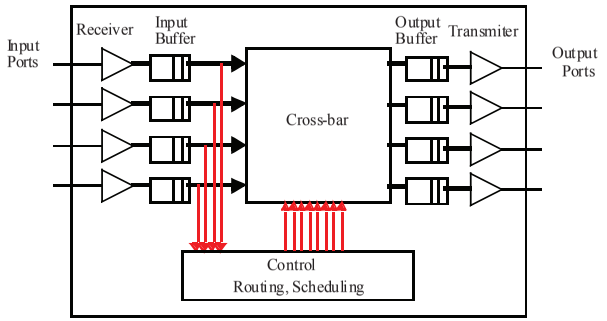
CS252
Graduate Computer Architecture
Lecture 16

Multiprocessor Networks (con't)
March 16th, 2011

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

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

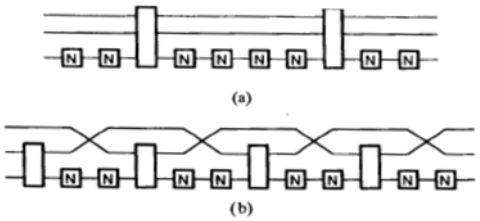
Review: The Routing problem: Local decisions



- Routing at each hop: Pick next output port!

Reducing routing delay: Express Cubes

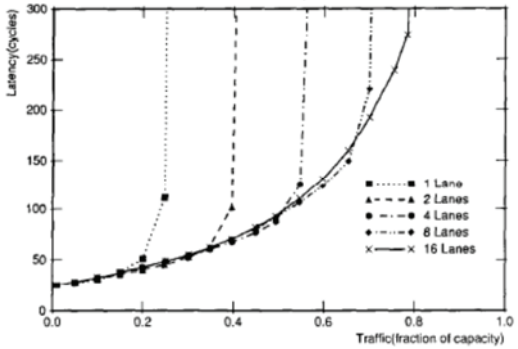
- Problem: Low-dimensional networks have high k
  - Consequence: may have to travel many hops in single dimension
  - Routing latency can dominate long-distance traffic patterns
- Solution: Provide one or more "express" links



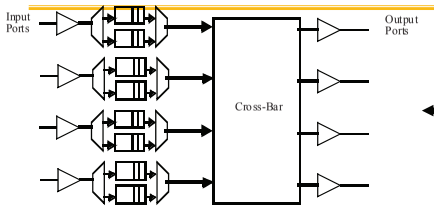
  - Like express trains, express elevators, etc
    - » Delay linear with distance, lower constant
    - » Closer to "speed of light" in medium
    - » Lower power, since no router cost
  - "Express Cubes: Improving performance of k-ary n-cube interconnection networks," Bill Dally 1991
- Another Idea: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

Review: 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):



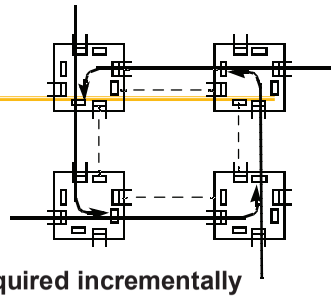
When are virtual channels allocated?



Hardware efficient design For crossbar

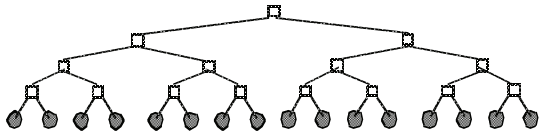
- Two *separate processes:*
  - Virtual channel allocation
  - Switch/connection allocation
- Virtual Channel Allocation
  - Choose route and free output virtual channel
  - Really means: Source of link tracks channels at destination
- Switch Allocation
  - For incoming virtual channel, negotiate switch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

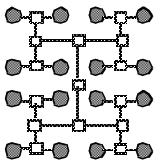
Deadlock Freedom



- 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



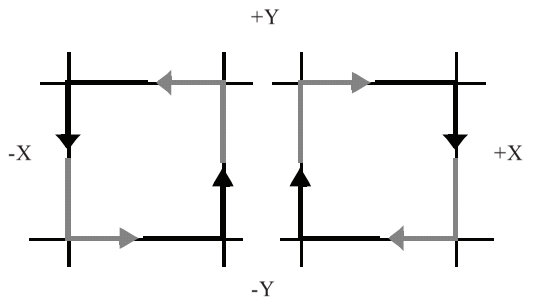


- Why is the obvious routing on X deadlock free?
  - butterfly?
  - tree?
  - fat tree?
- Any assumptions about routing mechanism? amount of buffering?

Up*-Down* routing for general topology

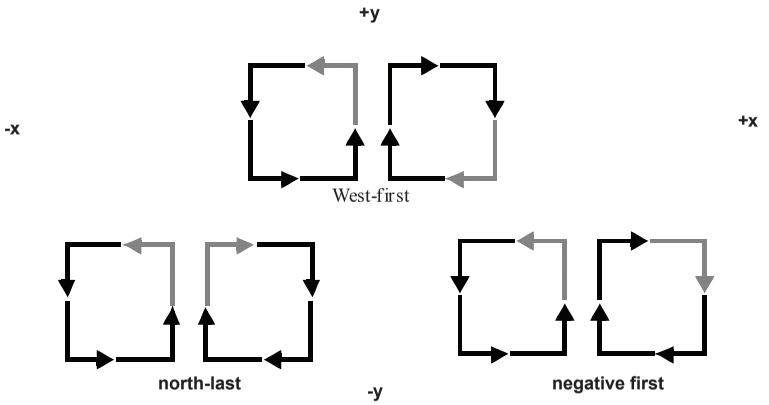
- Given any bidirectional network
- Construct a spanning tree
- Number of the nodes increasing from leaves to roots
- UP increase node numbers
- Any Source -> Dest by UP*-DOWN* route
  - up edges, single turn, down edges
  - Proof of deadlock freedom?
- Performance?
  - Some numberings and routes much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

Turn Restrictions in X,Y

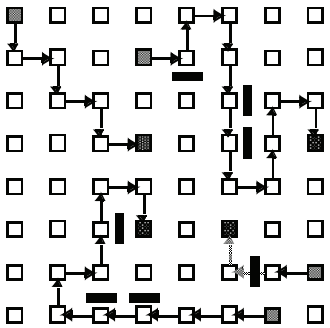


- XY routing forbids 4 of 8 turns and leaves no room for adaptive routing
- Can you allow more turns and still be deadlock free?

Minimal turn restrictions in 2D



Example legal west-first routes



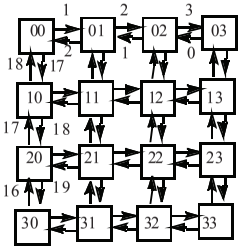
- Can route around failures or congestion
- Can combine turn restrictions with virtual channels

General Proof Technique

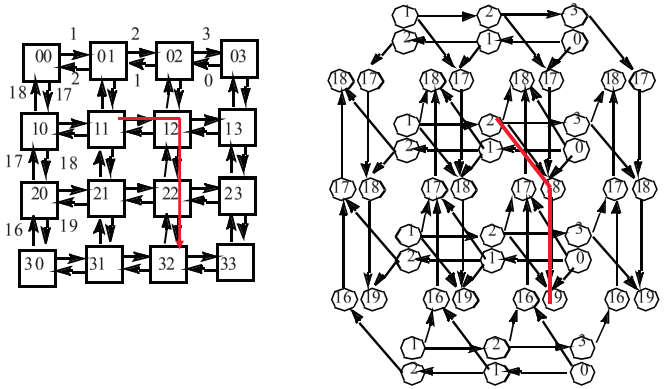
- resources are logically associated with channels
- messages introduce dependences between resources as they move forward
- need to articulate the possible dependences that can arise between channels
- show that there are no cycles in Channel Dependence Graph
  - find a numbering of channel resources 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

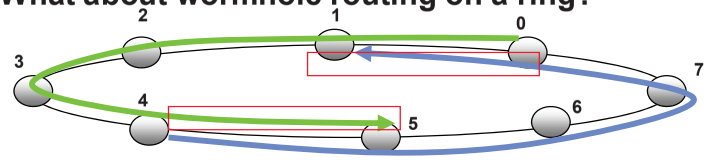


Channel Dependence Graph

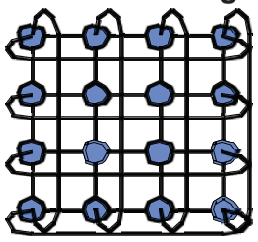


More examples:

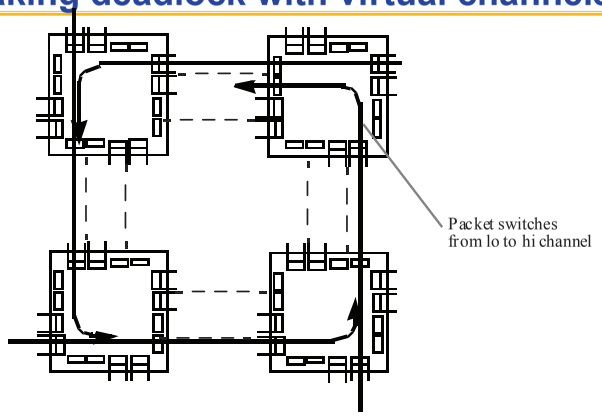
- What about wormhole routing on a ring?



- Or: *Unidirectional* Torus of higher dimension?



Breaking deadlock with virtual channels



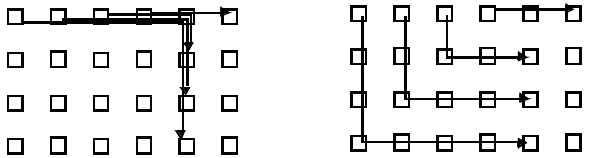
- Basic idea: Use virtual channels to break cycles
  - Whenever wrap around, switch to different set of channels
  - Can produce numbering that avoids deadlock

Administrative

- Should be working full blast on project by now!
  - I'm going to want you to submit an update after spring break
  - We will meet shortly after that
- Midterm I: 3/30
  - Two-hour exam over three hours.
    - » Tentatively 2:30-5:30
  - Material: Everything up to today is fair game (including papers)
  - One page of notes – both sides
  - Plenty of sample exams up on the handouts page

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

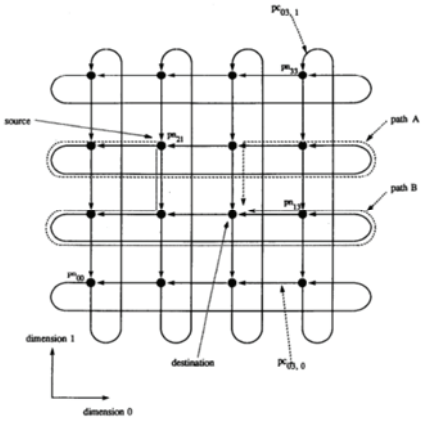


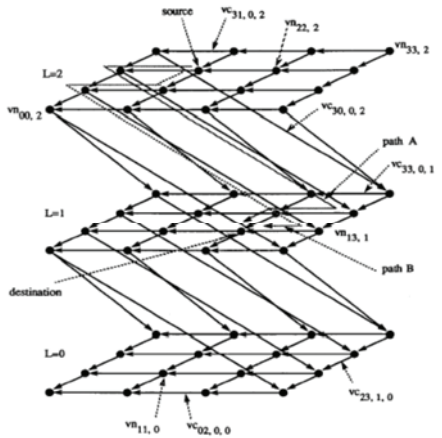
- fully/partially adaptive, minimal/non-minimal
- can introduce complexity or anomalies
- little adaptation goes a long way!

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





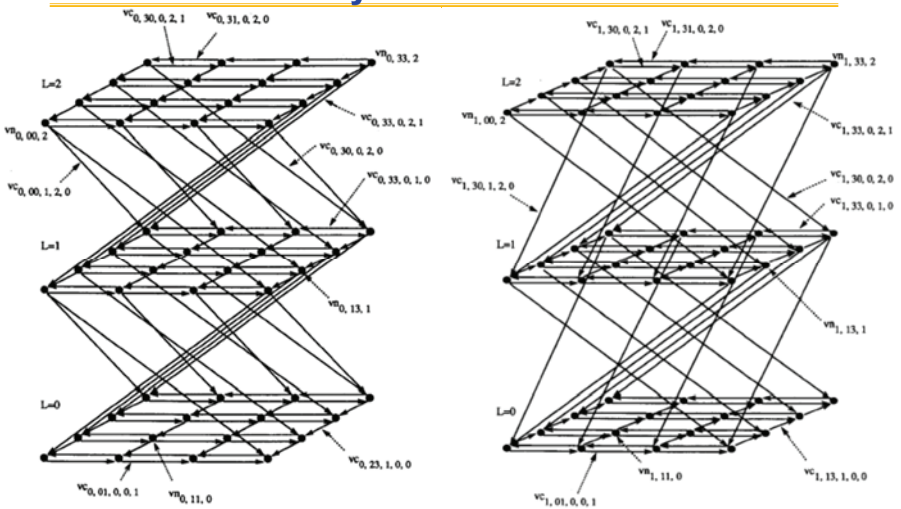
Physical Network

- 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

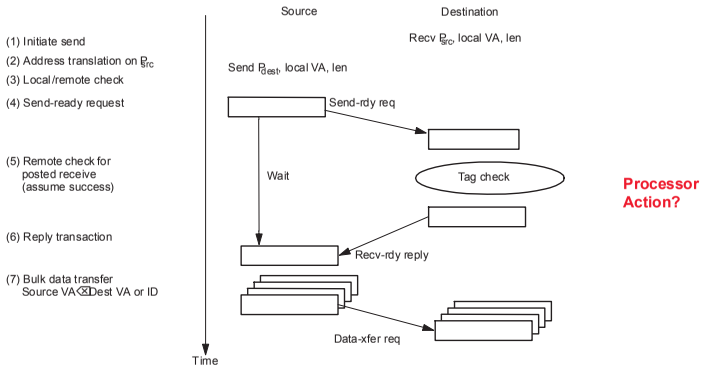
Use of virtual channels for adaptation

- 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

Message passing

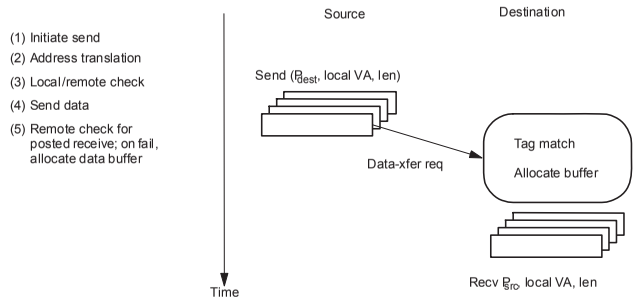
- Sending of messages under control of programmer
  - User-level/system level?
  - Bulk transfers?
- How efficient is it to send and receive messages?
  - Speed of memory bus? First-level cache?
- Communication Model:
  - Synchronous
    - » Send completes after matching recv and source data sent
    - » Receive completes after data transfer complete from matching send
  - Asynchronous
    - » Send completes after send buffer may be reused

Synchronous Message Passing



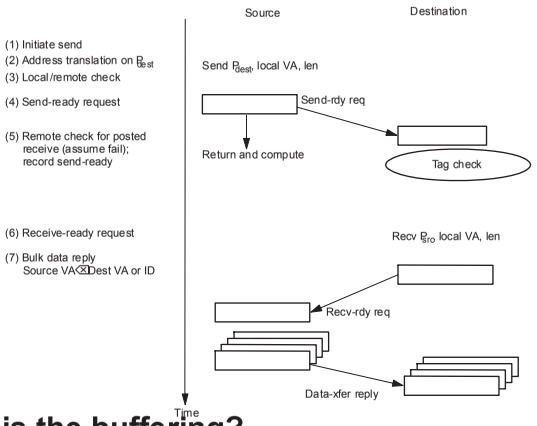
- Constrained programming model.
- Deterministic! What happens when threads added?
- Destination contention very limited.
- User/System boundary?

Asynch. Message Passing: Optimistic



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

Asynch. Msg Passing: Conservative



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

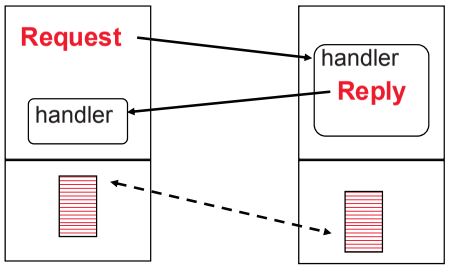
Features of Msg Passing Abstraction

- Source knows send data address, dest. knows receive data address
  - after handshake they both know both
- 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
- Optimistically, can be 1-phase transaction
  - Compare to 2-phase for shared address space
  - Need some sort of flow control
    - » Credit scheme?
- More conservative: 3-phase transaction
  - includes a request / response
- Essential point: combined synchronization and communication in a single package!

Discussion of Active Messages paper

- Thorsten von Eicken, David E. Culler, Seth Copen Goldstein, Laus Erik Schauser:
  - "Active messages: a mechanism for integrated communication and computation"
- Essential idea?
  - Fast message primitive
  - Handlers must be *non-blocking!*
  - Head of message contains pointer to code to run on destination node
  - Handler's sole job is to integrate values into running computation
- Could be compiled from Split-C into TAM runtime
- Much better balance of network and computation on existing hardware!

Active Messages



- User-level analog of network transaction
  - transfer data packet and invoke handler to extract it from the network and integrate with on-going computation
- Request/Reply
- Event notification: interrupts, polling, events?
- May also perform memory-to-memory transfer

Summary

- 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