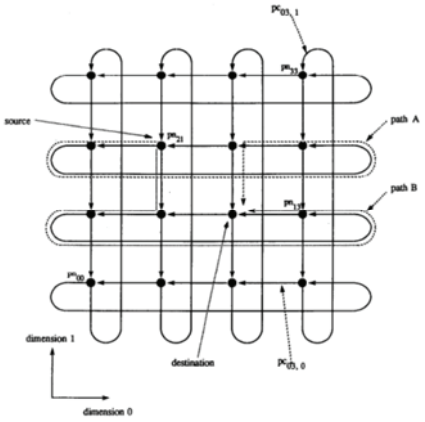


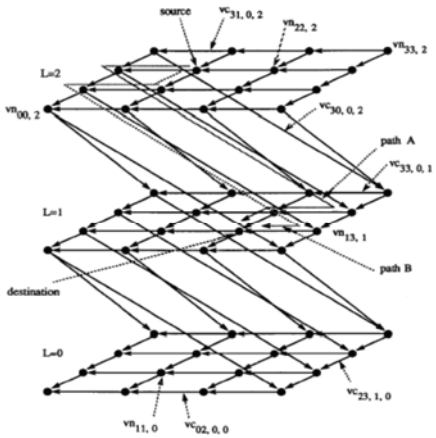
- fully/partially adaptive, minimal/non-minimal
- can introduce complexity or anomalies
- little adaptation goes a long way!

Paper Discusion: 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

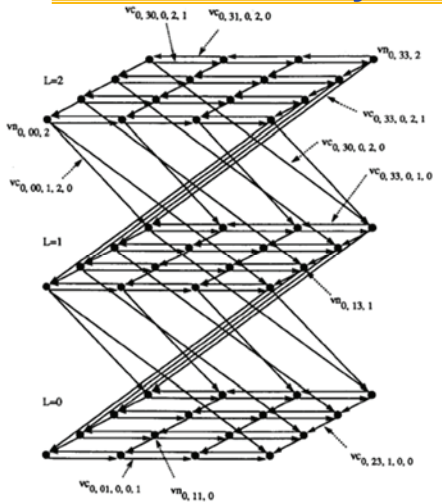




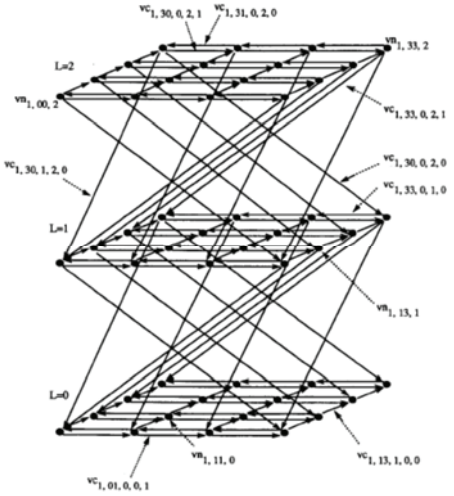
Physical Network
- Wrap-around channels necessary but can cause deadlock

Virtual Network
- Use VCs to avoid deadlock
- 1 level for each wrap-around

Bi-directional 4-ary 2-cube: 2 virtual networks



Virtual Network 1



Virtual Network 2

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-1} 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 ✥ 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

- Fair metrics of comparison
  - Equal cost: area, bisection bandwidth, etc
- Routing Algorithms restrict 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