Routing Algorithms

Today's topics:

Deterministic, Oblivious Adaptive, & Adaptive models

Problems:

- efficiency
- livelock
- deadlock

Review

- Network properties are a combination
  - topology
  - topology dependent routing algorithm
  - switch micro-architecture
    - » plus a bunch of things that are "sub-influences"
      - virtual channels
      - packet size
      - error recovery protocol
      - internal switch data- and control-path
- Huge variation of approaches in the research literature
  - goal = cover the breadth
    - » depth is Pandora's box
      - specialist expertise requires years not a semester
- Terminology
  - phit – physical unit – a per clock transfer
  - flit – flow control unit
  - packet – logical unit of transfer

Addressing Modes

- Routing model is dependent upon address spec's
  - source-routed
    - » at each hop – packet field determines exit port
      - not dissimilar from ethernet table based routing
      - dynamic congestion independent possible
    - » routing algorithm is simple – do what the source says
  - absolute
    - » topology dependent definition of the destination
      - topological basis for the address – e.g. NEWS
      - which way to go?
        - – topology dependent
          - – simple or complex calculation
            - – twisted torus – complex
            - – 2D mesh – simple
  - relative
    - » topology dependent
      - relative path based on where I am now
        - – GPS like (calculate here – destination) difference
        - – simple example is 2D mesh – NEWS offset

Routing Models

- Note
  - terminology varies over the years
    - » this one is Dally-speak
      - from the excellent text by Dally & Towles
- Deterministic
  - fixed route between source-destination pairs
    - » problem
      - no dynamic congestion avoidance
- Oblivious
  - dynamic path choice
  - BUT – independent of load
    - » e.g. static load balancing at source
- Adaptive
  - load based routing
    - » TRICK: can local observation of load = global optimum?

Routing Model Issues

- **Deterministic**
  - what happens if something fails
    - » need to determine failure point
      - how? – timeout
        - – how long should you wait?
    - » update routing tables
      - depending on topology – request/reply traffic may conflict
- **Oblivious & Adaptive**
  - same request/reply conflict
  - alternate paths provide opportunity
    - » topology dependent however
      - consider
        - – quad mesh
        - – fat-tree/folded Clos
        - – n-dimensional networks

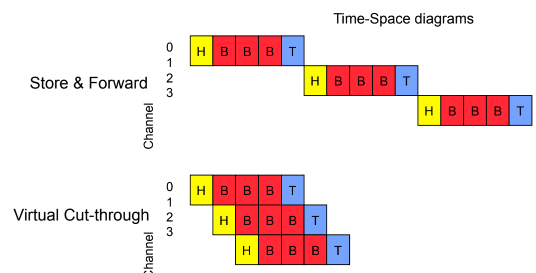
Adaptive vs. Non-Adaptive

- **Deterministic routing**
  - guarantees in order packet delivery delivery
- **Oblivious and Adaptive routing**
  - packets may arrive out of order
- **Out of order issues**
  - reassembly required at end point
  - packet header overhead
    - » packet #, total packets this message fields required
    - » packet overhead is an issue
      - look at ethernet
        - – min packet size = 64 bytes
          - – 48 byte header for IP
            - – 48/64 = 75% overhead
        - – max packet size = 1518 bytes
          - – 48/1518 = 3% overhead
    - » BUT packet latency is an issue
      - route around congestion = out of order
        - – tradeoffs?

Routing Models II

- **At each hop = flow control dependent**
  - Store and forward
    - » flow-control: packet based
    - » receive entire packet, check correct, route
    - » latency non-optimal but minimize occupancy
  - Wormhole
    - » flit != packet
      - digest header & route
    - » packet may now occupy multiple switches
      - head of line blocking problem now has greater "extent"
  - Virtual cut-through
    - » packet based flow control
    - » route decision doesn't need to wait for entire packet to arrive

Time/Space Viewpoint



Routing Properties

- **Key issues**
  - deliver the packet to the prescribed destination
    - » functional correctness issue
  - deadlock avoidance
    - » break
      - incremental claim & circular dependence
      - e.g. 5 philosophers problem
  - livelock avoidance
    - » avoid lots of action – no progress situation
      - harder to detect w/ local view
      - hence packet must carry history
        - – e.g. ethernet "time to live" + end to end protocol capability
        - – OR route without livelock possibility
  - avoid hop specific "head of line blocking"
    - » previous packet goes to destination X
    - » next packet goes to destination Y
      - but can't get through the current hop since X packet holds the buffer

Head of Line Blocking

- **Enter virtual channels**
  - Jose Duato (UPC) book – definitive source
  - basic idea
    - » create packet dependent flow – call it a VC
    - » each VC
      - separately buffered and routed
      - one flow blocked to different destination
        - – let other flow proceed
        - – still in order delivery unless adaptively routed
- **VC's serve multiple purposes**
  - head of line blocking
  - priority – may be age based
    - » express channel for control packets
  - deadlock avoidance
    - » special "last VC" → deterministic

Classic Stages

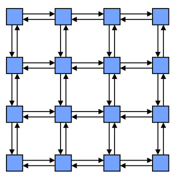
- **Route**
  - determine where packet is destined
- **VC allocation**
  - decide which VC packet is assigned to
    - » how?
      - bump VC at ever hop
        - – buffering overhead
      - bump when HOL blocking indicated
        - – better
- **Switch allocation**
  - arbitrate for route through the datapath
    - » switch μarch dependent
- **Switch traversal**
  - move packet to output port
    - » output buffered?
      - speed matching
      - link retry

Deadlock Avoidance

- **Critical needs**
  - avoid cycles
    - » packet can't come back to the same place
  - avoid request reply inter-dependence
    - » natural logical cycle
      - can't incrementally request same resources
- **Topology dependent**
  - fat tree
    - » no problem
      - req-response on different channels
  - 2D mesh or N-D topology
    - » deterministic dimension order routing
  - adaptive routing
    - » more complicated
      - need to limit "how" you adapt

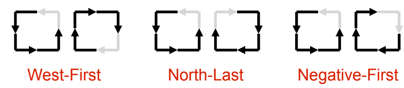
Deterministic 2D Mesh Example

- **Dimension order routing* - deadlock avoidance**
  - **x before y (or vice versa)**
    - » **separates request/reply traffic resource claiming**
      - **add VC's for HOL blocking – no problem**

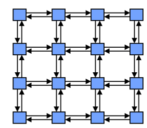


Deadlock Avoidance - Adaptive

- **Still need to avoid cycles**
  - **enter turn model**
    - » **dimension order routing adaptive variant**



- » **modification**
  - **never come back**
    - **incrementally pick 3 of NEWS in some order – problems w/ REQ & REPLY?**



consider – numbering paths
choose any +1 option

Deadlock Avoidance VC's

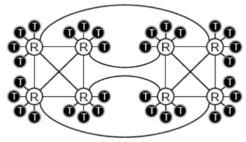
- **Separate VC's into 2 groups**
  - **request & reply**
    - » **each one treated as a separate flow**
      - **deadlock**
        - **dimension order or turn model**
- **OR**
  - **randomly pick a bigger VC**
    - » **to avoid head of line blocking**
    - » **problems?**
      - **assign to last VC**
        - **dimension or turn model limited**

Deadlock Avoidance: N dimensions

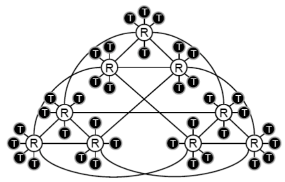
- **Dimension order routing & deterministic**
  - **same as with 2D**
- **Turn model & adaptive**
  - **a bit more complicated**
    - » **simple load balancing scheme**
      - **Valiant – randomly route to another dimension on first hp**
  - **requirement – avoid cycles**
- **N-dimension addressing**
  - **N element vector**
    - » **binary N-cube = 2 nodes per dimension**
      - **example – CalTech Cosmic Cube**
    - » **n-ary N-cube = n elements per dimension**
      - **example HyperX – to appear SC09**
        - **paper on the class web site**
        - **generalization of the flattened butterfly idea**
      - **examine this one since it's a more general case**

HyperX Topology

- **N dimensions**
  - switches in each dimension are fully connected
  - next dimension – link to "mirrors"
    - » L = # dimensions
    - » S_n = # of switches in n^{th} dimension
    - » ignore K for now, T= # terminals per switch (direct network)



(a) $L = 2, S_1 = 2, S_2 = 4, K = 1, T = 4$



(b) $L = 2, S_1 = 3, S_2 = 3, K = 1, T = 4$

HyperX Routing

- **Dimension ordered**
  - pick some order – it works
- **Adaptive = DAL (Whacko!)**
  - significant path diversity
    - » source = N element index
      - aligned dimension: (source – destination) = 0
      - offset dimension: (source – destination) != 0
      - minimum path = # offset dimensions
    - » take any offset dimension for minimal route
      - adaptive = deroute in some dimension
      - mark dimension as derouted
        - – one deroute per dimension to avoid cycles
    - » wait too long
      - move to VC1 for dimension order routing

DAL: Load Latency Graphs

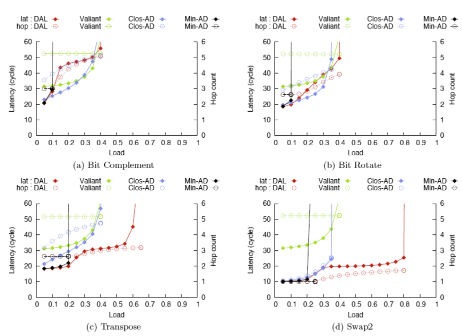


Figure 6: Load-latency graphs on the (a) Bit Complement, (b) Bit Rotate, (c) Transpose, and (d) Swap2 traffic patterns [3] of a regular HyperX network with $N = 4096$, $R = 32$, $\beta = 0.5$, $T = 8$, $S = 8$, $K = 1$.

What's the Point?

- **Topology influences routing algorithm**
- **Routing algorithm influences performance**
  - we've yet to consider switch micro-architecture
    - » it's an influence as well
      - power & latency impact
- **For now**
  - point is
    - » deterministic doesn't take consider congestion
    - » oblivious – e.g. Valiant
      - load balances but doesn't adapt to congestion
    - » DAL
      - more complicated but dynamically adapts to congestion
  - trade-off
    - » more complex = extra overhead in lightly loaded networks
    - » less complex = suffers under near-saturation loads
    - » also observe
      - saturation point

Avoiding Livelock

- Deterministic routing
  - not a problem
- Oblivious
  - adapt once – also no problem
- Adaptive routing
  - key
    - » need some sort of "damping" mechanism
  - DAL
    - » naturally damps
      - no return to aligned dimension
  - common bottleneck
    - » overloaded destination
      - DON'T put packets into orbit – e.g. Post Office
      - adapt early – R2/Fedex adaptive credit model

Concluding Remarks

- Topology and routing algorithm are joined at the hip
  - what do you choose – depends
    - » system size and load
      - over provisioning is common
      - "thin client" model doesn't apply here
        - – more true for bigger systems
- Inherent Catch-22
  - simple = fast under light loads
  - complex = faster under heavy loads
    - » how often does this happen?
      - Amdahl's law applies
- Bottom line
  - as core counts/socket and # sockets increas in the "cloud"
    - » commensurate increase in interconnect bandwidth will be required
    - » cost = f(area, power, latency) will be increasingly important
      - topology and routing algorithm will have a big impact
      - switch μarch as well – next lecture