Periodic Task Scheduling

- EDF
- Rate Monotonic Algorithm (RMA)
- Deadline Monotonic Algorithm (DMA)

Handling Periodic Tasks using EDF

A periodic task set is schedulable under EDF **if and only if** it satisfies the following criteria:

$$\sum_{k=1}^{n} e_k / p_k = \sum_{k=1}^{n} u_k \leq 1$$

In this expression, we assumed that the period of the task is same as its deadline. In reality this need not be true. In this case, the schedulability criteria need to be changed as:

Thus, if $p_i > d_i$ then each task needs e_i amount of computing time every **Min(p_i,d_i)** duration of time. Therefore we can rewrite the denominator p_i of the above expression as **min(p_i,d_i).**

EDF on periodic tasks (Cont'd)

Note that if $p_i < d_i$ it is possible that a set of tasks is schedulable under EDF even if the task set fails to meet the above criteria.

This means the above criteria is conservative when $p_i < d_i$ and is **not** a necessary condition, but only a sufficient condition for EDF schedulable.

Exercise 4.1: Determine whether the task set given below is EDF schedulable.

$T_1 = (e_1=10, p_1=20)$
$T_2 = (e_2=5, p_2=50)$
$T_3 = (e_3=10, p_3=35)$

Handling Periodic tasks: Rate Monotonic Algorithm (RMA)

- Under EDF there is no strict notion of priority of tasks – It uses deadlines as the key criteria
- In RMA, *tasks are assigned priorities based on the frequency of occurrences*, i.e., tasks arrival rates (Ex: 20msecs task has higher priority than a 50msecs task which in turn has higher priority than a 100msecs task)
- Scheduling Policy: Looking at the task characteristics programmer decides the priority of the tasks ***before*** they are scheduled and then run RMA to schedule the tasks; At any point in time, tasks with highest priority are alone scheduled from the set of ready tasks.

RMA (Cont'd)…

- Optimal Uniprocessor (single CPU/core) static priority scheduling algorithm ($d_i \neq p_i$):

  "*If RMA cannot schedule a set of periodic tasks, no other algorithm can schedule*"

- Schedulability Criteria:

Utilization Bound 1: Sum of the utilization due to tasks must be less than or equal to 1 (Necessary [illegible]

$$\sum_{i=1}^{n} \frac{e_i}{p_i} = \sum u_i \le 1$$

e: Execution time

p: Period of the task

n: Number of tasks

u_i: Utilization due to task i

RMA (Cont'd)…

- Schedulability Criteria: Utilization Bound 2 - Sufficient Condition (by Liu & Leyland 1971)

$$\sum u_i \le n(2^{\frac{1}{n}} - 1)$$

Q: *How does the plot of the bound look like? What do we infer from the trend?*

Q: *For very large n, what is the maximum utilization that can be achieved?*

RMA (Cont'd)…

- Exercise 4.2– Test if the following tasks can be scheduled using RMA;

Task: (execution time, periodicity of the task, deadline=period)

T1: (20,100); T2: (30,150); T3: (60,200)

Solution to be discussed in the lecture

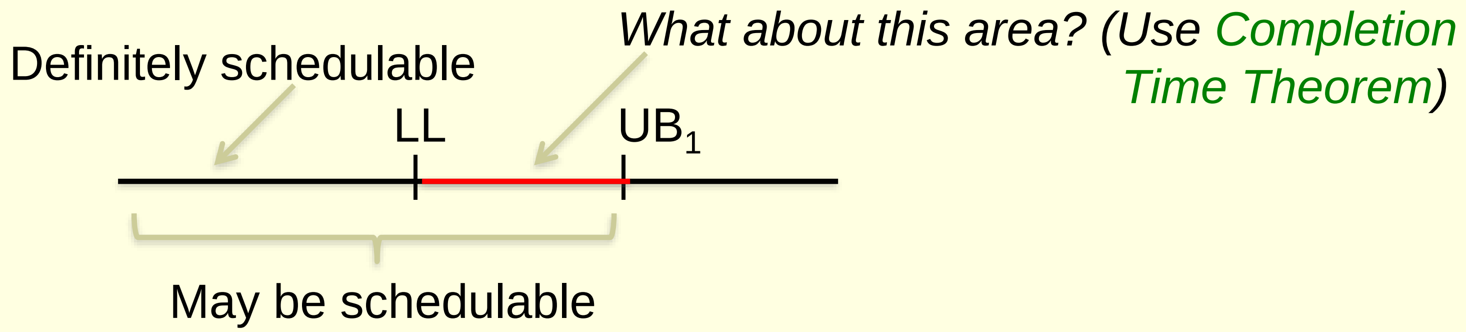
- Liu & Leyland condition– Conservative criteria!

  It assures only 69% (ln 2) utilization as n tends to infinity!

- It is only a *sufficient* condition – That is, **if** LL bound is satisfied then definitely tasks are schedulable using RMA;

RMA (Cont'd)…

- However, if a task set fails to satisfy LL bound, still it may be schedulable! This is shown by ***Completion Time Theorem*** (Liu & Lehoczky, 1989)

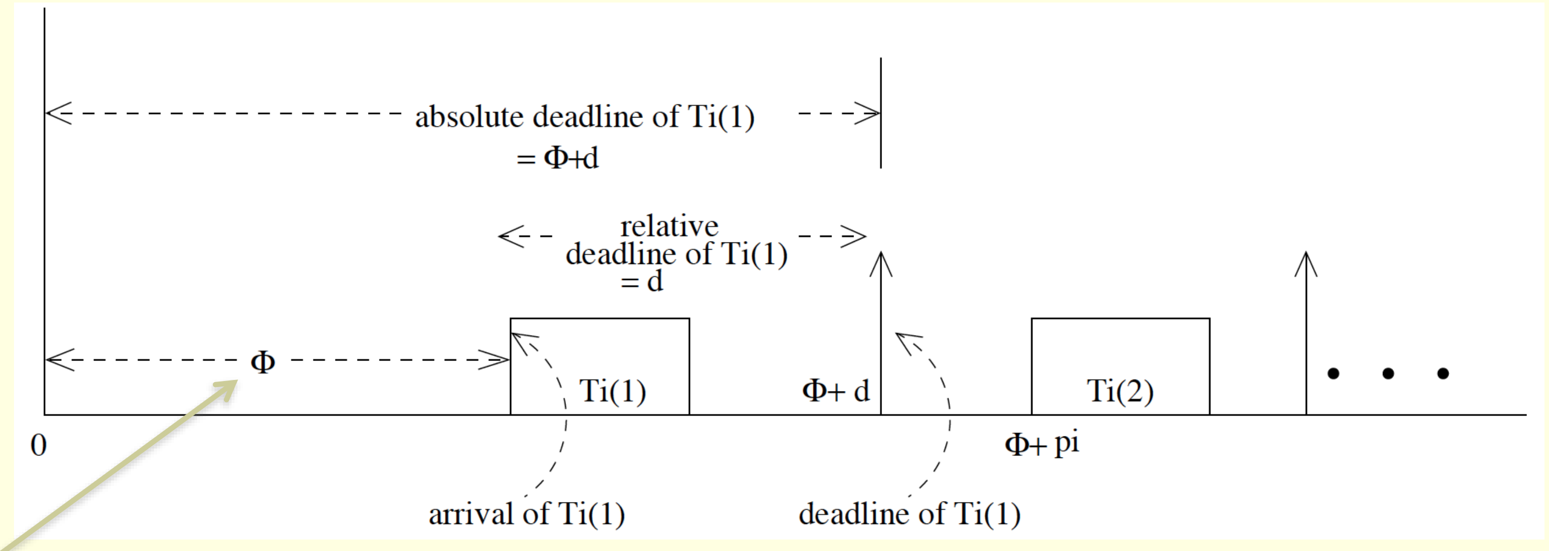


Completion Time Theorem: If each of a set of tasks individually meets its first deadline under zero phasing then the task set is RMA schedulable for all task phasings. (*Lehoczky criterion*)

RMA (Cont'd)…

Checking the schedulability of a task set:

1. Consider zero phasing for all tasks
2. Derive schedules till the first deadline of each task
3. Observe if each task is schedulable
4. Then the entire task set is schedulable using RMA



Phase (delay in the occurrence of the first instance of a task i)

RMA (Cont'd)…

- Exercise 4.3– Test if the following tasks can be scheduled using RMA ;

Task: (execution time, periodicity of the task, deadline=period)

T1: (20,100); T2: (30,150); T3: (90,200)

Solution to be discussed in the lecture

- Exercise 4.4 – Can we derive a formal expression based on the above observation from Exercise 4.3?

Solution – Solution will be provided; But follow Ex 4.3 to generalize.

RMA - Analysis with context switching overhead

So far the influence of overheads were ignored. Let us include an overhead – context-switching time to see its effect on the schedulability criteria and hence, the performance of the algorithm.

We assume that the context switching time is constant and equals c secs. Thus the increase in execution time of e_i of each task T_i is at most (e_i+2c). The factor 2c is the worst case number of content switches per task (per preemption) - first switch when a task preempts a currently running task and another c when this task completes.

Thus in the RMA schedulability criterion, we replace e_i by (e_i+2c) for each T_i.

Exercise 4.5**:** Determine whether the task set given below is RMA schedulable. $T_1 = (e_1=20, p_1=100)$; $T_2 = (e_2=30, p_2=150)$; $T_3 = (e_3=90, p_3=200)$; Overhead factor **c** does not exceed 1 msec.

Task self-suspension

A task might suspend itself from execution when it needs to perform I/O operations or when it waits for some events/conditions to occur/satisfy.

So, what happens when a self suspension happens?

When a task suspends itself, it is removed from the ready queue and it will be put in a blocked queue. OS takes next eligible ready task to run.

We will derive a formal expression for a revised schedulability criterion when a task undergoes at most single suspension.

Task self-suspension

Consider an ordered set of tasks $\{T_1,\ldots,T_n\}$, such that $pri(T_j) > pri(T_{j+1})$. When a task T_i suspends itself we are concerned about *how much time this overhead will add to the overall execution* such that the given task set is schedulable.

Assume that each higher priority task suspends only once.

Then, if b_i denotes the duration of a task suspension time then the total delay (bt_i) that task T_i might incur**:**
(b_i) due to its own self suspension + (single) suspension of all higher priority tasks influences the finish time of task T_i.

Question is by how much?

Task self-suspension

$bt_i = b_i + \Sigma_k \min(e_k,b_k), k = 1,\ldots,i-1$ (*worst case*)

Convince yourself with the following values. Let k be a higher priority task and i be a lower priority task and let $e_i = 10$

Case 1: Let $e_k = 3$, $b_k = 5$ ($e_k < b_k$)

Case 2: Let $e_k = 5$, $b_k = 3$ ($e_k > b_k$)

In Case 1, max execution of i happens during b_k and i needs to wait for e_k to finish. Hence, it is $e_k = \min(e_k, b_k)$ influences the response time.
In Case 2, max possible execution by task i only upto b_k (overlap period of suspension time of k), which is again a min of e_k and b_k.

Hence in both cases $\min(e_k, b_k)$ is to be considered.

Task self-suspension

Exercise 4.6 First derive a generalized version to take into account the self suspension overheads. Then, verify if the following tasks are RMA Schedulable with self-suspension overheads.

T1(10,50), T2(25,150), T3(50,200). Let the respective maximum task suspension times be 3, 3, and 5 msecs.

Note that in this case tasks are already in increasing rates.

RMA for Harmonic tasks

While RMA has an interesting property in deciding the priority of the tasks, it is still difficult to know exactly the start time and finish time of each task within a given period. Especially for tasks with low priority, i.e., long periods, the start and finish times could be different.

The *hyper-period = LCM(p_i : for all i)* is by and large too big. However, suppose if the task periods are in harmonic progression, i.e. multiples, it is possible to find the start time and finish time of each task quickly because the resulting schedule becomes more regular.

Remark: Assumes periodic time-triggered inputs by an application in which case start times of tasks may be predictable with high confidence.

RMA for harmonic tasks

The task periods in a task set are said to be harmonically related, IFF for any two tasks i and k, whenever $p_i > p_k$, it should imply that p_i is an integral multiple of p_k. For example, T1 = (5,30), T2=(12,60), T3=(8,120) are harmonically related tasks.

Theorem: For a set of harmonically related tasks, RMA schedulability criterion is given by

$$\Sigma_i(e_i/p_i) = \Sigma_i u_i \leq 1.$$

Proof - Exercise 4.7 - *To be discussed during the lecture.*