Real-time Scheduling
Periodic tasks

Tullio Facchinetti
<tullio.facchinetti@unipv.it>

24 marzo 2023

http://robot.unipv.it/toolleeo

Why periodic scheduling?

several computing tasks are inherently periodic

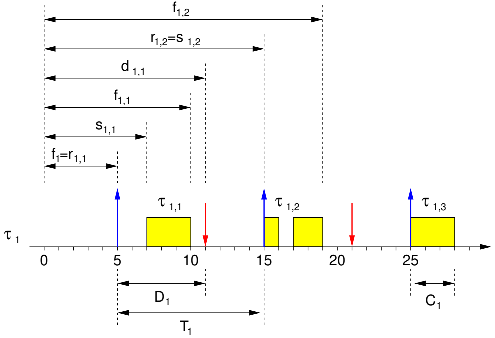
- sensory acquisition
- actuators driving
- control loops
- operation planning
- data visualization

Examples (from small UAV application):

Task	Period [ms]	Task	Period [ms]
GPS	1000.0	Power check	500.0
Inclinometer	200.0	Servo control	20.0
Temperature	1000.0	Control loop	12.5
Accelerometer	12.5	Communication	100.0
Gyroscopes	12.5		

Model of periodic tasks

a task is defined as $\tau_i = (C_i, D_i, T_i)$
the phase is (usually) neglected



param.	meaning
τ_i	i-th periodic task
$\tau_{i,j}$	j-th instance (job) of τ_i
Φ_i	phase of task τ_i
T_i	period of task τ_i
D_i	relative deadline of task τ_i
C_i	computation time of task τ_i
$r_{i,j}$	release time of $\tau_{i,j}$
$s_{i,j}$	start time of task $\tau_{i,j}$
$f_{i,j}$	finishing time of task $\tau_{i,j}$
$d_{i,j}$	absolute deadline associated with job $\tau_{i,j}$ ($d_{i,j} = r_{i,j} + D_i$)

$$\tau_1 = (3,\ 6,\ 10)$$

Assumptions

- all instances of a task (job) have the same WCET (duration)
- all jobs have the same relative deadline, which is assumed to be equal to the period, i.e. $D_i = T_i$ (**implicit deadlines**)
- tasks are independent: no precedence constraints or shared resources
- tasks are not self-suspending
- full preemption
- the kernel overhead is neglected

Utilization

$$U_i = \frac{C_i}{T_i}$$

$$U = \sum_{i=1}^{n} U_i = \sum_{i=1}^{n} \frac{C_i}{T_i}$$

- **fraction of time used by the processor** to execute the periodic tasks
- it does not depend from the scheduling algorithm, but only on tasks parameters

it must hold → $U \leq 1$

i.e., processor load $\leq 100\%$

Least Upper Bound of U

Given a scheduling algorithm A

Least Upper Bound of U

$$U_{lub}(A)$$

represents	The highest value of U such that every task set is schedulable by A
in other terms	Every task set, such that $U \leq U_{lub}(A)$, is schedulable by A

Least Upper Bound of U

Caution!

What happens if a task set has $U > U_{lub}$?

FALSE the task is not schedulable

TRUE the schedulability test based on U_{lub} can not tell whether the task set is schedulable or not

In other words: the test is sufficient but not necessary

there exist schedulable task sets having $U > U_{lub}$

- in any case, it must hold $U \leq 1$
- in fact, if $U > 1$ no algorithm can generate a feasible schedule

Static/dynamic priorities

Static priority algorithm

- priorities are assigned on the basis of fixed parameters
- can be assigned to tasks before their activation

Dynamic priority algorithm

- priorities are assigned on the basis of parameters that change value during the system running

Optimality

an algorithm is said optimal
if it minimizes some cost function
defined on the generated schedule

from the schedulability viewpoint:

an optimal algorithm
always finds a feasible schedule if one exists

Considered scheduling algorithms: fixed priority

Rate Monotonic

Priorities assigned inversely proportional to the period

- shorter period $\Rightarrow$ higher priority
- larger period $\Rightarrow$ lower priority

RM is optimal in the class of
fixed priority algorithms

Considered scheduling algorithms: dynamic priority

Earliest Deadline First

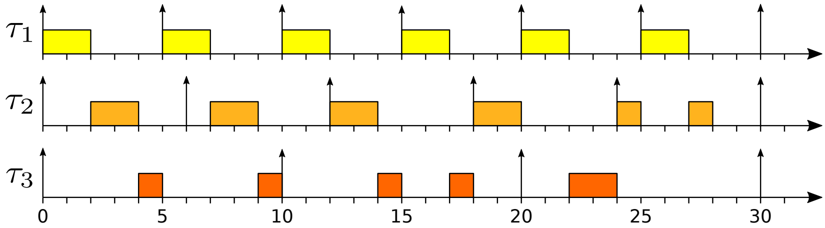
Priorities assigned inversely proportional to the absolute deadline

- closer absolute deadline $\Rightarrow$ higher priority
- farter absolute deadline $\Rightarrow$ lower priority

EDF is optimal in the class of
dynamic priority algorithms

Rate Monotonic (RM)

the priority of a task τ_i is inversely proportional to its period T_i

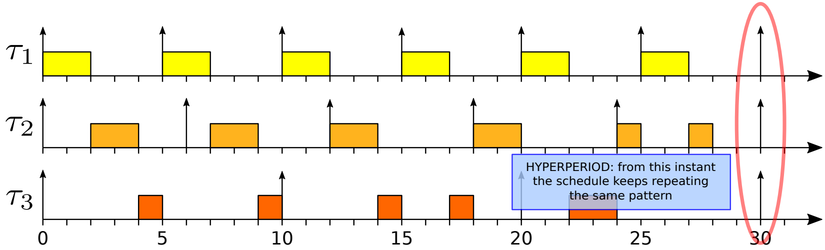


$\tau_1 = (2, 5)$	$\tau_2 = (2, 6)$	$\tau_3 = (2, 10)$
$U_1 = 2/5 = 0.4$	$U_2 = 2/6 = 0.333$	$U_3 = 2/10 = 0.2$
shortest period ($T_1 = 5$)	medium period ($T_2 = 6$)	largest period ($T_3 = 10$)
highest priority	medium priority	lowest priority

$$U = 2/5 + 1/3 + 1/5 = (6 + 5 + 3)/15 = 14/15$$

Rate Monotonic (RM)

the priority of a task τ_i is inversely proportional to its period T_i

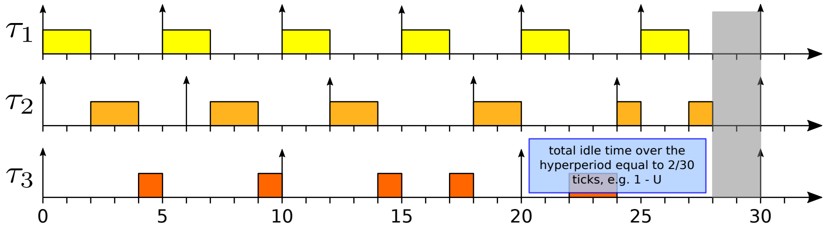


$\tau_1 = (2, 5)$	$\tau_2 = (2, 6)$	$\tau_3 = (2, 10)$
$U_1 = 2/5 = 0.4$	$U_2 = 2/6 = 0.333$	$U_3 = 2/10 = 0.2$
shortest period	medium period	largest period
$(T_1 = 5)$	$(T_2 = 6)$	$(T_3 = 10)$
highest priority	medium priority	lowest priority

$$U = 2/5 + 1/3 + 1/5 = (6 + 5 + 3)/15 = 14/15$$

Rate Monotonic (RM)

the priority of a task τ_i is inversely proportional to its period T_i

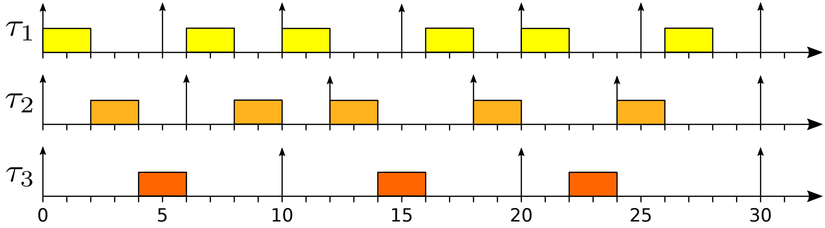


$\tau_1 = (2, 5)$	$\tau_2 = (2, 6)$	$\tau_3 = (2, 10)$
$U_1 = 2/5 = 0.4$	$U_2 = 2/6 = 0.333$	$U_3 = 2/10 = 0.2$
shortest period ($T_1 = 5$)	medium period ($T_2 = 6$)	largest period ($T_3 = 10$)
highest priority	medium priority	lowest priority

$$U = 2/5 + 1/3 + 1/5 = (6 + 5 + 3)/15 = 14/15$$

Earliest Deadline First (EDF)

the priority of a job $\tau_{i,j}$ is inversely proportional to its absolute deadline $d_{i,j}$



$\tau_1 = (2, 5)$ $\tau_2 = (2, 6)$ $\tau_3 = (2, 10)$

Schedulability test when $D_i = T_i$

Two different schedulability test available for RM

"historical" test:

$$U_{lub}(n) = n(2^{1/n} - 1)$$

C. L. Liu and James W. Layland, "Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment", Journal ACM 20, 1, pp. 46-61, **1973**.

more recent and accurate:

$$\prod_{i=1}^{n}(U_i + 1) \leq 2$$

E. Bini, G. C. Buttazzo, G. M. Buttazzo, "Rate Monotonic Analysis: the Hyperbolic Bound", IEEE Transactions on Computers 52 (7), pp. 933-942, **2003**.

Notes on the historical test

LL test for 2 tasks:

$$U_{\mathbf{lub}}(2) = 2(\sqrt{2} - 1) \simeq 0.8284$$

two tasks are schedulable if their total utilization $U < 0.8284$

For every task set:

$$\lim_{n \to \infty} U_{\mathbf{lub}}(n) \simeq 0.69$$

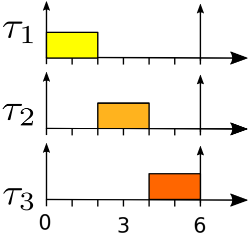
every task set is schedulable by RM if $U < 0.69$

Proof that the schedulability test is only sufficient

There exist schedulable task sets having $U > U_{lub}$

Simple example:

- 3 periodic tasks τ_1, τ_2, τ_3
- $\tau_1 = (2, 6)$, $\tau_2 = (2, 6)$, $\tau_3 = (2, 6)$
- $U = 1/3 + 1/3 + 1/3 = 1$
- $U_{lub}(3) = 3(2^{1/3} - 1) \simeq 0.7798$
- although $U > U_{lub}$ $(1 > 0.7798)$, the task set is trivially schedulable by RM



Schedulability test of EDF when $D_i = T_i$

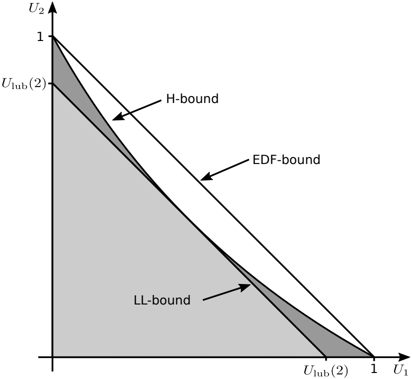
Schedulability test for EDF

$$U_{lub}(n) = 1$$

- independent from the number of tasks n
- can fully utilize the processor (up to 100%)

Comparison of schedulability bounds

comparison among EDF-, hyperbolic (H-) and Liu and Layland (LL-) bounds



LL: $U_{\mathbf{lub}}(2) = 2(\sqrt{2} - 1) = 0.8284$

... what if $D_i \neq T_i$?

In case of fixed priority, the **Deadline Monotonic (DM)** is used

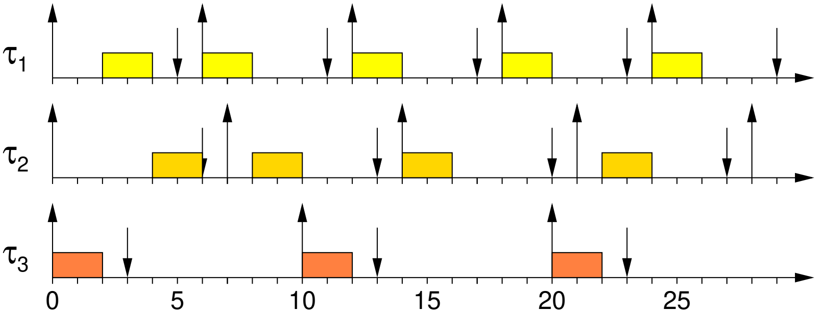
- the priority is inversely proportional to the relative deadline D_i of task τ_i
- Deadline Monotonic is optimal in the class of fixed priority algorithms

In case of dynamic priorities, **EDF** is used

schedulability tests are different for both algorithms w.r.t. to implicit deadlines

Deadline Monotonic: example

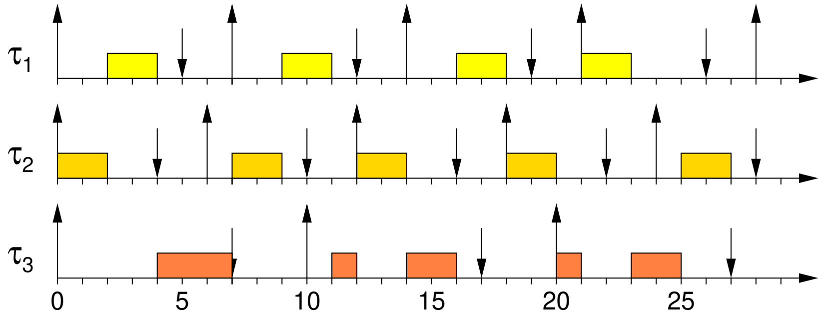
the priority of a task τ_i is inversely proportional to its relative deadline D_i



$\tau_1 = (2, 5, 6)$ $\tau_2 = (2, 6, 7)$ $\tau_3 = (2, 3, 10)$

EDF with $D_i \neq T_i$: example

the priority of a job $\tau_{i,j}$ is inversely proportional to its absolute deadline $d_{i,j}$



$\tau_1 = (2, 5, 7)$ $\tau_2 = (2, 4, 6)$ $\tau_3 = (3, 7, 10)$