Distributed Deadlock Detection

Slides are based on the book chapter from Distributed Computing: Principles, Algorithms and Systems (Chapter 10) by Kshemkalyani and Singhal

Pradipta De
pradipta.de@sunykorea.ac.kr
Deadlocks?

- Communication deadlock
  - A → B → C → A, but the buffers have run out

- Resource deadlock
  - Competing for exclusive access to same resource
Deadlock Handling Strategies

• Deadlock Prevention
  – acquire all required resources simultaneously, or preempt another process that holds the resource
    • Inefficient (can limit concurrency) and impractical (requires strict ordering of granting resources)

• Deadlock Avoidance
  – A resource is granted to a process if resulting global state is safe
  – Requires processes to report expected usage of a resource in future
  – Djikstra’s Bankers algorithm

• Deadlock Detection
  – Examine the process resource interaction across all processes, and look for some property
Wait-for-Graph

Wait-for-Graph (WFG) denotes processes as nodes, and a directed edge from Pi to Pj denotes that Pi is waiting for Pj to release a resource.

A cycle in a WFG indicates that there is a deadlock.
Resource Request Models

• Single resource model
• The AND model
• The OR model
• The AND-OR model
• The (p-out-of-q) model
Centralized Solution

• One control site maintains the WFG
• WFG on the control site is kept up-to-date
  – On any change to resource graph, send message to control site
  – Periodically send message to control site
  – Control site pulls information from each site on demand
• Control site checks for cycle in the WFG to detect deadlock
False Deadlock

Resource R1@S1, and R2@S2
Transaction T1@S3 and T2@S4 starts at the same time

\[
\begin{align*}
\text{T1:} & \quad \text{Lock R1} \\
& \quad \text{Unlock R1} \\
& \quad \text{Lock R2} \\
& \quad \text{Unlock R2}
\end{align*}
\]

\[
\begin{align*}
\text{T2:} & \quad \text{Lock R1} \\
& \quad \text{Unlock R1} \\
& \quad \text{Lock R2} \\
& \quad \text{Unlock R2}
\end{align*}
\]

Can be avoided by using Lamport’s global time
Classification of Distributed Deadlock Detection Algorithms

- **path-pushing**
  - path information transmitted, accumulated

- **edge-chasing**
  - probes are sent along edges
  - A blocked process propagates the probe along its outgoing edges in WFG
  - single returned probe initiated by the process indicates a cycle

- **diffusion**
  - Query probes are sent along all edges
  - Running processes discard query
  - all queries returned indicates a cycle

- **global state detection**
  - take and use snapshot of system state
Path Pushing Algorithm

• The site waits for deadlock related information from other sites that are deadlocked

• The site combines the received information with its local WFG to build an updated WFG

• For all cycles ‘EX > T1 > T2 > Ex’ which contains the node ‘Ex’, the site transmits them in string form ‘Ex, T1, T2, Ex’ to all other sites where a sub transaction of T2 is waiting to receive a message from the sub transaction of T2 at that site
Edge-Chasing Algo: Chandy et al.

- The algorithm works for AND request model
- Uses a special message, probe, which is a triplet $(i,j,k) \Rightarrow$ process $P_i$ initiated deadlock detection, and the probe is being forwarded by home site of $P_j$ to the home site of $P_k$
- Probe travels along the edges of the WFG
- Deadlock detected when probe returns to the initiating process
To determine if a blocked process is deadlocked

if $P_i$ is locally dependent on itself
then declare a deadlock
else for all $P_j$ and $P_k$ such that
(a) $P_i$ is locally dependent upon $P_j$, and
(b) $P_j$ is waiting on $P_k$, and
(c) $P_j$ and $P_k$ are on different sites
send probe $(i, j, k)$ to the home site of $P_k$
Edge-Chasing Algo: Chandy et al.

On receiving probe \((i,j,k)\):

if (a) \(P_k\) is blocked, and
(b) \(dependent_k(i)\) is false, and
(c) \(P_k\) has not replied to all requests of \(P_j\)

then begin

\(dependent_k(i) = true\)

if \(k=i\), then declare that \(P_j\) is deadlocked

else for all \(P_m\) and \(P_n\) such that
(i) \(P_k\) is locally dependent upon \(P_m\),
(ii) \(P_m\) is waiting on \(P_n\),
(iii) \(P_m\) and \(P_n\) are on different sites,

send probe \((i, m, n)\) to the home site of \(P_n\)
Example

- Initiates Deadlock Detection
- Sends probe on behalf of P1
- P6 and P7 forwards the probe
Performance -- Edge Chasing: Chandy

• One probe per message on each edge of WFG, spanning across two sites

• Message complexity:
  – $m(n-1)/2$, where deadlock involves $m$ processes and spanning across $n$ sites

• Delay in detecting deadlock: $O(n)$
Global State Detection Based Algorithm

• Exploit the following facts:
  – A snapshot of the execution can be obtained without pausing the execution
  – If there is a deadlock, the state will not change ➔
    deadlock is a stable property