Networks: Routing and Design

CS 258, Spring 99
David E. Culler
Computer Science Division
U.C. Berkeley

Outline

- Routing
- Switch Design
- Flow Control
- Case Studies

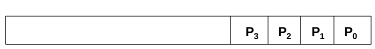
Routing

- Recall: routing algorithm determines
  - which of the possible paths are used as routes
  - how the route is determined
  - R: N x N -> C, which at each switch maps the destination node n_d to the next channel on the route
- Issues:
  - Routing mechanism
    - » arithmetic
    - » source-based port select
    - » table driven
    - » general computation
  - Properties of the routes
  - Deadlock feee

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)



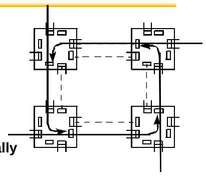
- Source-based
  - message header carries series of port selects
  - used and stripped en route
  - *CRC? Packet Format?*
  - CS-2, Myrinet, MIT Artic
- Table-driven
  - message header carried index for next port at next switch
    - » o = R[i]
  - table also gives index for following hop
    - » o, 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

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 dest. buffer
    - » channels along a route
- How do you avoid it?
  - constrain how channel resources are allocated
  - ex: dimension order
- How do you prove that a routing algorithm is deadlock free

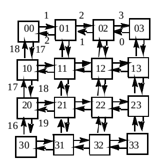


Proof Technique

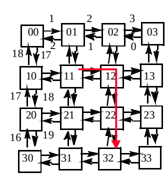
- resources are logically associated with channels
- messages introduce dependences between resources as they move forward
- need to articulate the possible dependences that can arise between channels
- show that there are no cycles in Channel Dependence Graph
  - find a numbering of channel resources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

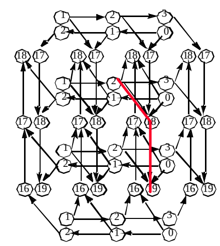
Example: k-ary 2D array

- Thm: 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



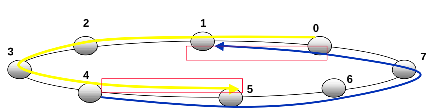
Channel Dependence Graph





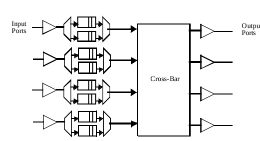
More examples:

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



Deadlock free wormhole networks?

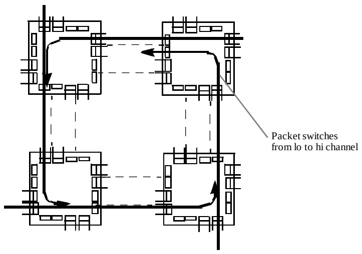
- Basic dimension order routing techniques don't work for k-ary d-cubes
  - only for k-ary d-arrays (bi-directional)
- Idea: add channels!
  - provide multiple "virtual channels" to break the dependence cycle
  - good for BW too!



  - Do not need to add links, or xbar, only buffer resources
- This adds nodes the the CDG, remove edges?

NOW Handout Page 2

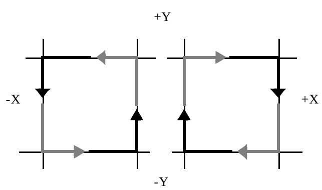
Breaking deadlock with virtual channels



Up*-Down* routing

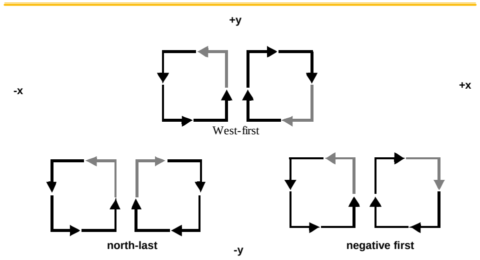
- Given any bidirectional network
- Construct a spanning tree
- Number of the nodes increasing from leaves to roots
- UP increase node numbers
- Any Source -> Dest by UP*-DOWN* route
  - up edges, single turn, down edges
- Performance?
  - Some numberings and routes much better than others
  - interacts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

Turn Restrictions in X,Y

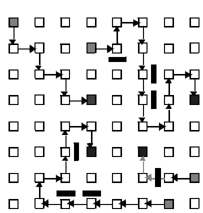


- XY routing forbids 4 of 8 turns and leaves no room for adaptive routing
- Can you allow more turns and still be deadlock free

Minimal turn restrictions in 2D



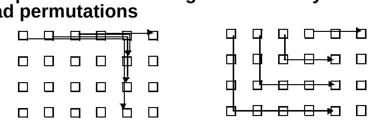
Example legal west-first routes



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

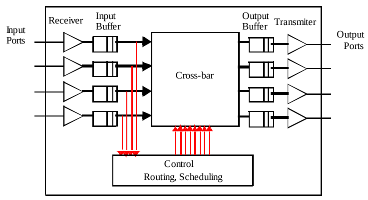
Adaptive Routing

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

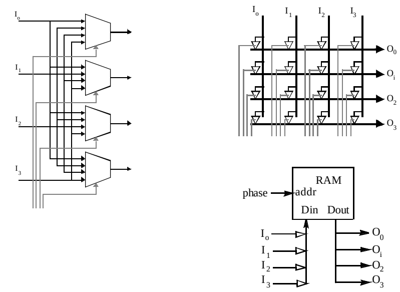


- fully/partially adaptive, minimal/non-minimal
- can introduce complexity or anomolies
- little adaptation goes a long way!

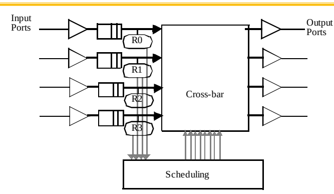
Switch Design



How do you build a crossbar

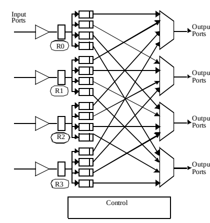


Input buffered swtich



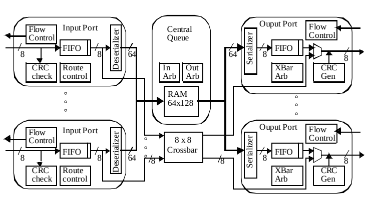
- Independent routing logic per input
  - FSM
- Scheduler logic arbitrates each output
  - priority, FIFO, random
- Head-of-line blocking problem

Output Buffered Switch



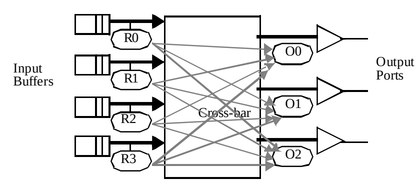
- How would you build a shared pool?

Example: IBM SP vulcan switch



- Many gigabit ethernet switches use similar design without the cut-through

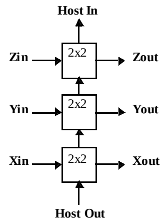
Output scheduling



- n independent arbitration problems?
  - static priority, random, round-robin
- simplifications due to routing algorithm?
- general case is max bipartite matching

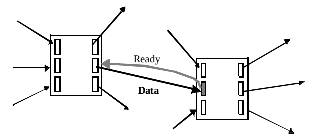
Stacked Dimension Switches

- Dimension order on 3D cube?
- Cube connected cycles?



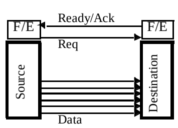
Flow Control

- 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

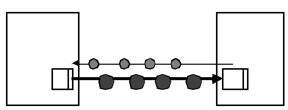


Examples

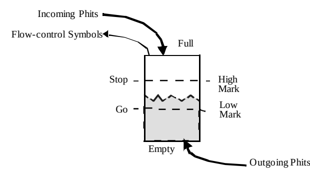
- Short Links



- long links
  - several flits on the wire



Smoothing the flow

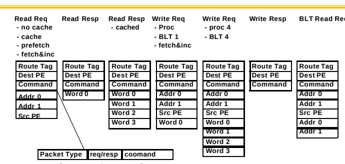


- How much slack do you need to maximize bandwidth?

Link vs global flow control

- Hot Spots
- Global communication operations
- Natural parallel program dependences

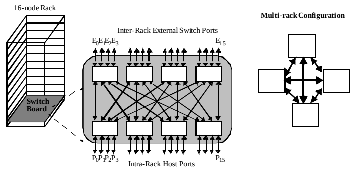
Example: T3D



- 3D bidirectional torus, dimension order (NIC selected), virtual cut-through, packet sw.
- 16 bit x 150 MHz, short, wide, synch.
- rotating priority per output
- logically separate request/response
- 3 independent, stacked switches
- 8 16-bit flits on each of 4 VC in each directions

NOW Handout Page 5

Example: SP



- 8-port switch, 40 MB/s per link, 8-bit phit, 16-bit flit, single 40 MHz clock
- packet sw, cut-through, no virtual channel, source-based routing
- variable packet <= 255 bytes, 31 byte fifo per input, 7 bytes per output, 16 phit links
- 128 8-byte ‘chunks’ in central queue, LRU per output
- run in shadow mode

Summary

- **Routing Algorithms restrict the set of routes within the topology**
  - simple mechanism selects turn at each hop
  - arithmetic, selection, lookup
- **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**
- **Switch design issues**
  - input/output/pooled buffering, routing logic, selection logic
- **Flow control**
- **Real networks are a ‘package’ of design choices**

NOW Handout Page 6