Real Time Systems and Control Applications

A Brief Review

- **Given:**
  - **A set of real-time tasks**
  - **A scheduling algorithm**
- **Is the task set schedulable?**
  - **Yes → all deadlines met, forever**
  - **No → at some point a deadline might be missed**

- **Ways to schedule**
  - **Without priority (e.g. cyclic executive)**
  - **Static priorities**
  - **Dynamic priorities**
    - **Priorities is computed at runtime**
    - **More flexible, but less predictable**

Static Priority Assignment

- **A higher priority task is executed first than a lower priority task.**

- **Priority based on Criticality?**
  - **If we consider hard real time applications, they are of the same level of importance.**

- **Shorter period tasks get higher priority**

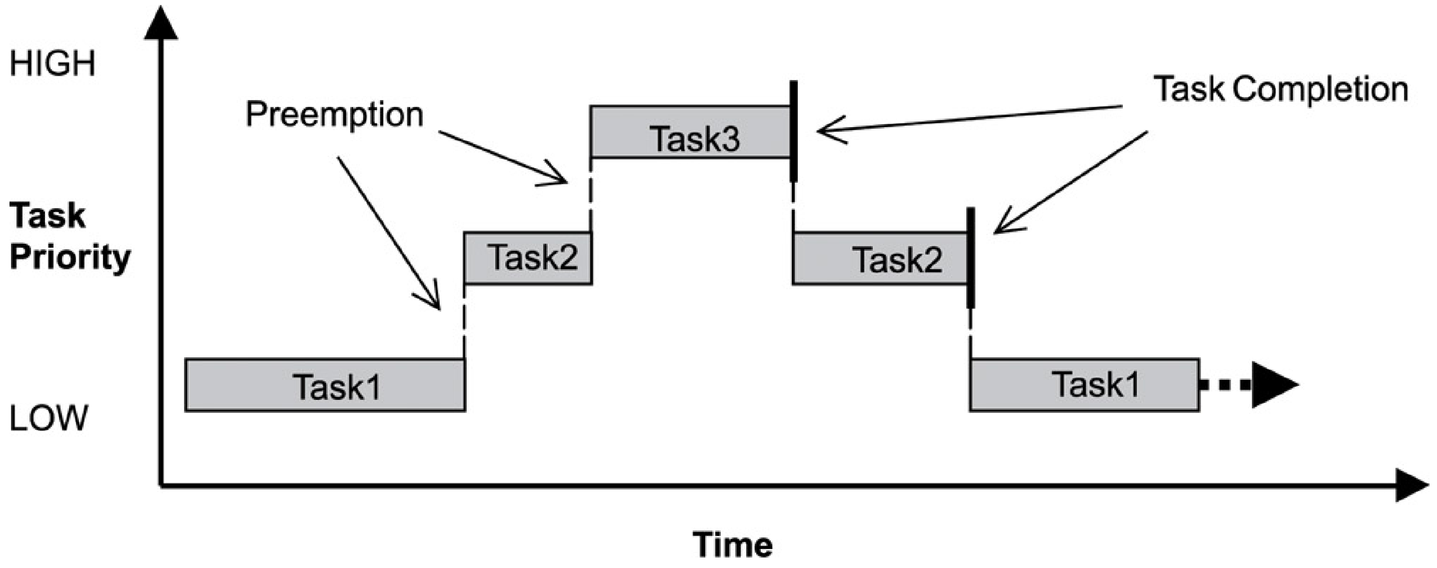
  → Rate monotonic (RM)
- **Tasks with shorter relative deadlines get higher priority**

  → Deadline monotonic (DM)

- **Both RM and DM…**
  - **Have good theoretical properties**
  - **Work well in practice**

RM Assumptions

- Tasks are running on uniprocessor system
- Tasks are preemptive
- There is no OS overhead for preemption.



Rate Monotonic (RM) Scheduling Algorithm

- RM is a static-priority approach, and one of the most popular algorithms.
- At any time instant, an RM scheduler executes the instance of the ready task that has the highest priority.
- The priority of a task is inversely related to its period (i.e. if task Ti has a period pi, and another task Tj has a period pj, then Ti has a higher priority than Tj, if pi < pj.
  - If two or more tasks have the same period, the scheduler selects one of these jobs at random.

Example

- You are given a set of periodic tasks:

  T1: 0, 5, 10, 15, 20, …

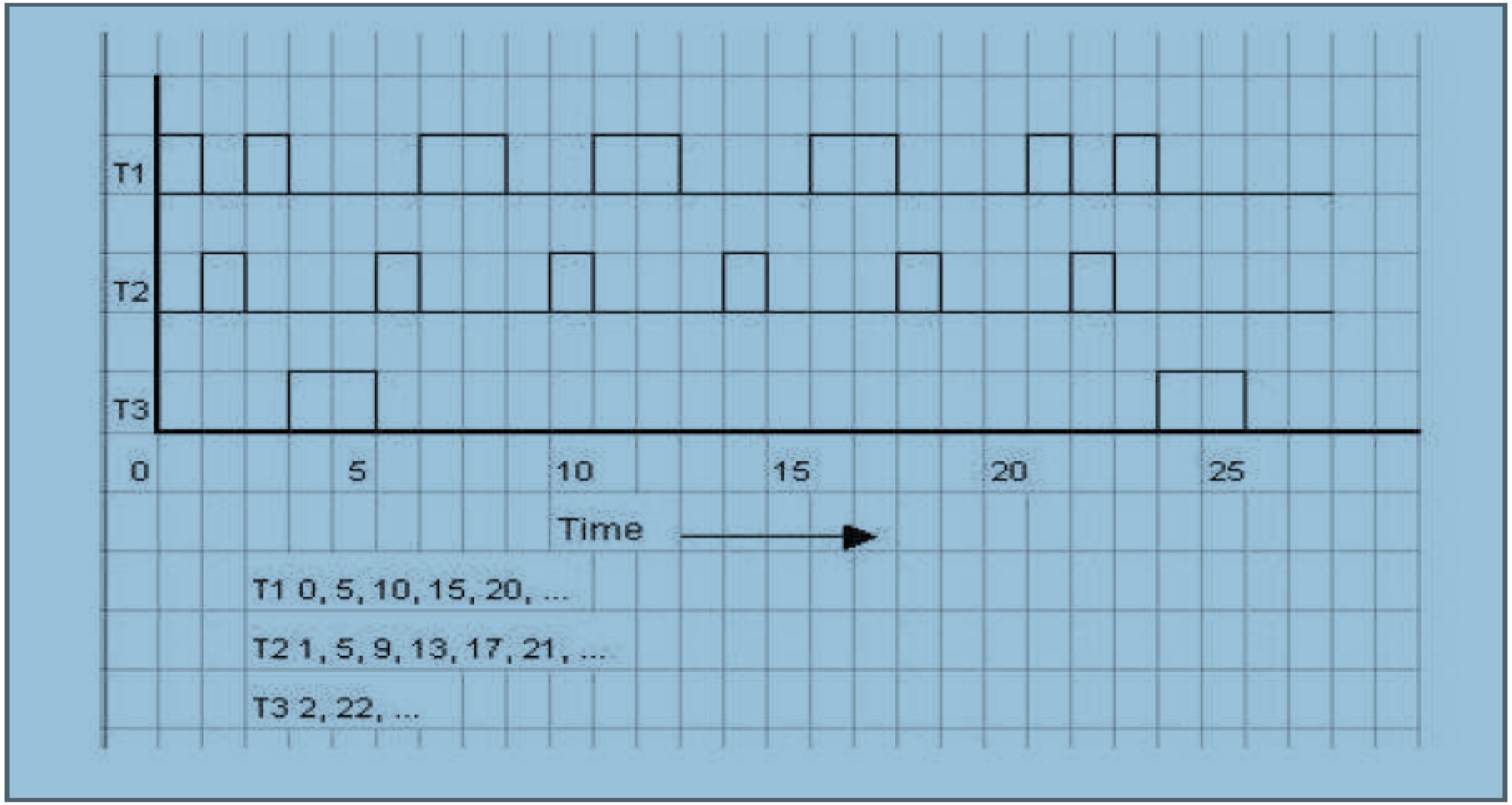
  T2: 1, 5, 9, 13, 17,…

  T3: 2, 22, 42, 62, …

- Here, we use a 4-tuple expression to represent the tasks:

  T1(0; 5; 2; 5), T2(1; 4; 1; 4), T3(2; 20; 2; 20)

Question: Can RM make the tasks meet their deadlines?



T1 0, 5, 10, 15, 20, ...

T2 1, 5, 9, 13, 17, 21, ...

T3 2, 22, ...

Difference between CE and RM

- T1: (2,1) and T2:(3,1)

Think what happens if an instance (say the second instance) of T1 is delayed for 1 time unit?

- With CE (f=1):

T1 T2 T1 T2 T1 I (no late arrival)

T1 T2 ? T2 T1 I (deadline missed when the 2nd instance of T1 is late!)

- With RM:

T1 T2 T1 T2 T1 I (no late arrival)

T1 T2 I T1 T1 T2 (all deadlines met when the 2nd instance of T1 is late!)

Schedulability Test

- Determining if a specific set of tasks satisfying certain criteria can be successfully scheduled (completing execution of every task by its specified deadline) using a specific scheduler.

- This test is often done at compile time, before the computer system and its tasks start execution.

Optimal Scheduler

- Optimal scheduler is one which may fail to meet the deadline of a task, only if no other scheduler can meet it.

- Note that “Optimal” in real-time scheduling does not necessarily mean “fastest average response time” or “shortest average waiting time”.

Schedulability Test for RM (Test 1)

- There are n periodic processes, independent and preemptable.
- $Di \geq pi$ for all processes
- Periods of all the tasks are integer multiples of each other
- A necessary and sufficient condition for such tasks to be scheduled on a uniprocessor using RM algorithm:

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

An Example

- Consider a task set with 3 tasks: T1:(4; 1); T2:(2; 1); T3:(8; 2). Is this task set schedulable?

- Note that p1 = 2*p2, p3 = 4*p2 = 2*p1
- The task set belongs to the special class of tasks for which the above schedulability test applies
- Now U = 1/4+1/2+2/8 = 1
- Therefore this task set is RM schedulable

Schedulability Test for RM (Test 2)

If the tasks have arbitrary periods, a sufficient but not necessary schedulability condition is:

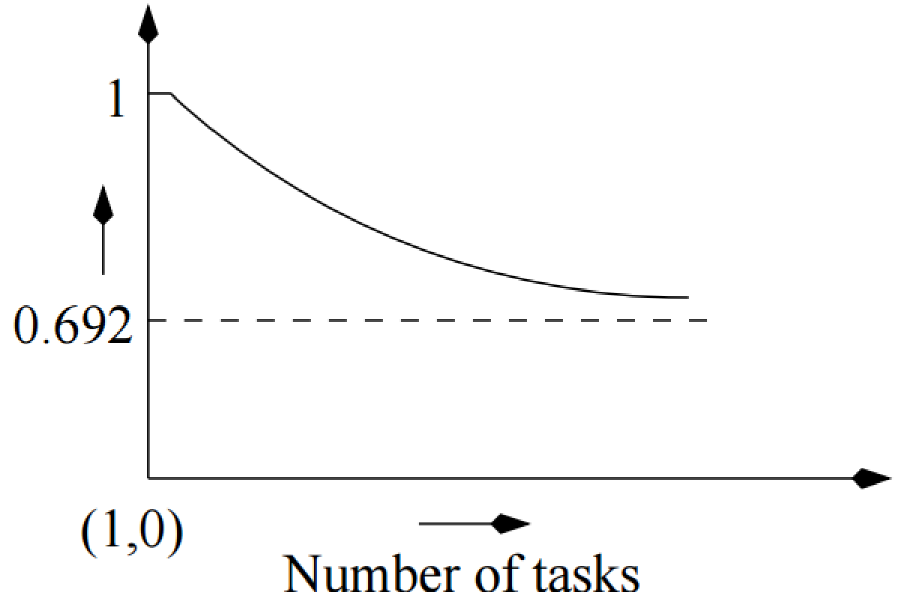
$$U \leq n\left(2^{1/n} - 1\right)$$

where n is the number of periodic tasks.

Task sets with a utilization smaller than $n\left(2^{1/n} - 1\right)$ are schedulable by RM algorithm.

For Different n Values

- Consider the case where there is only one task in the system, i.e. n=1, $U \leq 1$
- Similarly, for n=2, we get $U \leq 0.824$
- For n=∞, we get $U \leq 0.693$.



Proof of case n → ∞

- $\lim_{n\to\infty} n\left(2^{1/n} - 1\right) = \infty \cdot 0$
- Apply L'Hospital's Rule: $\lim_{n\to\infty} n\left(2^{1/n} - 1\right) = \lim_{n\to\infty} \frac{\frac{d}{dn}\left(2^{1/n} - 1\right)}{\frac{d}{dn}\left(1/n\right)}$

$$= \lim_{n\to\infty} \frac{\left(2^{1/n}\right)\ln 2\ \left(1/n\right)'}{\left(1/n\right)'} = \lim_{n\to\infty} \left(2^{1/n}\right)\ln 2 = \ln 2$$

Consider the sufficient condition in Test 2

- If CPU utilization of a system for real-time tasks is less than 69.3% (i.e., U<=69.3%), we know that these periodic tasks are schedulable using RM.
- A set of real-time tasks T1:(4; 1); T2:(2; 1); T3:(8; 2) incurs CPU utilization 100% ($> n\left(2^{1/n} - 1\right)$). Is it still schedulable using RM?
- If a set of tasks are not schedulable, the CPU utilization must be larger than 69.3%. (true or false?)

Schedulability Test for RM (Test 3)

- A sufficient and necessary condition for scheduability by RM algorithm can be derived as follows:
- Consider a set of tasks (T1; T2; … Ti) with (p1 < p2 < p3 < … < pi). Assume all tasks are in phase. The moment T1 is released, the processor will interrupt anything else it is doing and start processing this task as it has the highest priority (lowest period). Therefore the only condition that must be satisfied to ensure that T1 can be feasibly scheduled is that:

$$e1 \leq p1$$

- This is clearly a necessary and sufficient condition.

- The task T2 will be executed successfully if its first instance can find enough time over the time interval (0, p2) that is not used by T1. (p2 is the period of T2 and the first instance of T2 must complete before the second instance arrives)
- Suppose T2 finishes at t. The total number of instances of task T1 released over the time interval [0; t) is $\left\lceil \frac{t}{p_1} \right\rceil$
- If T2 is to finish at t, then every instance of task T1, released during time interval (0; t), must be completed and in addition there must be e2 time available for execution of T2, i. e. the following condition must be satisfied:

$$t = \left\lceil \frac{t}{p_1} \right\rceil e_1 + e_2$$

How To Find Such t for T2?

- Note that every interval has infinite number of points, so we cannot exhaustively check for every possible t.
- Consider $\left\lceil \frac{t}{p_1} \right\rceil$, it only changes at multiples of p1, with jumps of e1. So if we can find an integer k, such that the time t = k*p1$\geq$ k*e1 +e2 and k*p1$\leq$p2, we have the necessary and sufficient condition for T2 to be schedulable under the RM algorithm.

Next, consider Task T3

- It is sufficient to show that the first instance of T3 completes before the arrival of its next instance at p3. If T3 completes its execution at t, then by an argument similar to that for T2, We must have:

$$t = \left\lceil \frac{t}{p_1} \right\rceil e_1 + \left\lceil \frac{t}{p_2} \right\rceil e_2 + e_3$$

- T3 is schedulable iff there is some t ∈(0; p3) such that the above condition is satisfied. Again the right side of above equation changes in multiples of p1 and p2. It is therefore sufficient to check that

$$t = \left\lceil \frac{t}{p_1} \right\rceil e_1 + \left\lceil \frac{t}{p_2} \right\rceil e_2 + e_3$$

is satisfied for some t that is a multiple of p1 and/or p2, such that t ≤ p3.

General Statement of Test 3

Test 3 for schedulability under RM algorithm can now be stated as: The time demand function for task i $(1 \leq i \leq n)$:

$$\omega_i(t) = \sum_{k=1}^{i} \left\lceil \frac{t}{p_k} \right\rceil e_k \leq t$$

$$0 \leq t \leq p_i$$

holds for a time instant t chosen as follows:

$$t = k_j p_j, \qquad (j = 1, \dots . i\ )$$

and

$$k_j = 1, \dots, \left\lfloor \frac{p_i}{p_j} \right\rfloor$$

iff the task Ti is RM-schedulable.

An Example: T1(3, 1), T2(5,1), T3(10, 2)

- For the first i tasks:
- When i=1, k1=1, t=p1=3, so w_1(t) = e1=1< p1=3
- When i=2, k1=1 and k2=1, t=3 and t=5, we have

  w_2(3)=e1+e2=2<3

  w_2(5)=2e1+e2=3<5
- When i=3, k1=1,2,3; k2=1,2; k3=1. So t=3,6,9,5,10 and

  w_3(3) = e1+e2+e3=4>t=3

  w_3(5) = 2e1+e2+e3=5=t

  w_3(6) = 2e1+2e2+e3=6=t

  w_3(9) = 3e1+2e2+e3=7<t

  w_3(10)=4e1+2e2+e3=8<10

So it is RM schedulable

End