Lecture 5

Fixed Priority Scheduling

- Fixed-priority online scheduling
- Rate-Monotonic scheduling
  - The CPU utilization bound
- Deadline-Monotonic and arbitrary priorities
  - Worst-Case Response-Time analysis
Last lecture (4)

Task scheduling basics

• The concept of temporal complexity
• Definition of schedule and scheduling algorithm
• Some basic scheduling techniques (EDD, EDF, BB)
• The static cyclic scheduling technique
Online scheduling with fixed priorities

The schedule is built while the system is operating normally (online) and is based in a static criterium (priority).

The ready queue is sorted by decreasing priorities. Executes first the task with highest priority.

If the system is preemptive, whenever a task job arrives to the ready queue, if it has higher priority than the one currently executing, it starts executing while the latter one is moved to the ready queue.

Complexity: $O(n)$
Online scheduling with fixed priorities

- **Pros**
  - Scales
  - Changes on the task set are immediately taken into account by the scheduler
  - Sporadic tasks are easily accommodated
  - Deterministic behavior on overloads
    - Tasks are affected by priority level (lower priority are the first ones)

- **Cons**
  - More complex implementation
  - Requires a kernel with support to fixed priorities
  - Higher execution overhead (scheduler + dispatcher)
  - Overloads (e.g. due to programming errors or unpredicted events) at higher priority tasks may block the execution of lower priority ones
Online scheduling with fixed priorities

Priority assignment to tasks
• Inversely proportional to period (RM – Rate Monotonic)
  Optimal among fixed priority scheduling criteria
• Inversely proportional to deadline (DM – Deadline Monotonic)
  Optimal if D<=T
• Proportional to the task importance
  Typically reduces the schedulability – not optimal
Online scheduling with fixed priorities

Schedulability tests

As the schedule is built online it is fundamental to know a priori if a given task set is schedulable (i.e., its temporal requirements are met)

There are two types of schedulability tests:

• Based on CPU utilization rate
• Based on response time
Schedulability tests for RM based on task utilization
(with preemption, *n independent tasks*, D=T)

- Liu&Layland's (1973) *Least Upper Bound*

\[
U(n) = \sum_{i=1}^{n} \frac{C_i}{T_i} \leq n \left(2^n - 1\right) \Rightarrow \text{One activation per period guaranteed}
\]

- Bini&Buttazzo&Buttazzo's (2001) *Hyperbolic Bound*

\[
\prod_{i=1}^{n} \left(\frac{C_i}{T_i} + 1\right) \leq 2 \Rightarrow \text{One activation per period guaranteed}
\]
RM Scheduling

**Interpretation of the Liu&Layland test**

\[ U(n) > 1 \implies \text{task set not schedulable (overload)} \quad \text{– necessary condition} \]
\[ U(n) \leq \text{Bound} \implies \text{task set is schedulable} \quad \text{– sufficient condition} \]
\[ 1 \geq U(n) > \text{Bound} \implies \text{test is indeterminate} \]

Liu&Layland

\[ U(1) \leq 1 \]
\[ U(2) \leq 0.83 \]
\[ U(3) \leq 0.78 \]
\[ \ldots \]
\[ U(\infty) \leq \ln(2) \approx 0.69 \]
### Task properties

<table>
<thead>
<tr>
<th>( \tau_i )</th>
<th>( T_i )</th>
<th>( C_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>0.5</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>0.5</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>2</td>
</tr>
</tbody>
</table>

\[
U = \frac{0.5}{2} + \frac{0.5}{3} + \frac{2}{6} \\
U = 0.75 < 0.78 \implies 1 \text{ activation per period is guaranteed}
\]
RM Scheduling – example 2

<table>
<thead>
<tr>
<th>$\tau_i$</th>
<th>$T_i$</th>
<th>$C_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>0.5</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>0.5</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>3</td>
</tr>
</tbody>
</table>

$U = 0.5/2 + 0.5/3 + 3/6 = 0.92 > 0.78 \implies 1$ activation per period is not guaranteed.
However it is feasible.
RM Scheduling – example 3

Task properties

<table>
<thead>
<tr>
<th>$\tau_i$</th>
<th>$T_i$</th>
<th>$C_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>2.1</td>
</tr>
</tbody>
</table>

$U = \frac{1}{3} + \frac{1}{4} + \frac{2.1}{6} = 0.93 > 0.78 \implies 1$ activation per period not guaranteed, with deadline miss by $\tau_3$
RM Scheduling – particular case

\[ U = \frac{1}{2} + \frac{2}{4} = 1 \]

Harmonic periods

Schedulable iif \( U(n) \leq 1 \)
In some cases tasks may have large periods (low frequency) but require a short response time. In these cases we assign a deadline shorter than the period, and the scheduling criteria is the deadline.

In these cases we can also use utilization-based tests. The adaptation is simple, but the test is very pessimistic.

\[
U'(n) = \sum_{i=1}^{n} \frac{C_i}{D_i} \leq n \left(2^n - 1\right)
\]
For arbitrary fixed priorities, including RM, DM e others, the response time analysis allow to obtain an exact test (i.e., necessary and sufficient condition) in the following conditions: preemption, synchronous release, independent tasks and \( D \leq T \)

Worst-case response time (WCRT) = maximum time interval between arrival and finish instants. \[ R_{wc_i} = \max_k (f_{i,k} - a_{i,k}) \]

Schedulability test based on the WCRT

Compute \( R_{wc_i} \) \( \forall \ i \)

\( \forall \ i, R_{wc_i} \leq D_i \iff \) Task set is schedulable
Response-time analysis

The WCRT of a given task occurs when the task is activated at the same time as all other high-priority tasks (critical instant).

Computing $R_{wc_i}$:

$$\forall_i R_{wc_i} = I_i + C_i$$

$$I_i = \sum_{k \in hp(i)} \left\lceil \frac{R_{wc_i}}{T_k} \right\rceil \times C_k$$

Number of times that higher priority task $k$ is activated in the $R_{wc_i}$ time interval

$l_i$ – interference caused by the execution of higher priority tasks
Response-time analysis

The equation is solved iteratively. Stop conditions are:

• **A deadline is violated** \((R_{wc_i}>D_i)\)

• **Convergence** (two successive iterations yield the same result)
  - \(R_{wc_i}(m+1)=R_{wc_i}(m)\)

\[
R_{wc_i}(0) = \sum_{k \in hp(i)} C_k + C_i
\]

......

\[
R_{wc_i}(m+1) = \sum_{k \in hp(i)} \left[ \frac{R_{wc_i}(m)}{T_k} \right] C_k + C_i
\]
# Response-time analysis

### Critical instant

<table>
<thead>
<tr>
<th>$\tau_i$</th>
<th>$T_i$</th>
<th>$C_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>0.5</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>0.5</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>3</td>
</tr>
</tbody>
</table>

$Rwc_1$: ?  
$Rwc_2$: ?

---

**DETI * STR 2014/2015**

17
Response-time analysis

Task properties

<table>
<thead>
<tr>
<th>$\tau_i$</th>
<th>$T_i$</th>
<th>$C_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>0.5</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>0.5</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>3</td>
</tr>
</tbody>
</table>

$Rwc_1$: $Rwc_1(0) = C_1 = 0.5$
$Rwc_2$: $Rwc_2(0) = C_1 + C_2 = 1$
$Rwc_2(1) = \left[ \frac{Rwc_2(0)}{T_1} \right] * C_1 + C_2 = 1$
$Rwc_2 = 1$
Response-time analysis

Task properties

<table>
<thead>
<tr>
<th>( \tau_i )</th>
<th>( T_i )</th>
<th>( C_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>0.5</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>0.5</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>3</td>
</tr>
</tbody>
</table>

Critical instant

\( Rwc_3: \) ?
Response-time analysis

Task properties

<table>
<thead>
<tr>
<th>( \tau_i )</th>
<th>( T_i )</th>
<th>( C_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>0.5</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>0.5</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>3</td>
</tr>
</tbody>
</table>

Rwc\(_3\):  

\[
\begin{align*}
Rwc_3(0) &= C_1 + C_2 + C_3 = 4 \\
Rwc_3(1) &= \left[ Rwc_3(0)/T_1 \right]C_1 + \left[ Rwc_3(0)/T_2 \right]C_2 + C_3 = 5 \\
Rwc_3(2) &= \left[ Rwc_3(1)/T_1 \right]C_1 + \left[ Rwc_3(1)/T_2 \right]C_2 + C_3 = 5.5 \\
Rwc_3(3) &= \left[ Rwc_3(2)/T_1 \right]C_1 + \left[ Rwc_3(2)/T_2 \right]C_2 + C_3 = 5.5 \\
Rwc_3 &= 5.5
\end{align*}
\]
Restrictions to the schedulability tests previously presented

The previous schedulability tests must be modified in the following cases:

- Non-preemption
- Tasks not independent
  - Share mutually exclusive resources
  - Have precedence constraints
- It is also necessary to take into account the overhead of the kernel, because the scheduler, dispatcher and interrupts consume CPU time
Impact of non-preemption

Executes without preemption

Block and deadline miss

Task properties

<table>
<thead>
<tr>
<th>$\tau_i$</th>
<th>$T_i$</th>
<th>$C_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>0.5</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>0.5</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>3</td>
</tr>
</tbody>
</table>
Summary of 5

- **On-line** scheduling with **fixed-priorities**
- The **Rate Monotonic** scheduling policy – schedulability analysis based on utilization
- The **Deadline Monotonic** and arbitrary deadlines scheduling policies
- Response-time analysis