Scheduling Techniques and Analysis

by *Gerrit Muller* HSN-NISE
e-mail: `gaudisite@gmail.com`
`www.gaudisite.nl`

Abstract

The choice of scheduling technique and it's 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 Monotolic Scheduling (RMS). We provide how-to information for RMS, based on Rate Monotonic Analysis (RMA).

August 21, 2020
status: preliminary draft
version: 0

Embedded Systems INSTITUTE

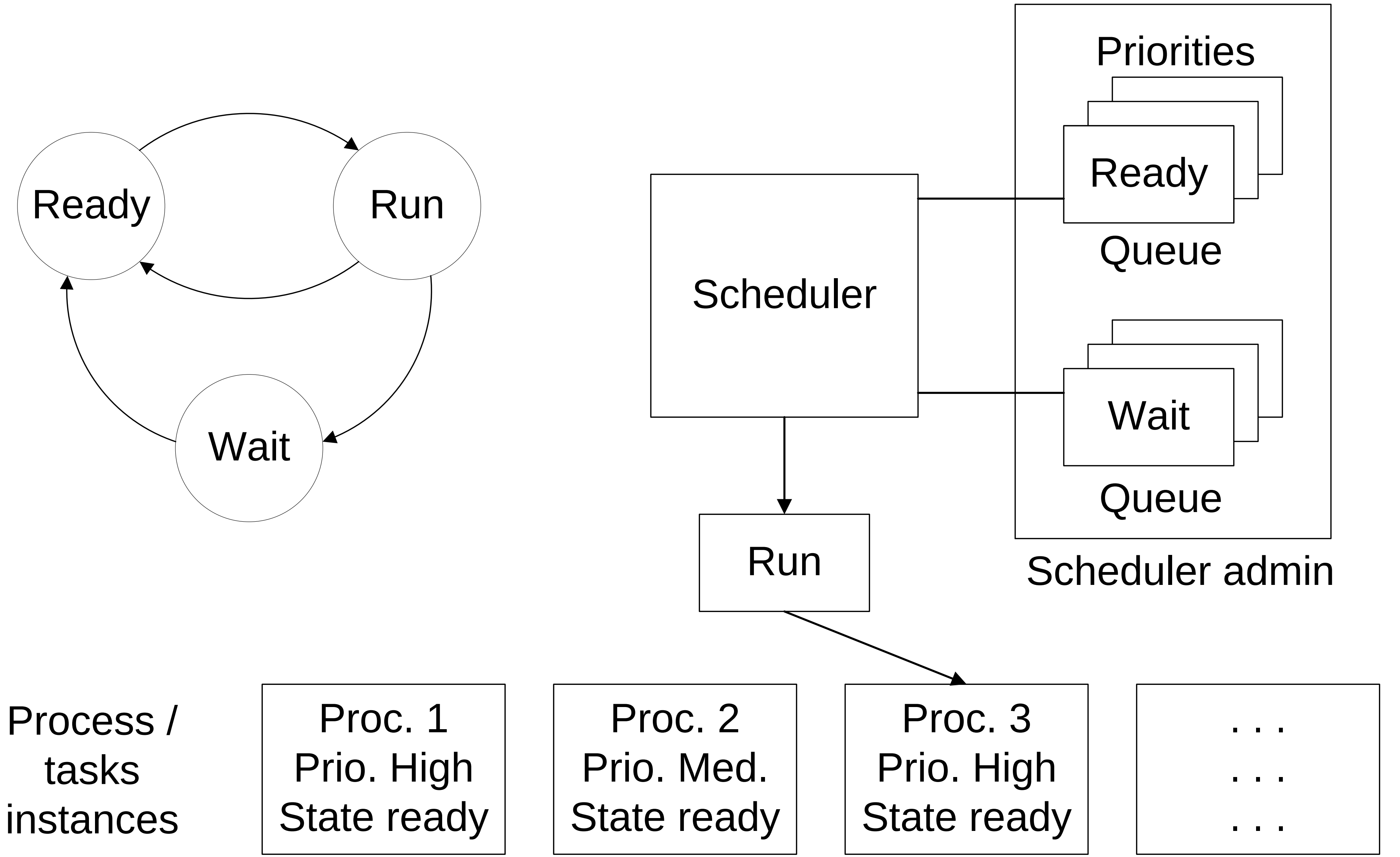
Theory Block Scheduling Techniques and Analysis

Theory Hard Real Time Scheduling

Earliest Deadline First (EDF)

Rate Monotonic Scheduling (RMS)

Real Time Scheduling



Earliest Deadline First

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

EDF = Earliest Deadline First

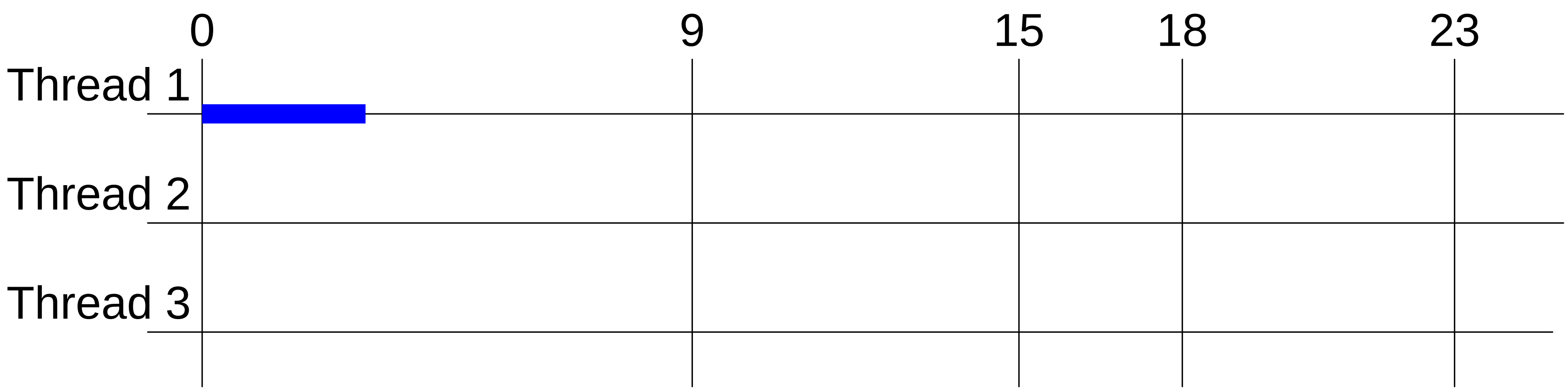
Earliest Deadline based scheduling
for (a-)periodic Processing

The theoretical limit for any number of processes
is 100% and so the system is schedulable.

Calculate loads and determine thread activity (EDF)

Thread	Period = deadline	Processing	Load
Thread 1	9	3	33.3%
Thread 2	15	5	
Thread 3	23	5	

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



Source: Ton Kostelijk - EXARCH course

Rate Monotonic Scheduling

• Determine deadlines (period)	in terms of Frequency or Period (1/F)
• Assign priorities	Highest frequency (shortest period) ==> Highest priority
• Constraints	Independent activities Periodic Constant CPU cycle consumption Assumes Pre-emptive scheduling

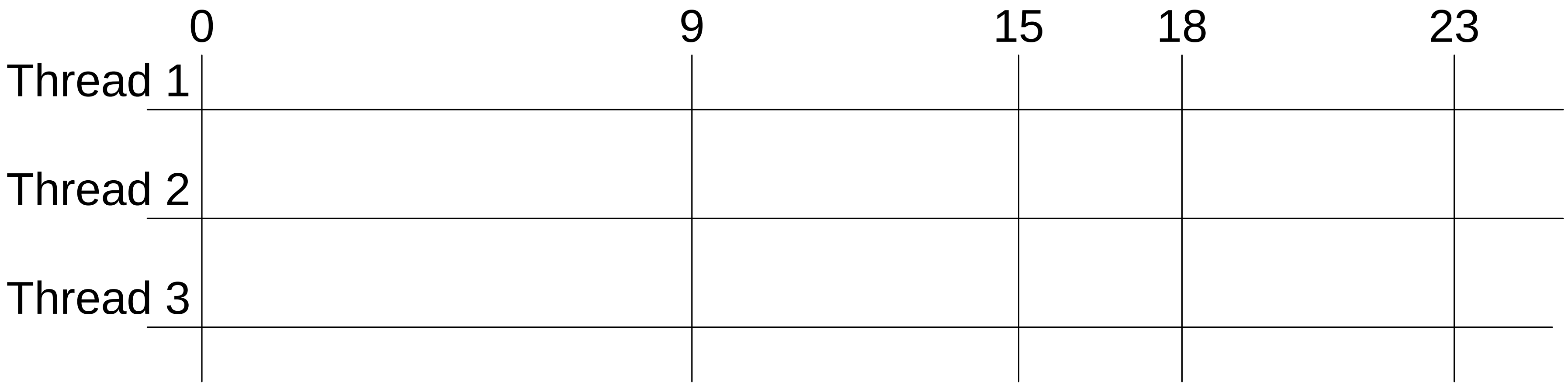
RMS = Rate Monotonic Scheduling

Priority based scheduling for Periodic Processing of tasks with a guaranteed CPU - load

Calculate loads and determine thread activity (RMS)

Thread	Period = deadline	Processing	Load
Thread 1	9	3	33.3%
Thread 2	15	5	
Thread 3	23	5	

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:

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

- 1.00 , 0.83 , 0.78 , 0.76 , ... log(2) = 0.69

Source: Ton Kostelijk - EXARCH course

RMS Evaluation

- RMS cannot utilize 100% (1.0) of CPU, but for 1, 2, 3, 4, ... ∞ 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

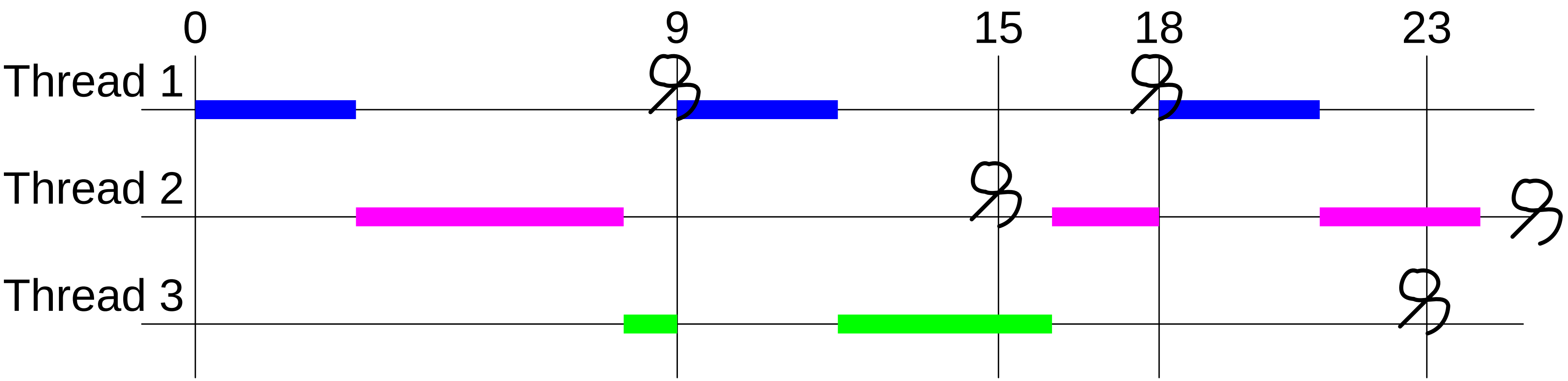
RMS Evaluation (continued)

- 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)

Thread	Period = deadline	Processing	Load
Thread 1	9	3	33.3%
Thread 2	15	5	33.3%
Thread 3	23	5	21.7%
			88.3%

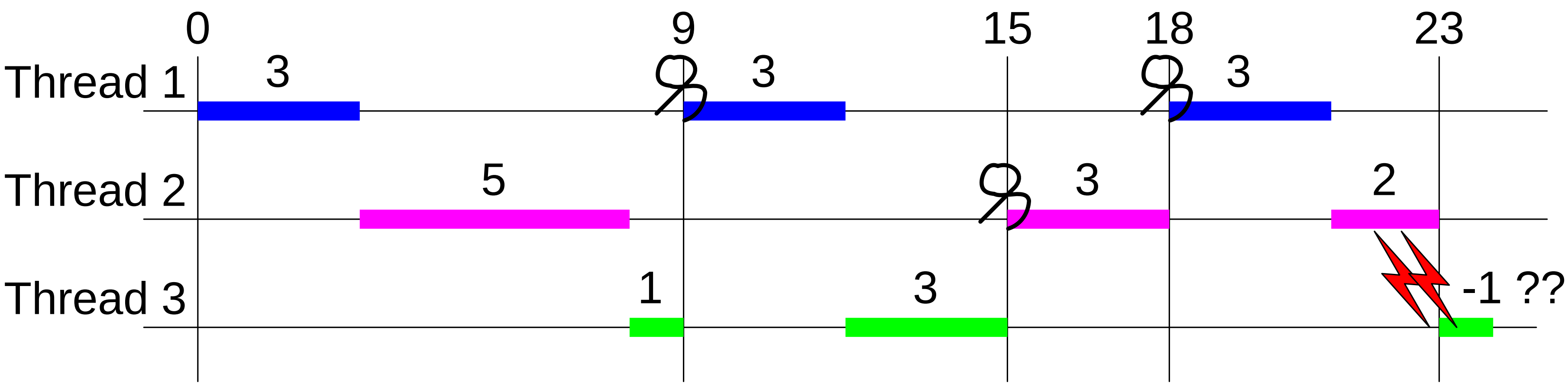


Source: Ton Kostelijk - EXARCH course

Answer RMS Exercise

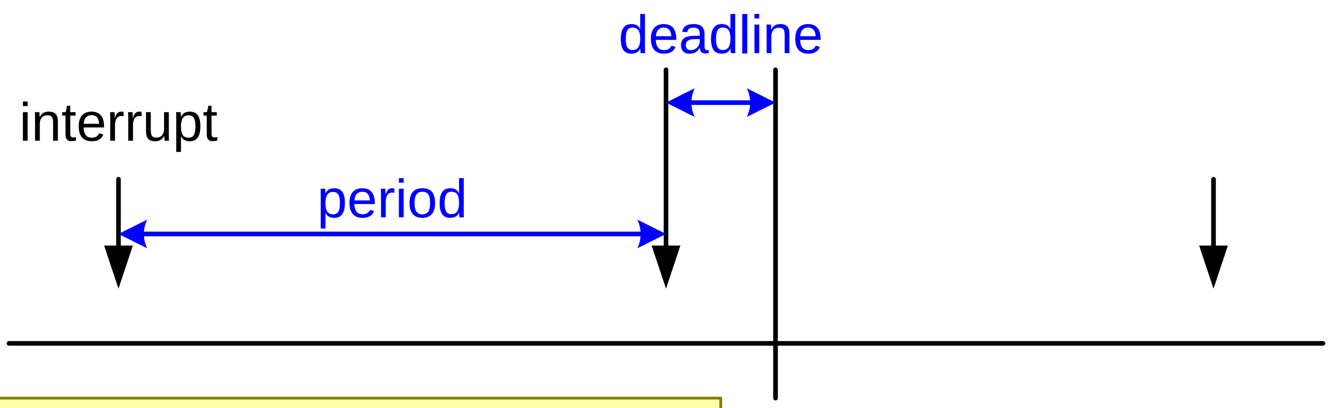
Answers: loads and thread activity (RMS)

Thread	Period = deadline	Processing	Load
Thread 1	9	3	33.3%
Thread 2	15	5	33.3%
Thread 3	23	5	21.7%
			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

Colofon

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*.