Deadlock: Part II - Recovery

Approach

- Employ recovery mechanisms to break deadlocked configurations of messages

- Predicated on the following hypothesis
  - Deadlocks are rare → make them rare
  - Therefore recovery costs are acceptable
  - Recovery is less resource intensive than avoidance
Key Questions

- Probability of occurrence
- Characterization
- On-line detection
- Recovery techniques

Probability of Occurrence

- Influential factors
  - Routing freedom
    - Exponential decrease in probability of occurrence
  - Number of blocked packets
    - Correlated blocking patterns
  - Number of resource dependency cycles
    - Increases with routing freedom
  - Presence of virtual channels
    - Reduces the probability of blocking
Characterization of Deadlocks

- **Deadlock set**
  - Set of messages that are deadlocked

- **Resource set**
  - Set of buffer resources occupied by the deadlocked set

- **Knot cycle density**
  - Number of unique cycles within a knot
    - Captures complexity of formation of deadlocked message configurations

---

Example

- **Deadlock set** – all messages
- **Resource set** – channels \( vc_0 - vc_{15} \)
- **Knot density** - 24
Impact of Routing Freedom

- Addition of additional channel for m₄
- Extension of the range of the routing function sufficient to break the knot
  - Note that it is not possible to reach any other node from VC₁₆

Performance Issues

- Cyclic non-deadlocks can form
  - Extended blocking – “sludgelock”
  - Occurs when available routing freedom is being extensively exploited at high loads
- Deadlock avoidance places acyclic dependency guarantees before routing freedom
- Deadlock recovery emphasizes routing freedom first, and hence can have a performance advantage
On-Line Deadlock Detection

- Exact detection vs. heuristics
  - False detection

- Local vs. centralized detection

- Time-outs
  - At nodes with packet headers
    - Counter + comparator
  - Optimal value depends on message length

- Concurrent recovery
  - Local prioritization for selection of recovery packet

Deadlock Recovery Principles

- Progressive deadlock recovery
  - Robin hood approach – de-allocate resources from normal packets and assign to the recovery packet
    - Remove any message from a deadlocked cycle
    - Ensure its progress towards the destination

- Regressive deadlock recovery
  - De-allocate resources from deadlocked packets
  - Typically destroy packets
  - Need some recovery mechanism, for example, end-to-end flow control
Trade-offs

- Deadlock recovery at the source is typically regressive
  - De-allocate and re-inject

- Deadlock recovery in the network can be both
  - Regressive – propagate de-allocation signals upstream to release resources and abort the packet
  - Progressive – re-allocation of resources from normal to deadlocked packets

Progressive Deadlock Recovery

- Floating virtual channel can be used by all VCs
- Utilized via a separate control path
  - Effectively cycle stealing on the physical channels
- Use of recovery lanes must be deadlock free
Recovering from Deadlock

- Eliminates one cycle

Implementation Issues

- Number of deadlock buffers
  - Impact on crossbar size

- Location
  - Centralized vs. edge
    - Rate at which deadlocked packets can drain
  - At the switch output vs. switch input
Optimizations

- Sequential progressive recovery
  - Only one packet is permitted to enter the deadlock recovery lane
  - Mutual exclusion via a circulating token
  - Recovery lane implements a connected routing sub-function with no cyclic dependencies

- Concurrent recovery
  - Hamiltonian path based
  - Spanning tree based

Summary

- Routing protocols must be designed to be correct
  - Applications to fault tolerance and multicast

- Recovery vs. Avoidance
  - Resource commitment vs. latency impact

- Customized solutions can favor recovery