Routing Algorithms - I

Reading Assignment

- Sections 4.1-4.5
- Alternative: relevant papers.
  - References provided
Overview

• Separate the routing decisions from implementation
  - Choice of path vs.
  - Performance
    - E.g., buffering, route computation implementation

• Routing Algorithm fixed by
  - Routing function
  - Selection function
    - A single function for deterministic protocols

• Primary correctness concern
  - Deadlock and livelock freedom

• Focus in key principles
  - Not coverage of all algorithms designed to date

Virtual Channel Router

Note: Separation of physical and virtual paths, e.g., shortest path may traverse multiple virtual networks
Classes of Routing Algorithms

- **Deterministic and Oblivious Routing**
  - Deterministic protocols are oblivious
    - Converse may not be true
  - Variety of implementations
    - Source Routing, finite state machines, interval routing, table-lookup

- **Partially Adaptive**

- **Fully, maximal, and true fully adaptive**
  - Fully: maximize alternative physical paths
  - Maximal: maximize all routing options
  - True: no constraints on VC usage

- **Non-minimal Routing**

- **Hybrid or multi-phase protocols**
Deterministic Routing Algorithms

- Strictly increasing or decreasing order of dimension
- Routing function always returns the same output channel for each destination

Tori

- Dimension order routing
- Create an acyclic channel dependency graph with the following routing function - $c_{ij}$ when $j < i$, $c_{ji}$ when $j > i$
Physical Router

Deterministic routing

Virtual Channel Router

Remember, we can have virtual lanes
Deterministic Routing Algorithms: Implementation Issues

- Relatively, the most inexpensive to implement
  - Each node implements a routing function
    - Shared or private logic across channels
  - Absolute vs. relative addressing

- Header update
  - Necessary for relative addressing
  - Necessary to maintain uniformity of implementation

- Implementation
  - Table look-up vs. finite state machine

Source Routing

- All routes are pre-computed at the source
  - Stored as a sequence of port addresses at intermediate routers
- Each intermediate router uses the header flit to identify the output port
  - Can be extended to use virtual channels
Characteristics

- Simple fast, route computation at intermediate routers
  - Header update

- Oblivious routing
  - The source-destination pair does not need to identify a unique path → not deterministic

- Complexity of computing paths
  - Ensuring deadlock freedom?

- Extensions to virtual channels
  - We will come back to this

Filling in the Routing Tables

- How do you find deadlock free paths?
  - Non-trivial problem

- Short-cuts
  - Obtain a channel dependency graph for a specific algorithm
  - Find paths in channel dependency graphs!
  - End with virtual channel assignment

Application-Specific Oblivious Routing

Channel Dependence Graph

Two different Channel Dependency Graphs

Flow Graph

Oblivious routing

Interval Routing

<table>
<thead>
<tr>
<th>Channel</th>
<th>Node</th>
<th>Interval</th>
</tr>
</thead>
<tbody>
<tr>
<td>X+</td>
<td>6</td>
<td>8-15</td>
</tr>
<tr>
<td>X-</td>
<td>6</td>
<td>0-3</td>
</tr>
<tr>
<td>Y+</td>
<td>6</td>
<td>4-5</td>
</tr>
<tr>
<td>Y-</td>
<td>6</td>
<td>7</td>
</tr>
</tbody>
</table>

- Each output link corresponds to an interval of nodes
  - Union of intervals at a node is the set of all destination nodes
  - Must be able to distinguish invalid intervals (at the edges of a mesh)
- Use overlapping intervals for fault tolerance

Oblivious routing
Partially Adaptive Routing Algorithms

- Trade-off between hardware resources and adaptivity
  - Maximize adaptivity for given resources
  - Minimize resources required for a given level of adaptivity
- Typically exploit regular topologies


Planar Adaptive Routing

- Packets are routed adaptively in a series of two dimensional planes
  - Order of planes (dimensions) is arbitrary
- Routing in two dimension uses two virtual networks
  - Increasing and decreasing networks
Adaptive Routing in Two Dimensions

Increasing Network \( D_{i+1} \) \( \uparrow \)

Decreasing Network \( D_i \)

Partially Adaptive Routing

PAR in Multidimensional Networks

- Routing is fully adaptive in a plane
- When can you skip a plane?
PAR Properties

- Each plane is comprised of the following channels

\[ A_i = d_{i,2} + d_{i+1,0} + d_{i+1,1} \]

- Three virtual channels/link in meshes and six virtual channels/link in Tori

The Turn Model

- What is a turn?
  - From one dimension to another: 90 degree turn
  - To another virtual channel in the same direction: 0 degree turn
  - To the reverse direction: 180 degree turn

- Turns combine to form cycles
- Goal: prohibit the least number of turns to break all possible cycles
Turn Constraints

• Choice of prohibited turns is not arbitrary

• Alternative designs
  - Three combinations unique (within symmetry)
    - Three algorithms: west-first, north-last, negative-first

---

Partially Adaptive Routing ECE 8813a (23)

---

West First Routing

• Fully adaptive to the east, deterministic to the west (for non-minimal routing)
• Non-minimal is partially adaptive to the west
Generalization of the Turn Model

Application to Binary hypercubes
• Base is e-cube routing
• Identify up channels and down channels
• Adaptively route though one class and then the other
  • Turns prohibited from one class to the other

- Identify channel classes and prohibit turns between them
- Cycles are infeasible within a channel class
- Transitions between channel classes are acyclic
- Partially adaptive: number of shortest paths are reduced

Partially Adaptive Routing

P-Cube Routing

- For a given source destination pair compute the up channel set and down channel set
- Restrict turns from one set to the other
- Adaptively route within a set
- The escape path corresponds to some fixed order which remains a subset of channels
Fully Adaptive Routing

- Using all available paths
  - Minimal vs. non-minimal

- Distinguish between buffer-based schemes vs. channel-based schemes
  - Typically the former can be translated to the latter

Structured Buffer Pools

- Positive hop algorithm
  - $D+1$ buffers at each node ($D =$ diameter)
  - Nodes request/use buffers in strictly increasing order
  - Minimal path algorithm, valid for any topology
  - Large buffer requirements: $O(Diameter)$

Structured Buffer Pools: Extension

- Negative hop algorithm
  - Partition nodes into non-adjacent subsets
  - Order subsets
  - Down transitions request higher numbered buffer, else same numbered buffer
  - Number of buffers required in each node is given by
    \[ \frac{D(S-1)}{S} + 1 \]

Extensions to Wormhole Switching

- Basic version produces unbalanced use of virtual channels
  - distance rarely equal to diameter
- Extension: bonus cards.
  - Number is equal to unused hops: diameter - #req
  - Use this number to increase the number of choices of virtual channels at any node
  - Not the same as adaptivity
Virtual Networks

- Establish multiple virtual networks
  - Each network works for a specific destination set
  - Routing functions in a network are acyclic, but are typically not connected

- Establish routing constraints between virtual networks

- Simple customized protocols
  - Expensive in terms of virtual channels

Virtual Networks in a Mesh

- Each virtual network is constructed to have acyclic channel dependencies
  - Routing function in a virtual network is not connected
- Packets are injected into the appropriate virtual network
- Fully adaptive, no transitions between networks
- \(2^n\) virtual networks with \((n \cdot 2^n)\) virtual channels/node
References


Virtual Networks in a Mesh

- Extensions to 3D
- Extensions to tori and irregular topologies
  - Couple with source routing?
- Extensions to sub-topologies
  - Limit adaptivity to a subset of nodes
• Reduce the number of networks by 50% with one additional channel in dimension 0

• Total number of channels/node
  - Number of channels in each network + extra channel in dimensions 0 (in $2^{n-1}$ networks)

\[ n \times 2^{n-1} + 2^{n-1} = (n+1)2^{n-1} \]

---

• Extensions to $k$-ary $n$-cubes by introducing levels in each virtual network
  - One level for each wrap-around channel
  - At most $n$ levels are traversed (in addition to the original level)
  - $(n+1) \times 2^{(n-1)}$ virtual channels/physical channel for dimensions $>0$
Extensions to Tori

- Addition of layers (virtual networks) for each wrap-around connection
- Number of layers increases by #dimensions

Using Dynamic Message Dependencies

- Two virtual channel classes across each physical link
  - Adaptive channels & deterministic channels
- Fully adaptive use of adaptive channels
  - Keep track of #dimension reversals for each message
    - Moving from a dimension p to a lower dimension q
  - Label each channel with the DR# of the message
  - Messages cannot block on a channel with lower DR#
    - If no channel available, permanent transition to the deterministic channel
  - Dependencies between messages are acyclic

---

Summary of Design Techniques

- Ordered use of topological features
  - Dimensions and paths
- Ordered use of resources
  - Buffers/channels, and networks
- Order the message population
  - Each message is uniquely identified by some attribute, e.g., number of wrap-around channel crossings
  - Order blocking based on message population membership

Design Methodology

- Start with a network, set of channels $C_i$, and routing function $R_i$
  - $R_i$ is connected and deadlock free and may be deterministic/adaptive, minimal/non-minimal
- Split each physical channel into a set of additional virtual channels and define the new routing function

$$R(x, y) = R_i(x, y) \cup (C_{xy} \cap (C - C_i))$$

- Set of channels includes escape channels and adaptive channels
- Selection function can be defined in many ways
- For wormhole switching, verify that the extended channel dependency graph is acyclic: likely if $R$ is restricted to minimal paths
Example Binary Hypercube

- Start with dimension order e-cube algorithm
- Add additional channels for adaptive routing

Maximally Adaptive Routing

- Establish a relationship between routing freedom and resources
  - Maximize adaptivity for fixed resources
  - Minimize resources for target adaptivity

- Relationship between adaptivity and performance
  - Not obvious
  - Unbalanced use of physical or virtual channel resources
References

In 2D Meshes: Mad-Y

- Permit turns
  - From the Y1 channels to the X+ channels
  - From X- channels to Y2 channels

- Remove unnecessary turn restrictions

- Still overly restrictive!

Maximally Adaptive Routing

In 2D Meshes: Opt Y

- Further reduce the number of restrictions
  - Only restrict turns from Y1 to X-
  - Turns from X- to Y1 and 0-degree turns in Y only when X offset is 0 or positive

- Extensions to multidimensional meshes

- Basic idea: fully adaptive routing in one set of channels, and dimension order in the other set until specified lower dimension traversals are complete

Maximally Adaptive Routing
Routing with Minimum Buffer Requirements

- Key Idea:
  - Organize packet traffic into disjoint groups that use separate buffers in each node
  - Place acyclic routing restrictions in buffer usage

- Based on node orderings

Node Labeling for 2D Torus

- Right increasing node ordering
- Left increasing node ordering
- Inside increasing node ordering
- Outside increasing node ordering
Algorithm 1

- Algorithm
  - Packet moves from the injection queue to the A queue
  - Stay in the A queues as long as we can move to right along at least one dimension along a minimal path
  - Transition to the B queues under same rule for left traversals
  - Transition to the C queue and remain there until packet is delivered

- Note the de-coupling of node labeling from buffer labeling

Application

- Orderings analogous to virtual planes
  - Note the orderings are acyclic

- Extensions to edge buffers
  - Check Algorithm 2
True Fully Adaptive Routing

- Adaptivity extends across physical and virtual channels
- Deadlock recovery vs. deadlock avoidance

Maximally Adaptive Routing

Next

- Non-minimal routing
- Topology agnostic routing
- Extensions and application of similar principles