Deadlock

Overview: The Problem

- **Guaranteeing message delivery**
  - Fundamentally an issue of finite resources (buffers) and the manner in which they are allocated
  - How are distributed structural hazards handled?

- **Waiting for resources can lead to**
  - Deadlock: configuration of messages that cannot make progress
  - Livelock: messages that never reach the destination
  - Starvation: messages that never received requested resources
  - Even though the requested resource does periodically become available

Ensuring Packet Delivery

- **Deadlock**
  - Prevention
  - Avoidance
  - Recovery

- **Livelock**
  - Minimal paths
  - Restricted non-minimal paths
  - Probabilistic

- **Starvation**
  - Resource allocation

Reading

- Chapter 3 of Duato, Yalamanchili and Ni
Focus

- We are primarily concerned with deadlock
  - Starvation is addressed via “fair” sharing
  - Livelihood is addressed (at the moment) by constraining messages to minimal paths
- Relationship between resources, traffic and conditions
  - Some conditions can produce others
    - Deadlocked configuration \(\rightarrow\) livelock for other messages

Deadlocked Configuration of Messages

- Routing in a 2D mesh
- What can we infer from this figure?
  - Routing
  - Dependencies or Wait-for relationships

Some Definitions

- Routing function – \(R\) – delivers a set of channels
  - Memoryless, coherent, and connected
  - Note we define the domain as \(N \times N\)
- Selection function – picks a channel
  - Determines performance properties

Router and Network Model

- Blocked packets wait for first available channel
- Fair access to channel bandwidth

- Arbitrary topology
- Fair access to routing and arbitration unit
- Edge buffers
  - Theory can be extended to central buffers (we will not)
- Wait for any channel in a set
- For wormhole routing, allocated channel must be empty
  - See Appendix A for detailed list of assumptions
  - Consumption Assumption
**Concepts and Definitions**

- No assumptions about the paths delivered by $R$
  - Expect that minimal paths are always included in $R$
- **Coherent** routing function
  - If a routing function supplies a path, it also supplies all sub-paths of the path
- **Configuration**
  - Assignment of packets to a set of queues/buffers
- **Connected** routing function

**Message Configurations**

- **Canonical** message configurations
  - All messages deadlocked
- **Legal**, and **reachable** message configurations
  - A legal configuration must be possible with the routing function
    - E.g., non-minimal paths
  - Since routing functions are memoryless, all legal configurations are reachable

**Deadlock**

- A canonical, deadlocked, reachable, legal configuration is one which no message can advance
  - SAF and VCT
    - All packets are in transit
    - All adjacent buffers are full
  - Wormhole
    - Messages are in transit
    - Header flits and data flits cannot advance since all adjacent buffers are full
- Framework for proving deadlock freedom

**Channel Dependencies**

- SAF & VCT dependency
- Wormhole dependency
  - When a packet holds a channel and requests another channel, there is a direct dependency between them
  - Channel dependency graph $D = G(C,E)$
  - For deterministic routing: single dependency at each node
  - For adaptive routing: all requested channels produce dependencies, and dependency graph may contain cycles
Some Observations

- Deadlock is not synonymous with the occurrence of cyclic dependencies
  - Note the goal of guaranteeing message delivery
  - Are there exits from cycles?

- The set of dependencies that actually occur (wait-for) are a subset of the statically determined dependencies that can occur as determined by the routing function

Breaking Cycles in Rings/Torii

- The configuration to the left can deadlock
- Add (virtual) channels
  - We can make the channel dependency graph acyclic via routing restrictions (via the routing function)
- Routing function is $c_{0j}$ when $j < i$, $c_{1j}$ when $j > i$

Breaking Cycles in Rings/Torii (cont.)

- Channels $c_{00}$ and $c_{13}$ are unused
- Routing function breaks cycles in the channel dependency graph

Extension to Adaptive Routing

- Note the acyclic subgraph
- Routing function: is $c_{0j}$ $\forall j \neq i$ and $c_{1j}$ when $j > i$
- How can we formalize this?
Key Idea

- Informally: The deterministic routing sub-function is used to provide an escape path for an adaptive routing protocol
  - Packets are guaranteed to escape cyclic dependencies
- Formally the escape path is defined by a connected routing sub-function defined over a subset of channels
  - Routing sub-function supplies only escape channels
  - In practice, this concept only used as a proof mechanism
  - $R_i(x, y) \subseteq R(x, y)$ for all $x, y$ and $C_i$ is the set of channels supplied by $R_i$

Theorems

- **Theorem:** A connected routing function $R$ for an interconnection network $I$ is deadlock free iff there exists a routing sub-function $R_1$ that is connected and has no cycles in its extended channel dependency graph
- Holds for both adaptive and deterministic routing functions
  - In the case of the latter, $R = R_1$ and the extended channel dependency graph = channel dependency graph
  - Dally & Seitz 1987 for deterministic routing
  - Duato 1996 for adaptive routing

Properties

- Use of escape channels by a message is not unidirectional
  - If a message enters the escape network it can move back to the adaptive network
- Do not need to wait on a specific channel
  - Can wait on any channel including escape channels - the latter guaranteed to show up eventually if allocation is fair
- Application: start with a deterministic routing function and add channels to be used for adaptive routing
- Additional restriction required for wormhole (coherent) while this holds for SAF and VCTs

Robustness of the Model

- No assumptions are made about paths
  - At least one minimal path is necessary for performance
- The use of NxN routing functions are practical
  - What are the alternatives?
- No assumptions about packet generation rate, packet length and destinations (other than connectivity)
- For wormhole, only one message can occupy a buffer
  - Buffers at both ends are single buffer
  - Thus head flits always block occupying the head of the queue
  - Cannot access escape channels
Revisit Some Properties

- Canonical legal configuration
- Which legal configurations are reachable from an empty network for a given routing function?
- What is the importance of a legal configuration?
  - Can model a legal configuration as one that occurred by packet injection at the blocked node (VCT/SAF) or the packet with the tail node (wormhole)

Deadlock Avoidance in SAF &VCT Networks

- Routing sub-function $R$ is $c_{Ai}$ when $j < i$, $c_{Hi}$ when $j > i$
- A closer look at escape channel dependencies
  - $c_{Ai}$ is an escape channel for $n0$ but not $n3$
  - $c_{Hi}$ is an escape channel for $n3$
- Extended channel dependency graph must be acyclic

Example: Constructing Deadlock Free Routing Algorithms

- Channels returned by $R$
- Channels returned by $R_i$
- Deterministic Network
- Adaptive Network
Example: Constructing Deadlock Free Routing Algorithms

- R is deadlock free if R1 is deadlock free
- Making the North-last algorithm fully adaptive
  - Add an extra adaptive channel in the North Direction
  - The extra channel has no turn restrictions
  - N1, E, S, and W implement North-Last

Some More Observations

- Can have one escape network for all packets or multiple escape networks
- Construct dependency graph on escape channels
- Deadlock freedom: no cycles in the extended channel dependency graph of R₁
  - Must consider all direct and indirect dependencies
    - There still remain dependencies we have not yet considered!

Indirect Channel Dependencies

- Direct and indirect dependencies between escape channels
- Channel cᵢ is an escape channel for some destination
- Now we can focus all of our attention on the escape channels
  - Add other channels for adaptive routing
  - Define the extended channel dependency graph for escape channels
    - Capture the use of all channels for adaptive routing

Deadlock Avoidance in Wormhole Networks

- Routing function R is cᵢ<sub>AI</sub> or cᵢ<sub>BI</sub> ∀j ≠ i  cᵢ<sub>HI</sub> when j >I
- Routing sub-function R₁ is cᵢ<sub>AI</sub> when j<i, cᵢ<sub>HI</sub> when j >i
- Note the addition of indirect dependencies for Wormhole switching
  - Cross and direct dependencies between escape channels transmitted through adaptive channels
• Routing function R is $c_{Ai}$ or $c_{Bi} \forall j \neq i$, $c_{Hi}$ when $j > i$
• Routing sub-function $R_1$ is $c_{Ai}$ when $j < i$, $c_{Hi}$ when $j > i$
• Note the addition of indirect dependencies for Wormhole switching
  - Cross and direct dependencies between escape channels transmitted through adaptive channels

Example: Fully Adaptive Routing

- Each physical channel has a fully adaptive channel $a$, and DOR channel $b$
- Look at the channels supplied by R at node $S$

Extensions to Irregular Networks

- Create an appropriate routing sub-function
- Example:
  - Identify a distinguished node and construct a spanning tree
  - Label each channel as up/down depending on proximity of end points to root (break ties with unique node number)
  - Routing restriction: a valid path traverses zero or more links in the up direction followed by zero or more links in the down direction
- How does this avoid cycles?
- Note the absence of virtual channels!
Review

- The extended channel dependency graph must capture all dependencies
  - Direct and indirect cross dependencies
- For wormhole switching routing function must be coherent, otherwise the preceding theorem provides only a sufficient condition
- Complete the list of dependencies accommodated by the theorem

Extensions: Channel Classes

- Create equivalence classes of channels
  - For example, with no intra-class dependencies
- If the extended class dependency graph is acyclic, routing is deadlock free (sufficient condition)

Extensions

- Extending the domain of the routing function
  - Including source, channel, or history information
- Extension to central buffers
  - Channels are really queues
  - Associate routing functions with central queues and use queue dependencies to analyze deadlocks
  - Domain of the routing function may be or NxN or QxN
    - Example 2-D mesh: four queues/node, one corresponding to each direction in each dimension.
  - Queue dependency graph is acyclic
  - Mixed resource types (edge buffers and central queues) utilizes similar analysis techniques \(\rightarrow\) the extended resource dependency graph

Summarize Solution Philosophies

- Defined permitted dependencies (prevention)
  - Correct by construction (acyclic)
  - Minimal routing freedom
- Weaken the preceding requirement (avoidance)
  - Ensure cycles cannot lead to deadlock
  - Improve routing freedom
  - Some creative avoidance techniques based on the theory (next)
- Detect and recover (Next)
  - Predicated on unlikeliness of occurrence
  - Maximal routing freedom