

Distributed and hierarchical deadlock detection, deadlock resolution

- detection
 - ◆ distributed algorithms
 - Obermarck's path-pushing
 - Chandy, Misra, and Haas's edge-chasing
 - ◆ hierarchical algorithms
 - Menasce and Muntz's algorithm
 - Ho and Ramamoorthy's algorithm
- resolution

Distributed deadlock detection

- Path-pushing
 - ◆ WFG is disseminated as paths — sequences of edges
 - ◆ Deadlock if process detects local cycle
- Edge-chasing
 - ◆ Probe messages circulate
 - ◆ Blocked processes forward probe to processes holding requested resources
 - ◆ Deadlock if initiator receives own probe

Obermarck's Path-Pushing

- Individual sites maintain local WFGs
 - ◆ Nodes for local processes
 - ◆ Node "Pex" represents external processes
 - Pex -> P1 -> P2 -> P3 -> Pex
- Deadlock detection:
 - ◆ site Si finds a cycle that does not involve Pex – deadlock
 - ◆ site Si finds a cycle that does involve Pex – possibility of a deadlock
 - sends a message containing its detected cycle to all other sites
 - to decrease network traffic the message is sent only when $P1 > P3$
 - assumption: the identifier of a process spanning the sites is the same!
 - If site Sj receives such a message, it updates its local WFG graph, and searches it for a cycle
 - If Sj finds a cycle that does not involve its Pex – deadlocks
 - If Sj finds a cycle that does involve its Pex, it sends out a message...
- Can report a false deadlock

Chandy, Misra, and Haas's Edge-Chasing

- When a process has to wait for a resource (blocks), it sends a *probe* message to process holding the resource
- Process can request (and can wait for) multiple resources at once
- Probe message contains 3 values:
 - ◆ ID of process that blocked
 - ◆ ID of process sending message
 - ◆ ID of process message was sent to
- When a blocked process receives a probe, it propagates the probe to the process(es) holding resources that it has requested
 - ◆ ID of blocked process stays the same, other two values updated as appropriate
 - ◆ If the blocked process receives its own probe, there is a deadlock
- size of a message is $O(1)$

Performance evaluation of Obermarck's and Chandy-Misra-Haas algorithms

- Obermarck's
 - ◆ on average(?) only half the sites involved in deadlock send messages
 - ◆ every such site sends messages to all other sites, thus
 - $n(n-1)/2$ messages to detect deadlock
 - for n sites
 - ◆ size of a message is $O(n)$
- Chandy, Misra, and Haas's
 - ◆ given n processes, a process may be blocked on up to $(n-1)$ processes, thus
 - $m(n-1)/2$ messages to detect deadlock
 - m processes, n sites
 - ◆ size of a message is 3 integers

Menasce and Muntz' hierarchical deadlock detection

- Sites (called controllers) are organized in a tree
- Leaf controllers manage resources
 - ◆ Each maintains a local WFG concerned only about its own resources
- Interior controllers are responsible for deadlock detection
 - ◆ Each maintains a global WFG that is the union of the WFGs of its children
 - ◆ Detects deadlock among its children
- changes are propagated upward either continuously or periodically

Ho and Ramamoorthy's hierarchical deadlock detection

- Sites are grouped into disjoint clusters
- Periodically, a site is chosen as a central control site
 - ◆ Central control site chooses a control site for each cluster
- Control site collects status tables from its cluster, and uses the Ho and Ramamoorthy one-phase centralized deadlock detection algorithm to detect deadlock in that cluster
- All control sites then forward their status information and WFGs to the central control site, which combines that information into a global WFG and searches it for cycles
- Control sites detect deadlock in clusters
- Central control site detects deadlock between clusters

Estimating performance of deadlock detection algorithms

- Usually measured as the number of messages exchanged to detect deadlock
 - ◆ Deceptive since message are also exchanged when there is no deadlock
 - ◆ Doesn't account for size of the message
- Should also measure:
 - ◆ Deadlock persistence time (measure of how long resources are wasted)
 - ~ Tradeoff with communication overhead
 - ◆ Storage overhead (graphs, tables, etc.)
 - ◆ Processing overhead to search for cycles
 - ◆ Time to optimally recover from deadlock

Deadlock resolution

- resolution – aborting at least one process (*victim*) in the cycle and granting its resources to others
- efficiency issues of deadlock resolution
 - ◆ fast – after deadlock is detected the victim should be quickly selected
 - ◆ minimal – abort minimum number of processes, ideally abort less "expensive" processes (with respect to completed computation, consumed resources, etc.)
 - ◆ complete – after victim is aborted, info about it quickly removed from the system (no phantom deadlocks)
 - ◆ no starvation – avoid repeated aborting of the same process
- problems
 - ◆ detecting process may not know enough info about the victim (propagating enough info makes detection expensive)
 - ◆ multiple sites may simultaneously detect deadlock
 - ◆ since WFG is distributed removing info about the victim takes time