Abstract

The choice of scheduling technique and its parametrization impacts the performance of systems. This is an area where quite some theoretical work has been done. In this presentation we address Earliest Deadline First and Rate Monotonic Scheduling (RMS). We provide how-to information for RMS, based on Rate Monotonic Analysis (RMA).
Theory Block Scheduling Techniques and Analysis

Theory Hard Real Time Scheduling

Earliest Deadline First (EDF)

Rate Monotonic Scheduling (RMS)
Real Time Scheduling

Scheduler

Priorities

Ready

Queue

Wait

Queue

Scheduler admin

Run

Proc. 1
Prio. High
State ready

Proc. 2
Prio. Med.
State ready

Proc. 3
Prio. High
State ready

. . .

Process / tasks instances

Proc. 1

Proc. 2

Proc. 3

. . .
### Earliest Deadline First

<table>
<thead>
<tr>
<th>• Determine deadlines</th>
<th>in <strong>Absolute time</strong> (CPU cycles or msec, etc.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Assign priorities</td>
<td>Process that has the earliest deadline gets the highest priority (no need to look at other processes)</td>
</tr>
<tr>
<td>• Constraints</td>
<td>Smart mechanism needed for Real-Time determination of deadlines Pre-emptive scheduling needed</td>
</tr>
</tbody>
</table>

**EDF = Earliest Deadline First**

Earliest Deadline based scheduling for (a-)periodic Processing

The theoretical limit for any number of processes is 100% and so the system is schedulable.
Exercise Earliest Deadline First (EDF)

Calculate loads and determine thread activity (EDF)

<table>
<thead>
<tr>
<th>Thread</th>
<th>Period = deadline</th>
<th>Processing</th>
<th>Load</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thread 1</td>
<td>9</td>
<td>3</td>
<td>33.3%</td>
</tr>
<tr>
<td>Thread 2</td>
<td>15</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Thread 3</td>
<td>23</td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>

Suppose at t=0, all threads are ready to process the arrived trigger.

Source: Ton Kostelijk - EXARCH course
## Rate Monotonic Scheduling

<table>
<thead>
<tr>
<th>• Determine deadlines (period)</th>
<th>in terms of Frequency or Period (1/F)</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Assign priorities</td>
<td>Highest frequency (shortest period)</td>
</tr>
<tr>
<td></td>
<td>==&gt; Highest priority</td>
</tr>
<tr>
<td>• Constraints</td>
<td>Independent activities</td>
</tr>
<tr>
<td></td>
<td>Period</td>
</tr>
<tr>
<td></td>
<td>Constant CPU cycle consumption</td>
</tr>
<tr>
<td></td>
<td>Assumes Pre-emptive scheduling</td>
</tr>
</tbody>
</table>

RMS = Rate Monotonic Scheduling

Priority based scheduling for Periodic Processing of tasks with a guaranteed CPU - load
Exercise Rate Monotonic Scheduling (RMS)

Calculate loads and determine thread activity (RMS)

<table>
<thead>
<tr>
<th>Thread</th>
<th>Period = deadline</th>
<th>Processing</th>
<th>Load</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thread 1</td>
<td>9</td>
<td>3</td>
<td>33.3%</td>
</tr>
<tr>
<td>Thread 2</td>
<td>15</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Thread 3</td>
<td>23</td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>

Suppose at t=0, all threads are ready to process the arrived trigger.

Source: Ton Kostelijk - EXARCH course
Real-time scheduling theory, utilization bound

- Set of tasks with periods $T_i$, and process time $P_i$: load $u_i = P_i / T_i$

- Schedule is at least possible when tasks are independent and:

$$\text{Load} \equiv \sum U_i \leq n \left( \frac{1}{2^n} - 1 \right)$$

- $1.00, 0.83, 0.78, 0.76, \ldots \quad \log(2) = 0.69$

Source: Ton Kostelijk - EXARCH course
• RMS cannot utilize 100% (1.0) of CPU, but for 1, 2, 3, 4, ... \( \infty \) processes:
  1.00 , 0.83 , 0.78 , 0.76 , ... \( \log(2) = 0.69 \)

• RMS guarantees that all processes will always meet their deadlines, for any interleaving of processes.

• With fixed priorities, context switch overhead is limited

Source: Ton Kostelijk - EXARCH course
For specific cases the utilization bound can be higher:
up to 0.88 load for large $n$

A processor running only hard-real-time processes is rare. For soft-RT less of a problem

A lot of additional theory exists.
Meeting deadlines in hard-real-time systems
(L.P. Briand & D.M. Roy)

Source: Ton Kostelijk - EXARCH course
**Answers: loads and thread activity (EDF)**

<table>
<thead>
<tr>
<th>Thread</th>
<th>Period = deadline</th>
<th>Processing</th>
<th>Load</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thread 1</td>
<td>9</td>
<td>3</td>
<td>33.3%</td>
</tr>
<tr>
<td>Thread 2</td>
<td>15</td>
<td>5</td>
<td>33.3%</td>
</tr>
<tr>
<td>Thread 3</td>
<td>23</td>
<td>5</td>
<td>21.7%</td>
</tr>
</tbody>
</table>

88.3%

Source: Ton Kostelijk - EXARCH course
Answers: loads and thread activity (RMS)

<table>
<thead>
<tr>
<th>Thread</th>
<th>Period = deadline</th>
<th>Processing</th>
<th>Load</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thread 1</td>
<td>9</td>
<td>3</td>
<td>33.3%</td>
</tr>
<tr>
<td>Thread 2</td>
<td>15</td>
<td>5</td>
<td>33.3%</td>
</tr>
<tr>
<td>Thread 3</td>
<td>23</td>
<td>5</td>
<td>21.7%</td>
</tr>
</tbody>
</table>

88.3%

Source: Ton Kostelijk - EXARCH course
Extensions of the Application of RMS

*if* deadline <> 1/period   
then use period = 1/deadline

*if* CPU consumption varies  
then use worst case CPU consumption

*More advanced techniques are available, for instance in case of "nice" frequencies*
Summary

Theory Hard Real Time Scheduling

Earliest Deadline First (EDF):

- optimal according theory, but practical not applicable due to overhead

Rate Monotonic Scheduling (RMS):

- provides recipe to assign priorities to tasks
- results in predictable real time behavior
- works well, even outside theoretical constraints
The ASP™ course is partially derived from the EXARCH course developed at Philips CTT by Ton Kostelijk and Gerrit Muller.

Extensions and additional slides have been developed at ESI by Teun Hendriks, Roland Mathijssen and Gerrit Muller.