Scheduling Algorithm and Analysis

Priority Inversion
(Module 32)

Yann-Hang Lee
Arizona State University
yhlee@asu.edu
(480) 727-7507

Summer 2014
Priority Inversion in Synchronization

\( \tau_1 : \{ \ldots P(S1) \ldots V(S1) \ldots \} \)

\( \tau_3 : \{ \ldots P(S1) \ldots V(S1) \ldots \} \)

Attempt to lock S1 (blocked)

\( \tau_1(H) \)

\( \tau_2(M) \)

\( \tau_3(L) \)

S1 locked

S1 unlocked

Blocked

Time
Priority Inversion

- Delay to a task’s execution caused by interference from lower priority tasks is known as *priority inversion*.

- Priority inversion is modeled by *blocking time*.

- Identifying and evaluating the effect of sources of priority inversion is important in schedulability analysis.

**Sources of priority Inversion**

- Synchronization and mutual exclusion
- Non-preemptable regions of code
- FIFO (first-in-first-out) queues
Accounting for Priority Inversion

- Recall that task schedulability is affected by
  - preemption: two types of preemption
    - can occur several times per period
    - can occur once per period
  - execution: once per period
  - blocking: at most once per period for each request to a source

- The schedulability formulas are modified to add a “blocking” or “priority inversion” term to account for inversion effects
UB Test with Blocking

Include blocking while calculating effective utilization for each task:

\[ f_i = \sum_{j \in H_n} \frac{e_j}{p_j} + \frac{e_i}{p_i} + \frac{B_i}{p_i} + \frac{1}{p_i} \sum_{k \in H_1} e_k \]

- \( H_n \) Preemption (can hit \( n \) times)
- Execution
- Blocking
- \( H_1 \) Preemption (can hit once)
*RT Test with Blocking*

- Blocking is also included in the RT test

\[
a_{n+1} = B_i + e_i + \sum_{j=1}^{i-1} \left[ \frac{a_n}{\rho_j} \right] e_j
\]

where \( a_0 = B_i + \sum_{j=1}^{i} e_j \)

- Perform test as before, including blocking effect
Consider the following example

Periodic tasks

\( \tau_1 \) 100 msec

25 msec

\( \tau_2 \) 200 msec

50 msec

\( \tau_3 \) 300 msec

100 msec

Data Structure

10 msec

30 msec

What is the worst case blocking effect (priority inversion) experienced by each task?
Example: Adding Blocking

- Task $\tau_2$ does not use the data structure. Task $\tau_2$ does not experience any priority inversion.
- Task $\tau_1$ shares the data structure with $\tau_3$. Task $\tau_1$ could have to wait for $\tau_3$ to complete its critical section. But worse, if $\tau_2$ preempts while $\tau_1$ is waiting for the data structure, $\tau_1$ could have to wait for $\tau_2$'s entire computation.
- This is the resulting table:

<table>
<thead>
<tr>
<th>task</th>
<th>Period</th>
<th>Execution Time</th>
<th>Priority</th>
<th>Blocking delay</th>
<th>Deadline</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tau_1$</td>
<td>100</td>
<td>25</td>
<td>High</td>
<td>30+50</td>
<td>100</td>
</tr>
<tr>
<td>$\tau_2$</td>
<td>200</td>
<td>50</td>
<td>Medium</td>
<td>0</td>
<td>200</td>
</tr>
<tr>
<td>$\tau_3$</td>
<td>300</td>
<td>100</td>
<td>Low</td>
<td>0</td>
<td>300</td>
</tr>
</tbody>
</table>
UB Test for Example

- UB test with blocking:

\[
f_i = \sum_{j \in H_i} \frac{e_j}{p_j} + \frac{e_i}{p_i} + \frac{B_i}{p_i} + \frac{1}{p_i} \sum_{k \in H_i} e_k
\]

\[
f_1 = \frac{e_1}{p_1} + \frac{B_1}{p_1} = \frac{25}{100} + \frac{80}{100} = 1.05 > 1.00 \quad \text{Not schedulable}
\]

\[
f_2 = \frac{e_1}{p_1} + \frac{e_2}{p_2} = \frac{25}{100} + \frac{50}{200} = 0.5 < U(2)
\]

\[
f_3 = \frac{e_1}{p_1} + \frac{e_2}{p_2} + \frac{e_3}{p_3} = \frac{25}{100} + \frac{50}{200} + \frac{100}{300} = 0.84 > U(3)
\]

with additional RT test, \( \tau_3 \) is shown to be schedulable
Supplementary Slides