On-line scheduling of periodic tasks in RT OS

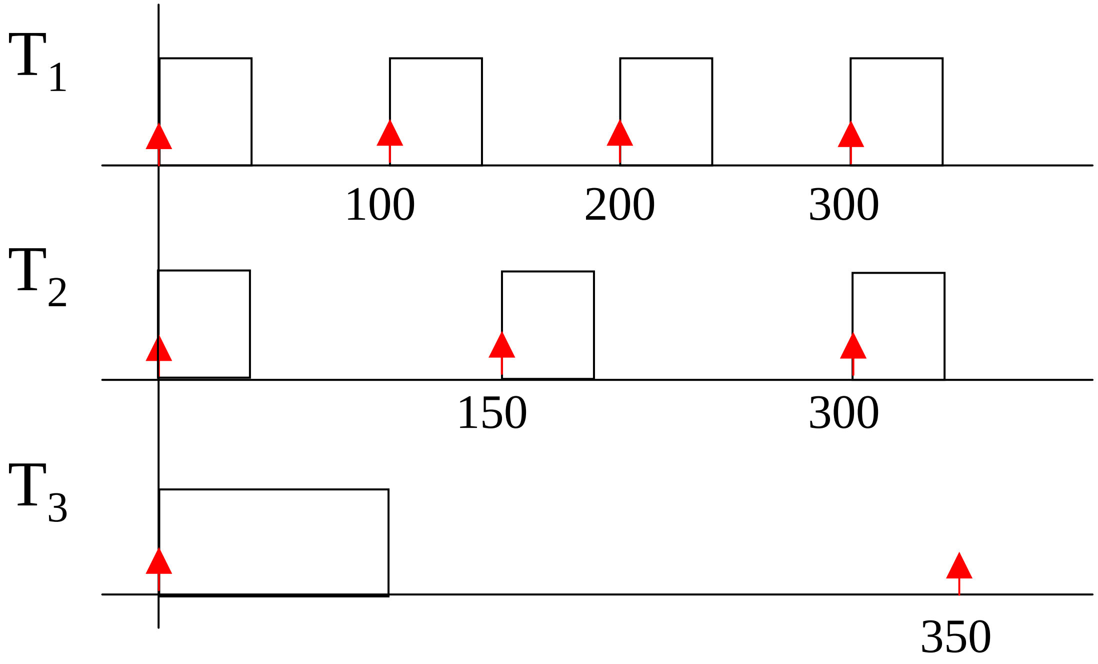
Even if RT OS is used, it is needed to set up the **task priority.**

The scheduling problem is solved on two levels:

- fixed priority assignment … by RMS
- dynamic scheduling … by priority based preemptive RT OS

Typical application

- RT $\Rightarrow$ deadline … it is assumed to be equal to the release time of the next period
- periodic tasks (aperiodic tasks are scheduled using so called servers)
- Example 1:



Task T_1:

processing time…..p_1=40;

period……………..τ_1=100;

Task T_2:

p_2=40;

τ_2=150;

Task T_3:

p_3=100;

τ_3=350;

Rate Monotonic Scheduling (RMS)

Basic assumptions:

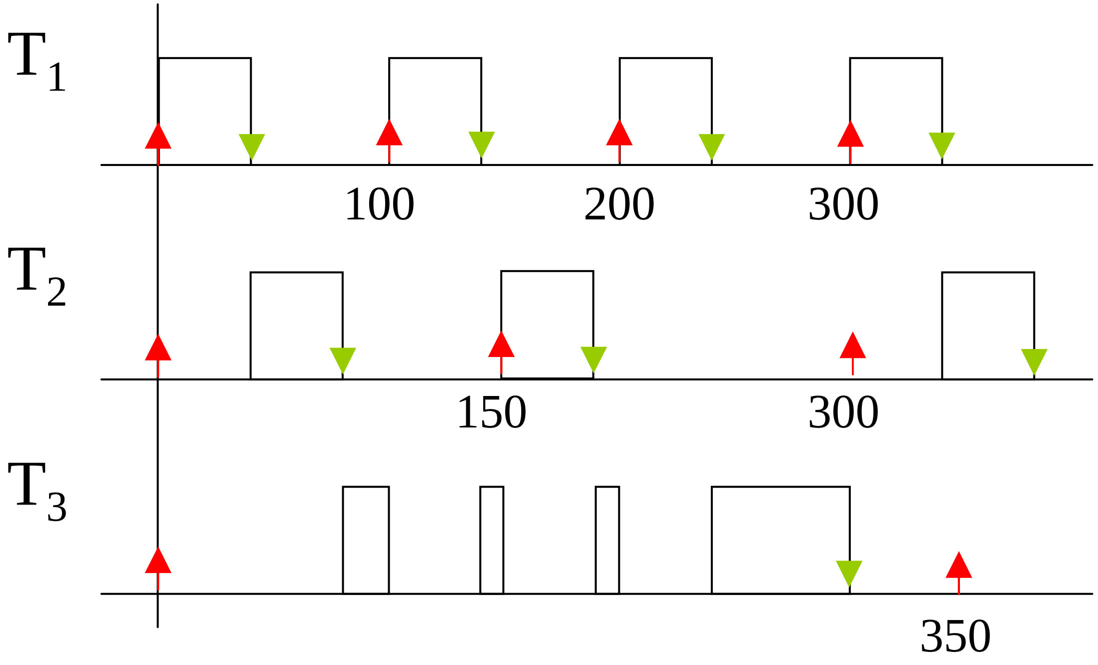
- tasks are executed by priority based **preemptive kernel**
- deadline is at the end of each period

RMS: Assign fixed priorities to the tasks according to their request rate (inverse to their period ~ deadline). **Highest priority** is assigned **to** the task with **highest frequency.**

Schedulability: ***n*** **periodic** and **independent** tasks are completed before their deadlines

Optimality: RMS is optimal among all **fixed** priority algorithms. There is no fixed priority algorithms able to schedule an application that is not schedulable by RMS

Solution to Example 1 using RMS



deadline of previous period ~ release time of the next period)

completion time

Processor utilization factor

Processor utilization factor ***U*** is fraction of processor time spent by execution of *n* tasks.

$$U = \sum_{i=1}^{n} \frac{p_i}{\tau_i}$$

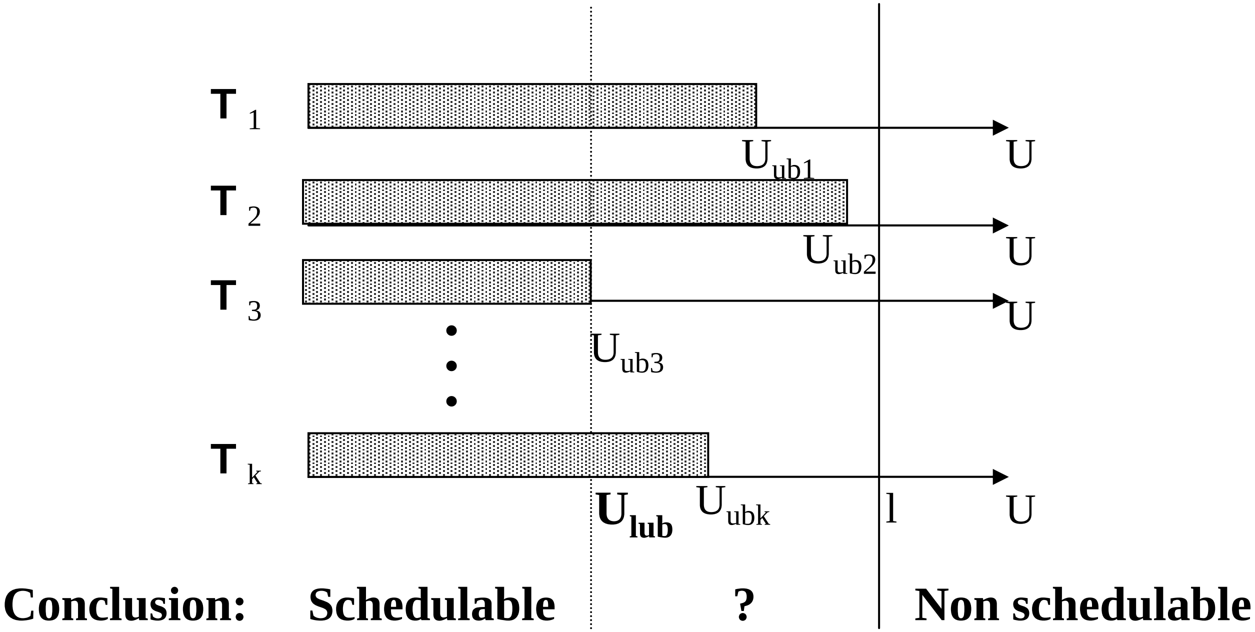
***Upper bound* of utilization factor $U_{ub}(\mathrm{T},A)$:**

• is the maximum value of *U* below which is the task set

T schedulable by algorithm *A*.

Least upper bound $U_{lub}(A)$ of utilization factor, is the minimum of utilization factor over all task sets that fully utilize processor:

$$U_{lub}(A) = min_{\mathbf{T}}\ U_{ub}(\mathbf{T},A)$$



„Utilization bound theorem“ for RMS

Sufficient condition:

Any set T of *n* independent periodic tasks is schedulable by RMS if:

$$U(n) \leq n(2^{1/n}-1)$$

n	1	2	3	4	5	6	7	8	9	10
U_{lub}	1,00	0,828	0,780	0,757	0,743	0,735	0,729	0,724	0,721	0,718

$$U_{lub}(RMS) = \lim_{n \to \infty} n(2^{1/n}-1) = \ln 2 = \mathbf{0.69}$$

Utilization bound theorem – **<u>is pesimistic</u>**

In case of example 1:

Task T_1: p_1=40; τ_1=100; $\Rightarrow$ 0.4
Task T_2: p_2=40; τ_2=150; $\Rightarrow$ 0.267
Task T_3: p_3=100; τ_3=350; $\Rightarrow$ 0.286

U = 0.4 + 0.267 + 0.28 = 0.953
exceeds *Utilization bound theorem* since

$$0.953 \not\leq 3(2^{1/3}-1) = 0.780$$

T_3 has lower priority than T_1 and T_2 (since there are no inter-task communications, T_3 cannot influence execution of T_1 and T_2), we can try "Utilization bound theorem for 2 tasks:

T_1 and T_2 do not **exceed** since: $0.667 \leq 2(2^{1/2}-1)=0.828$

To test schedulability of T_3 we will use „Completion time theorem"

Completion time theorem

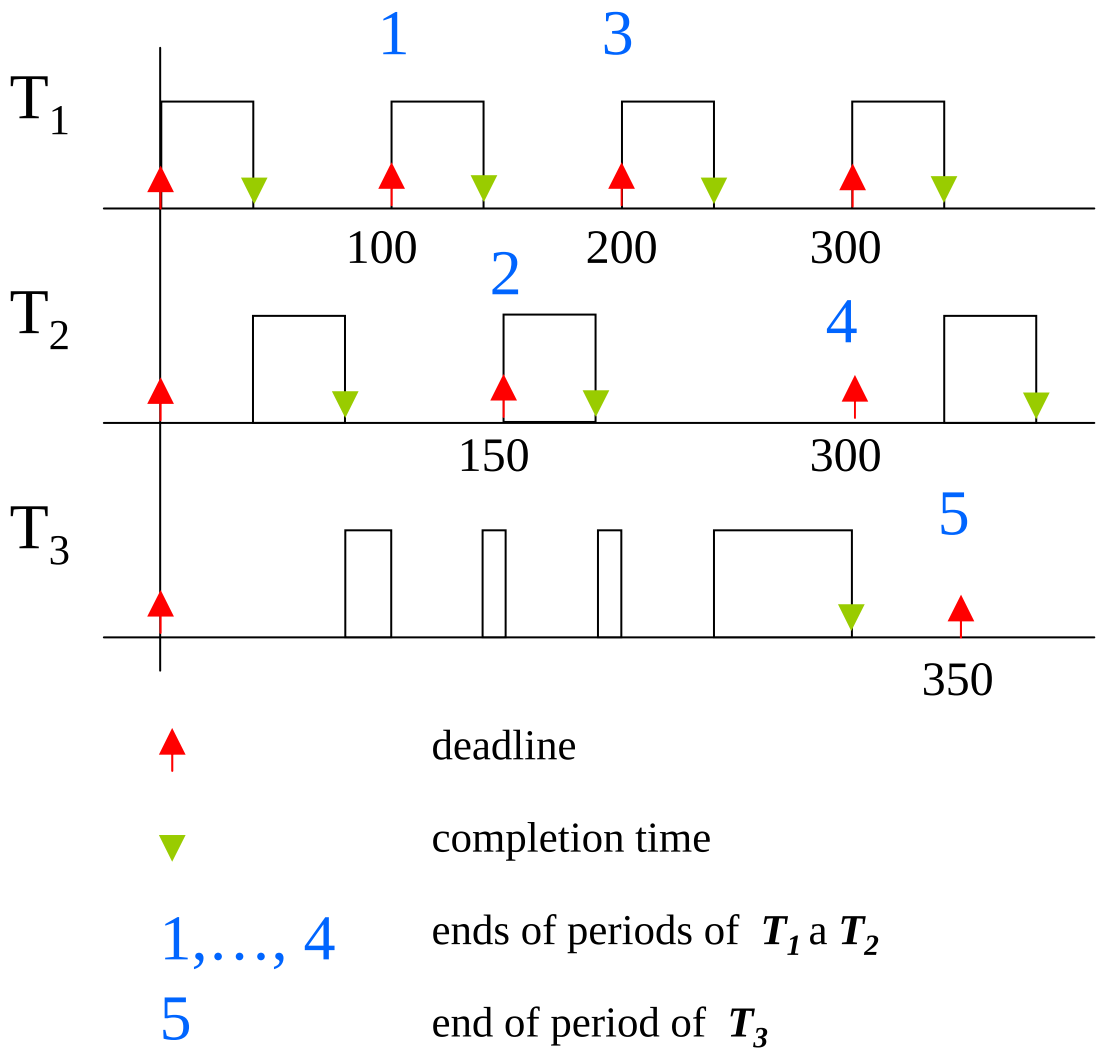
•**necessary and sufficient** condition for the set of independent periodic tasks using RMS

•worst case situation – **all tasks start at the same time** (worst phasing). ⇒ It is sufficient to examine one period of given task T_x .

•set of examined dates consists of the **end of $\boldsymbol{T_x}$ period** and **each end of higher task periods**.

•at each examined date we check whether **all tasks have been compeeted as often as they have been released**

In the case of example 1:



• It is not needed to derive a Gantt chart for tis analysis.

• Task T_3 is schedulable iff **at least one** of the following conditions hold:

1	$p_1 + p_2 + p_3 \le \tau_1$	40+40+100>100	NO
2	$2p_1 + p_2 + p_3 \le \tau_2$	80+40+100>150	NO
3	$2p_1 + 2p_2 + p_3 \le 2\tau_1$	80+80+100>200	NO
4	$3p_1 + 2p_2 + p_3 \le 2\tau_2$	120+80+100=300	YES
5	$4p_1 + 3p_2 + p_3 \le \tau_3$	160+120+100>350	NO

$\Rightarrow$ task T_3 is schedulable since at the worst phasing it is **completed in time 300**

RMA Extension by inter-tasks communication

When tasks share resource with mutual access (critical section), the task, which is inside critical section can cause blocking of the higher priority task waiting to enter critical section.

When the blocking of tasks is bounded, we can compute the ***longest duration of task blocking*** $\boldsymbol{B_i}$ and take it into account in **generalized utilization bound theorem** and in **generalized completion time theorem**.

…very **pessimistic** result

RMA Extension by aperiodic tasks - Fixed priority servers

Fixed priority servers

Assumption: neither periodic nor aperiodic tasks can be released infinitely often to consume infinite amount of the processor time

Solutions:

• they can run on the background (lowest priority)

• using servers – the server (periodic task) is ready to use its **capacity** $\mathbf{p_S}$ within **period** τ_S

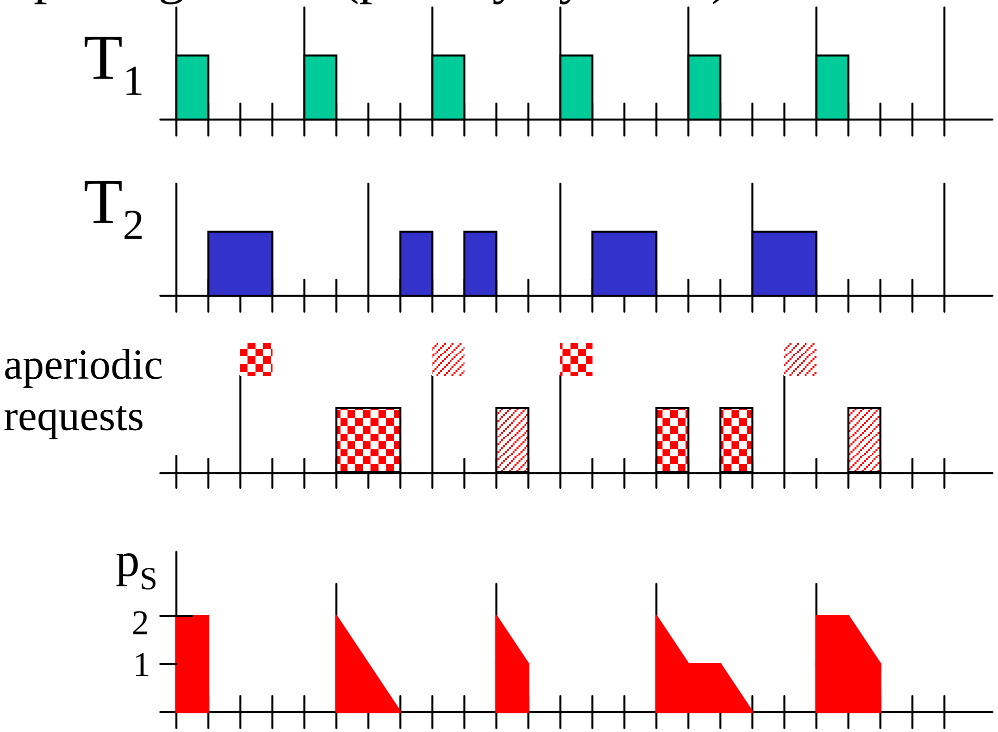
Polling server

• at the moment of its activation, the server **serves already released** aperiodic tasks using its **capacity p_S** .
• if there is no aperiodic task at this moment, then the server **capacity is erased** (as in the restaurant, when a waiter finds out that there is nobody requiring its service)

Example 2: two periodic tasks + polling server (priority by RMS)

	p_i	τ_i
T_1	1	4
T_2	2	6

	p_S	τ_S
server	2	5



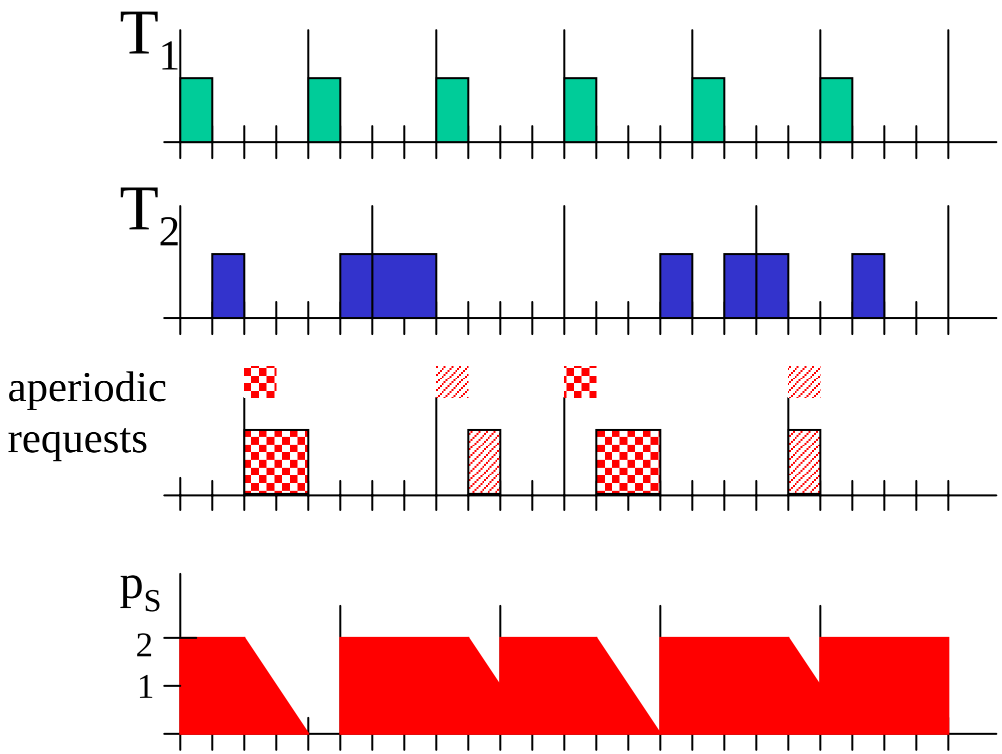
Defferable server

- the server serves aperiodic tasks using its capacity p_S
- unused **capacity is kept until the end of the period** (patient waiter)

Ex. 3: two periodic tasks + defferable server (priority by RMS)

	p_i	τ_i
T_1	1	4
T_2	2	6

	p_S	τ_S
server	2	5



RMS - conclusion

• Rate monotonic scheduling is good to specify the task priorities for preemptive kernels

• Good behavior in overload (unexpected prolongation of p_i) – lowest priority task deadline is exceeded first

• Extension by inter-task communication is too pessimistic

EDF in on-line scheduling

- EDF can be used as on-line scheduling rule (can be seen as dynamic priority assignment)
- **optimality**: since EDF does not make any specific assumption on the periodicity of the tasks, the optimality proven for aperiodic tasks also **holds for periodic tasks**

Theorem: **A set of periodic independent tasks is schedulable with EDF if and only if:**

$$U = \sum_{i=1}^{n} \frac{p_i}{\tau_i} \leq 1$$