Lecture 5: Deadlocks - I

Tushar Krishna

Assistant Professor
School of Electrical and Computer Engineering
Georgia Institute of Technology

tushar@ece.gatech.edu
Announcement: Lab 2

16-router NoC
Each router connected to one CPU + one Directory
(--num-cpus=16,
  --num dirs=16)
Just like Mesh_XY.py

DO NOT CONNECT 4 CPU/DIR to ONE ROUTER LIKE THE PAPER

Link Weights:
1 for X-direction links
2 for Y-direction links
Taxonomy of Routing Algorithms

- **Classification I: path length**
  - **Minimal**: shortest paths
    - Example: Greedy over Ring, XY over Mesh
  - **Non-minimal**: non-shortest paths
    - Example: Random and Adaptive over Ring/Mesh
Taxonomy of Routing Algorithms

- **Classification II: path diversity** (how to select between the set of all possible paths $R_{xy}$ from the source $x$ to the dest $y$)
  - **Deterministic**: always choose the same route between $x$ and $y$, even if $|R_{xy}| > 1$
    - **Example**: Greedy over Ring, XY over Mesh
    - + Easy to Implement
    - - Inefficient use of bandwidth
  - **Oblivious**: choose any of the routes in $R_{xy}$ without considering any information about current network state (i.e., congestion)
    - **Example**: Random over Ring, O1Turn over Mesh
    - + More path diversity
    - - Can lead to deadlocks (this lecture)
  - **Adaptive**: choose one of the routes in $R_{xy}$ depending on the current network state (i.e., congestion)
    - **Example**: Adaptive over Ring/Mesh
    - + Best use of available bandwidth
    - - Need to track congestion, can lead to deadlocks
Taxonomy of Routing Algorithms

- **Classification III – implementation**
  - Source Routing
  - Node-Table Routing
  - Combinational Circuits

*To be discussed when we discuss router microarchitecture!*
Recap: O1TURN Routing Algorithm

Randomly send over XY or YX

Minimal and Oblivious
Recap: Valiant’s Routing Algorithm

- To route from s to d
  - Randomly choose intermediate node d’
  - Route* from s to d’ (Phase I), and d’ to d (Phase II)

- Pros
  - Randomizes any traffic pattern
    - All patterns appear uniform random
  - Balances network-load
    - Higher throughput

- Cons
  - Non-minimal
    - Higher latency and energy
  - Destroys locality

Non-Minimal and *Oblivious

*can also be Adaptive
Deadlock

- A condition in which a set of **agents** wait indefinitely trying to acquire a set of **resources**

Note: holding buffer u == holding Channel 01 as no other packet can use channel 01 till buffer u becomes free

- Packet A holds buffer u (in 1) and wants buffer v (in 2)
- Packet B holds buffer v (in 2) and wants buffer w (in 3)
- Packet C holds buffer w (in 3) and wants buffer x (in 0)
- Packet D holds buffer x (in 0) and wants buffer u (in 1)
Classic Example:
Dining Philosopher’s Problem

Agents:
Philosophers

Resources:
Forks
Resource Dependence

Resource A is **dependent** on resource B if it is possible for A to be *held-by* an agent X and it is also possible for X to *wait-for* B.

Hold (i.e., wait for release)

Wait to acquire

Resource Dependence Graph
Deadlock Condition

- Agents hold and do not release a resource while waiting for access to another

- A cycle exists between waiting agents such that there exists a set of agents $A_0, \ldots, A_{n-1}$, where agent $A_i$ holds resource $R_i$, while waiting on resource $R_{(i+i \mod n)}$, for $i = 0, \ldots, n-1$

- To avoid deadlock – resource dependence graph should not have any cycles
Dealing with Deadlocks

Avoidance
- Guarantee that the network will never deadlock
- Almost all modern networks use deadlock avoidance

Recovery
- Detect deadlock and correct
Deadlock Avoidance

- Eliminate cycles in Resource Dependency Graph
  - Resource Ordering
    - Enforce a partial/total order on the resources, and insist that an agent acquire the resources in ascending order
    - Deadlock avoided since a cycle must contain at least one agent holding a higher numbered resource waiting for a lower-numbered resource which is not allowed by the ordering allocation
  - Implementation
    - Restrict certain routes so that a higher numbered resource cannot wait for a lower numbered resource
    - Partition the buffers at each node such that they belong to different resource classes. A packet on any route can only acquire buffers in ascending order of resource class
Deadlocks may occur if the turns taken form a cycle

- Removing some turns can make the routing algorithm deadlock free
Dimension Ordered Routing

XY Model

YX Model

O1Turn

Deadlock!
Deadlock-free Oblivious/Adaptive Routing Algorithms

West-First Turn Model

North-Last Turn Model

Negative-First Turn Model
Example 1

```
   d   
   /   
Y---X  Y---X
   /   
   s   

XY?  
YX?  
West-first?  
North-last?
```
Example 2
Resource (channel) Ordering

XY Model

West-First Turn Model
Can we eliminate any 2 turns?

Six turn model

Total Turn Models = 16
Deadlock Free = 12
Unique (non-symmetrical) = 3

Deadlock!
Channel Dependency Graph (CDG)

- Vertices represent network links (channels)
- Edges represent turns
  - $180^\circ$ turns not allowed, e.g., $AB \rightarrow BA$
Cycles in the CDG

The channel dependency graph $D$ derived from the network topology may contain many *cycles*

Flow routed through links $AB$, $BE$, $EF$
Flow routed through links $EF$, $FA$, $AB$
Deadlock!

Edges in CDG = Turns in Network
⇒ Disallow/Delete certain edges in CDG
Acyclic CDG

This is the West-first turn model!

Disable certain edges

Cyclic CDG
CDG for arbitrary topology

Deadlock free?

No

CDG for West-first turn model
Deadlock-free Routing Algorithm

Suppose: Diagonal links should be traversed last (i.e., no edge from blue/green channel to black)
Deadlock-free Routing Algorithm

Suppose: Diagonal links should be traversed last (i.e., no edge from blue/green channel to black)

Deadlock free? Yes
What about a Ring?
Acyclic CDG for a Ring

Route from E to C disabled
(E to D) and (D to C) allowed

Route from F to B disabled

Option 1

Problem? No route from E/F to B/C
Acyclic CDG for a Ring

Route from E to C disabled
(E to D) and (D to C) allowed

Route from E to A disabled

Option 2

Problem? No route from E to A/B/C
Acyclic CDG for a Ring

Route from E to C disabled
(E to D) and (D to C) allowed

Route from B to D disabled

Option 3

Acceptable CDG

Problem? E to C no longer minimal
Acyclic CDG for a Large Ring

Problem?

G, H, I have to take non-minimal paths to reach E!

D, C, B have to take non-minimal paths to reach F
Suppose two channels
Need not be physical channels

Need at least 2 classes of buffers - called “Virtual Channels”

Start in VC in Class0
After Dateline, jump to VC in Class1