Concurrency control &
Deadlock detection in distributed systems

presented by
Christoph Jud

cs341 – AS 2008
You know what a deadlock is!
Overview

• Deadlock example
• CORBA Exercise
• Deadlock strategies
  – Ignore the problem
  – Prevention
  – Avoidance
  – Detection
• Deadlock models
• Some deadlock detection algorithms
Mickey Mouse example

Current current = null;
try {
    current = atmServer.getCurrent();
    current.begin();

    getBank(bankToId).deposit(accountToId, amount);
    Thread.sleep(10000);
    getBank(bankFromId).withdraw(accountFromId, amount);

    current.commit(false);
} catch (Exception e) {
    ...
    current.rollback();
    ...
}

transfer(Bank1,Account1,Bank2,Account2,100);

transfer(Bank2,Account2,Bank1,Account1,100);
Locking

Concurrency control

List of locked resources.
- Account1
- Account2

ATM

Bank1
- Account1

Bank2
- Account2
Current current = null;
try {
    current = atmServer.getCurrent();
    current.begin();

    if (accountFromId.compareTo(accountToId) < 0 ){
        atmServer.lock(accountFromId, 1);
        atmServer.lock(accountToId, 1);
    }
    else{
        atmServer.lock(accountToId, 1);
        atmServer.lock(accountFromId, 1);
    }

    getBank(bankToId).deposit(accountToId, amount);
    Thread.sleep(10000);
    getBank(bankFromId).withdraw(accountFromId, amount);

    current.commit(false);
} catch (Exception e) {...
} finally{
    atmServer.unlock(accountToId, 1);
    atmServer.unlock(accountFromId, 1);
}
Overview

• Deadlock example
• CORBA Exercise
• Deadlock strategies
  – Ignore the problem
  – Prevention
  – Avoidance
  – Detection
• Deadlock models
• Some deadlock detection algorithms
Ignore the problem

- Nothing is done against deadlocks
- The most efficient method
  - But there is an appropriate likelihood that a deadlock occurs
Deadlock prevention

• All required resources are first locked
  – No deadlock can occur
• Highly inefficient
Deadlock avoidance

- Resources are only granted to a process if the global system state is safe.
- Allows more concurrency
Deadlock detection

• The best approach in distributed systems
• allows a high concurrency
• A state of process-resource interaction have to be known to detect cyclic waiting
  – Wait-for graph (WFG)
When is a process deadlocked?

• Think in an algorithmic view
Deadlock models

- Single resource model
Deadlock models

• AND model
Deadlock models

• OR model
Further deadlock models

- AND-OR model
- P-out-of-Q model
- Unrestricted model
Classes of algorithms

• Path-pushing algorithms
  – build global WFG
• Edge-chasing algorithms
• Diffusing computation-based algorithms
• Global state detection-based algorithms
  – take snapshot of the system and find deadlocks
    • without freezing the underlying system
    • without global time
Chandy-Misra-Haas algorithm

• For AND models
• Is an edge-chasing algorithm

deadlocked?

{P_{11}, P_{11}, P_{32}}

{P_{11}, P_{21}, P_{24}}

{P_{11}, P_{32}, P_{33}}

{P_{11}, P_{54}, P_{11}}

consumed
Chandy-Misra-Haas (AND)

- Complexity:
  - \( m(n-1)/2 \) messages to detect a deadlock involving \( m \) processes spanned over \( n \) sites

- Small and fixed message type
Chandy-Misra-Haas algorithm

• For OR models
• Is a diffusion computation algorithm
Chandy-Misra-Haas algorithm

query: \{P_{11}, P_{11}, P_{32}\}

num_{11}(i) = 2
wait_{11}(i) = true

query: \{P_{11}, P_{21}\}

num_{32}(i) = 1
wait_{32}(i) = true

num_{54}(i) = 1
wait_{54}(i) = true

query: \{P_{11}, P_{54}, P_{11}\}

query consumed

query: \{P_{11}, P_{24}, P_{54}\}

query: \{P_{11}, P_{24}, P_{33}\}

query: \{P_{11}, P_{32}, P_{33}\}

query consumed

num_{21}(i) = 1
wait_{21}(i) = true

num_{24}(i) = 1
wait_{24}(i) = true
Chandy-Misra-Haas algorithm

not deadlocked, only blocked!

\[
\begin{align*}
\text{num}_{11}(i) &= 1 \\
\text{wait}_{11}(i) &= \text{true} \\
\text{reply:} &\quad \{P_{11}, P_{11}, P_{54}\} \\
\text{num}_{54}(i) &= 0 \\
\text{wait}_{54}(i) &= \text{true} \\
\text{reply:} &\quad \{P_{11}, P_{54}, P_{24}\} \\
\text{num}_{24}(i) &= 0 \\
\text{wait}_{24}(i) &= \text{true} \\
\text{reply:} &\quad \{P_{11}, P_{24}, P_{21}\} \\
\text{num}_{21}(i) &= 1 \\
\text{wait}_{21}(i) &= \text{true} \\
\text{reply:} &\quad \{P_{11}, P_{21}, P_{11}\} \\
\text{num}_{24}(i) &= 1 \\
\text{wait}_{24}(i) &= \text{true} \\
\text{num}_{32}(i) &= 1 \\
\text{wait}_{32}(i) &= \text{true} \\
\end{align*}
\]
Chandy-Misra-Haas (OR)

• Complexity:
  – $e$ query messages and $e$ reply messages where $e$ is the number of edges

• Small and fixed message type
Kshemkalani-Singhal algorithm

- For P-out-of-Q models (generalized deadlocks)
- Based on the global state detection approach
- Relatively complex but the most efficient algorithm that exist for P-out-of-Q models
References

• Distributed Computing (2007)
  – Ajay Kshemkalyani, Mukesh Singhal
  – isbn: 978-0-521-87634-6

• Weblink:
  – www.cs.jhu.edu/~yairamir/cs418/os4/sld001.htm
Questions