Cyclic Executive Method

Problem (1):

A real-time system runs three tasks using the cyclic executive method. Tasks have periods of 30 ms, 40 ms, and 60 ms and execution times per period of 6, 8, and 10 ms respectively. The deadline of each task is equal to its period.

a) Find two suitable values for the minor cycle (frame) in the above system?
b) For one of the above two frame values **s**ketch possible execution schedule during the major cycle for the following two cases:
i) Time between consecutive executions of the first task should be equal.
ii) Execution of third task should start as early as possible after its period starts.

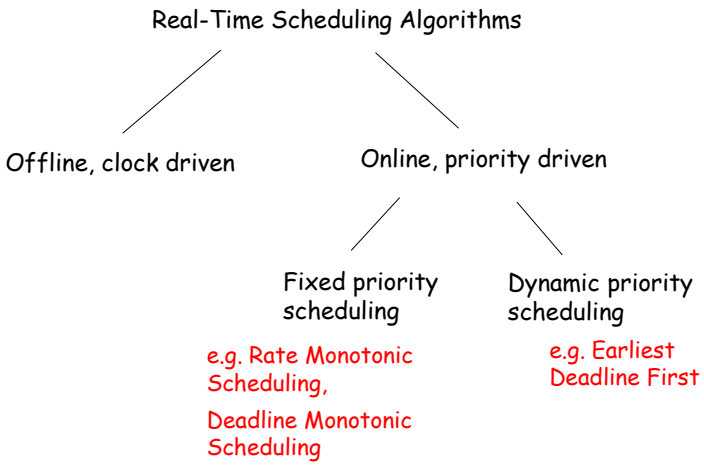
Cyclic Executive Method

Problem (2):

A real-time system using the cyclic executive method is required to run three tasks A, B, and C with periods of 30, 40, and 60 ms; and execution times per period of 5, 7, and 25 ms respectively. The deadline of each task is equal to its period.

a) Can a suitable minor cycle (frame) value and schedule be obtained for these tasks?
b) Repeat if task C can be split into two subtasks (coroutines) C1 (20 ms) and C2 (5 ms) that can be executed in different frames. Find an execution schedule if this is possible.

Real-time Scheduling



Rate Monotonic Scheduling

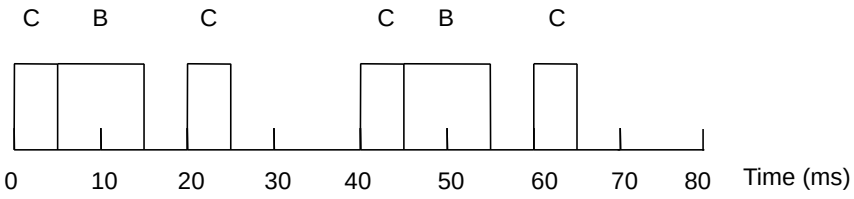
For a set of periodic tasks, the Rate Monotonic (RM) scheduling algorithm operates as follows:

- Each task is assigned a fixed priority: The shorter the period, the higher the priority.
- A task can be *pre-empted* by a request for a task with higher priority.

RM Scheduling is easy to implement even on simple hardware.

Rate Monotonic Scheduling

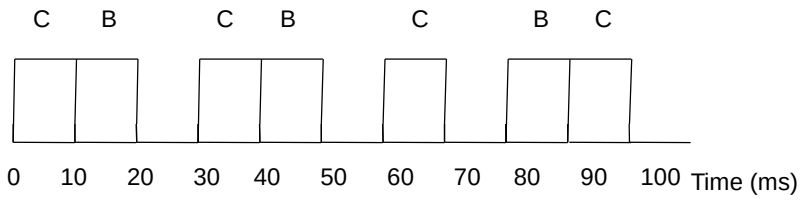
Example (1): Three tasks with periods of 80, 40, and 20 ms, and processing times of 40, 10, and 5 ms per period respectively. Deadline of each task is equal to its period.



These tasks are RM schedulable.

Rate Monotonic Scheduling

Example (2): Three tasks with periods of 50, 40, and 30 ms, and processing times of 12, 10, and 10 ms per period respectively. Deadline of each task is equal to its period.



A deadline of first task is missed. Tasks are not RM schedulable.

Rate Monotonic Scheduling

Thus, a task set may or may not be RM schedulable. The first basic results of RM schedulability were obtained by Liu and Layland in 1973. The following results apply for:

- Independent pre-emptable periodic tasks.
- For each task relative deadline=period.

Theorem (1): A sufficient condition for schedulability of n tasks using RM is :

$$\sum_{i=1}^{n} u_i \leq n\left(2^{1/n} - 1\right)$$

Rate Monotonic Scheduling

Theorem (2)- Critical Instant theorem: The worst response time for a task occurs when it is released simultaneously with the release of all tasks with higher priority.

This theorem leads to an exact schedulability test: If all tasks are started at the same time and each task meets its first deadline, then deadlines will always be met for any other combination of start times.

Theorem (3): No other fixed priority assignment can schedule a task set if RM priority assignment can't schedule it.

We say that RM is *optimal* among fixed priority algorithms.

Rate Monotonic Scheduling

Theorem (4): If the period of each task is an integer multiple of the periods of tasks with shorter periods, then the condition of RM schedulability becomes:

$$\sum_{i=1}^{n} u_i \leq 1$$

Later works found less restrictive schedulability tests based on utilization bounds, e.g. the hyperbolic bound (Bini et al 2003).

Earliest Deadline First (EDF)

- Higher priority is given to the task with the *nearest absolute deadline*.

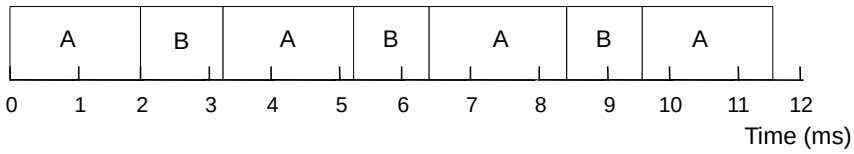
  On contrast with the RM algorithm, the EDF algorithm uses *dynamic task priorities*, i.e. a given task can have different priority levels at different times.

- Like the RM algorithm, a task can be *preempted* by a request for a task with higher priority (i.e. earlier deadline.)

Earliest Deadline First (EDF)

Example (3): Two tasks with periods of 3, and 4 ms, and processing times of 2, and 1.2 ms per period respectively. Deadline of each task is equal to its period.

These two tasks are not schedulable using the RM algorithm or *any* algorithm with fixed task priorities. However, EDF satisfy deadlines when two tasks start together.



Earliest Deadline First (EDF)

The following result was also proved by Liu and Layland (1973) under the same set of assumptions.

Theorem (5): EDF can be used to schedule any set of periodic tasks with:

$$\sum_{i=1}^{n} u_i \leq 1$$

Thus, by using dynamic priorities, EDF has a clear advantage of being able to schedule sets of tasks that would not be schedulable using the RM algorithm.

Earliest Deadline First (EDF)

- EDF is harder to implement and has higher CPU overheads.
- The same algorithm can be used with aperiodic tasks, but deadlines cannot be guaranteed under overloads (not suitable in a hard RT system)

Example (4): Three periodic tasks with periods of 100, 120, and 150 ms, and processing times of 20, 55, and 45 ms per period.

Sum of utilizations is 0.958, and tasks are not schedulable using RM. However, these three tasks are schedulable using EDF.

Response Time Analysis

An alternative test of schedulability is provided by Response Time Analysis, which can readily be extended to more general tasks models.

The worst-case response time R_{imax} of task i (here R_i for short) is the longest time that may be needed to finish its execution after its release time. We can write:

$$R_i = p_i + I_i$$

where p_i is the WCET of the task and I_i is the worst-case interference from other tasks. Test of schedulability reduces to

$$R_i \leq d_i \qquad \forall i$$

where d_i is the deadline of task i.

Response Time Analysis

For fixed priority preemptive scheduling of periodic tasks, worst-case interference occurs when task is released at $t = 0$ together with all higher priority tasks (critical instant theorem).

A task j of higher priority than task i will be released within the interval $[0, R_i]$ for $\lceil R_i/T_j \rceil$ times, each time pre-empting task i for a duration of p_j.

Let $hp(i)$ be the set of all tasks with higher priority than i. Then:

$$R_i = p_i + \sum_{j \in hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil p_j$$

Response Time Analysis

This equation can be solved iteratively

$$R_i^{k+1} = p_i + \sum_{j \in hp(i)} \left\lceil \frac{R_i^k}{T_j} \right\rceil p_j$$

Starting from initial guess $R_i^0 = 0$ or p_i , estimate will monotonically increase until it converges to a solution $\left(R_i^{k+1} = R_i^k\right)$ or until R_i^{k+1} exceeds d_i.

Note that this analysis can be used even if $d_i \leq T_i$, and may account for higher priority sporadic tasks, replacing the period by the minimum inter-arrival time.

Response Time Analysis

Example (5):

Task i	p_i	T_i	d_i
A	3	8	8
B	4	14	14
C	5	22	22

If fixed priorities are selected according to RM.

$$R_A = p_A = 3 < d_A$$

$$R_B = 4 + \left\lceil \frac{R_B}{8} \right\rceil \times 3$$ Which converges to 7 < d_B.

$$R_C = 5 + \left\lceil \frac{R_C}{8} \right\rceil \times 3 + \left\lceil \frac{R_C}{14} \right\rceil \times 4$$ Which converges to 22 = d_C.

Response Time Analysis

Example (6):

Task i	p_i	T_i	d_i
A	3	11	11
B	4	14	7
C	3	19	6
D	2	20	19

Using rate-monotonic priority assignment (A-B-C-D)

$$R_A = p_A = 3 < d_A$$

$$R_B = 4 + \left\lceil \frac{R_B}{11} \right\rceil \times 3$$ Which converges to 7 = d_B.

$$R_C = 3 + \left\lceil \frac{R_C}{11} \right\rceil \times 3 + \left\lceil \frac{R_C}{14} \right\rceil \times 4$$ Which converges to 10 > d_C.

RM fails.

Response Time Analysis

For a *deadline monotonic* priority assignment (C-B-A-D)

$R_C = p_C = 3 < d_C$

$R_B = 4 + \left\lceil \frac{R_B}{19} \right\rceil \times 3$ Which converges to 7 = d_B.

$R_A = 3 + \left\lceil \frac{R_A}{19} \right\rceil \times 3 + \left\lceil \frac{R_A}{14} \right\rceil \times 4$ Which converges to 10 < d_A.

$R_D = 2 + \left\lceil \frac{R_D}{19} \right\rceil \times 3 + \left\lceil \frac{R_D}{14} \right\rceil \times 4 + \left\lceil \frac{R_D}{11} \right\rceil \times 3$

Which converges to 19 = d_D.

It can be shown that deadline-monotonic is the optimal fixed priority assignment if deadline ≤ period for all tasks.