

Real Time Systems
Professor Rajib Mall
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur
Lecture 17
Rate Monotonic Scheduling Analysis

Welcome to this lecture, just to recollect, we were discussing about the even driven schedulers, and we said that there are two important schedulers for uniprocessors, if we understand those two, we are pretty much done understanding of the uniprocessor scheduling real time systems.

And these two are the rate monotonic and EDF, we had looked at the EDF scheduler, and we had identified its advantages disadvantages its implementation and so on. And we were discussing about the rate monotonic scheduling and we had discussed about few basic aspects of real time scheduling, using a rate monotonic scheduler. Now, let us proceed from that point onwards.

(Refer Slide Time: 01:08)

RMS (Rate Monotonic Scheduler)

- The priority of a task is proportional to its rate of arrival (frequency).
- The higher the arrival rate (or lower the period) of a task, the higher is its priority.

The slide features a graph with 'Priority' on the vertical axis and 'Frequency (Rate)' on the horizontal axis. A blue line starts from the origin and slopes upwards. Two red circles are placed on the line: one lower and further to the left, labeled '10 msec', and one higher and further to the right, labeled '1 msec'. To the right of the graph is a circular video inset showing Professor Rajib Mall. At the bottom of the slide, there are two logos: the Indian Institute of Technology Kharagpur logo on the left and a smaller logo on the right.

The rate monotonic scheduler, the crucial thing is that the tasks are assigned fixed priority by the designer, and the priority assigned to a task is proportional to its rate of arrival or the inter arrival time, and the faster the task arrives, that is the smaller is its inter arrival time at the task period, then the higher is its priority, the more frequent is the task, the higher is its priority.

And if we draw the frequency versus the priority, it is a linear plot and if a task has let us say arrival of 10 millisecond is the interarrival time then it will have a lower priority compared to a

task whose interarrival time is 1 millisecond. So, the faster the task arrives, the higher is the priority assigned to it by the designers.

(Refer Slide Time: 02:49)

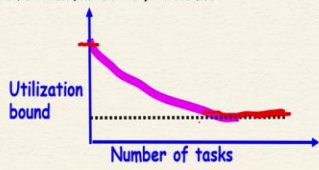
Sufficient Condition for RM Schedulability

- Utilization bound II (Liu and Layland 1971):

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

$1(2^1 - 1) = 1$

- u_i is the processor utilization due to task T_i
- n is the number of tasks



Now, an important problem that we need to address is that, if our application has a certain task set, can that be feasibly scheduled on a uniprocessor, and that is given in terms of utilization bounds, the first bound is intuitive we had discussed about the first bound, which is a necessary condition is that the sum of the utilization of all the tasks is less than equal to 1, means that the processor cannot be more busy than fully busy.

The utilization due to all tasks can be at most 100 % or 1. Now, the second utilization bound result was given by Liu and Layland way back in 1971, and they are elaborate derivation came up with this expression, we are not going to go through the derivation, it is not necessary actually to go through the derivation, what is more important is this result.

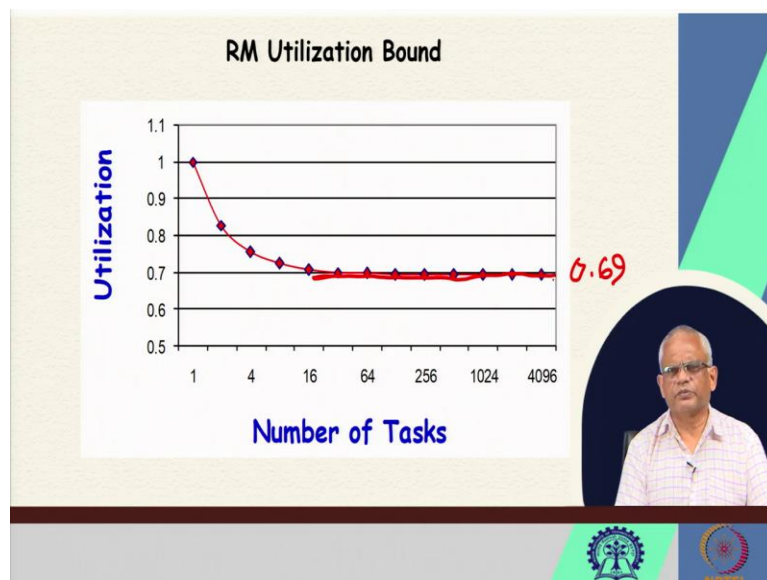
The result says that the $\sum u_i \leq n(2^{1/n} - 1)$, where n is the number of tasks in the application, so there are 2 terms here $n(2^{1/n} - 1)$. And if we plot the utilization bound, it kind of gives a bound see here that the utilization needs to be utilization due to the tasks in the application should be less than some bound, and the bound varies with a number of tasks in the application.

If the number of tasks in the application is only 1, then it simplifies the expression is $1(2^1 - 1) = 1$ or 100 %. With a single task, we can achieve 100 % utilization, and the task set will be still

feasibly scheduled. But then as the number of task set increases, the utilization bound decreases, let us look at these two terms $n(2^{1/n} - 1)$, and as n increases the first term increases linearly, but then the second term it decreases, because of the exponential factor here $2^{1/n}$ it decreases.

So, the decrease with n for the second term is much more than the increase in n of the first term, and therefore it keeps on decreasing, but at some point it just does not increase much, it just remains there. If we substitute large values for n , we will see that at what value it remains constant.

(Refer Slide Time: 06:42)

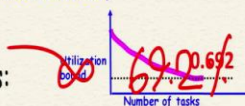



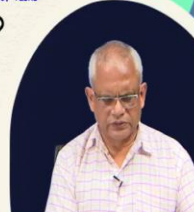
If we plot it for 1 task, it is 100 % utilization, then with 4 tasks, it is kind of 75 % and with 10 tasks or so it is 70 %, and then it just stabilizes just below 70 %, 0.69 to be precise, it just stabilizes at that point. So, as the number of tasks increases beyond 20 or so, it just remains 0.69.

(Refer Slide Time: 07:29)

Liu and Layland Bound

- The maximum utilization for a schedulable task set:
 - Falls as the number of tasks increases.
- For a very large number of tasks:
 - What is the maximum utilization permitted?


$$\sum u_i \leq \infty \left(2^{\frac{1}{\infty}} - 1 \right) = \ln 2 = 0.692$$



So, we could see from the plot that the maximum utilization with which a task set is feasibly schedulable falls as the number of tasks increases, and suppose the number of tasks is very large, let us say infinity, so what will be the maximum utilization that is permitted? For ∞ tasks, the two terms, one term becomes ∞ and the other term says $2^{(1/\infty)} - 1 = 0$, so $\infty \cdot 0 = 0$.

So, it is indeterminate form infinite into 0 form, and if we apply the L hospital's rule, it will give the result $\ln 2$, and $\ln 2$ is 0.692. So, as the number of task increases, the utilization bound converges to 0.692 or we can say 69.2 % is the maximum utilization of the processor that can be achieved due to this task set and for the tasks to be feasibly scheduled and their deadlines are met. So, this is the 69.2, as you can see that 0.692 is the point at which it establishes.

(Refer Slide Time: 09:48)

The slide is titled "RM Analysis -- Example 1". It contains the following elements:

- Task set:** $T_i = (e_i, p_i)$
 $T_1 = (2, 4)$ and $T_2 = (1, 8)$
- Schedulability check:**
 $2/4 + 1/8 = 0.5 + 0.125 = 0.625 \leq 2/(2^{1/2} - 1) = 0.82$
- Gantt chart:** A timeline from 0 to 8. Task T_1 (execution time 2, period 4) is scheduled at intervals [0, 2], [4, 6], and [8, 10]. Task T_2 (execution time 1, period 8) is scheduled at intervals [2, 3] and [10, 11].
- Handwritten notes:** Red circles around the numbers 2, 4, 1, 8 in the task set and around the numbers 0.625 and 0.82 in the schedulability check. A handwritten formula $n(2^{1/n} - 1)$ is written above the check. A small diagram shows two overlapping rectangles representing task execution.
- Speaker:** A man in a light-colored shirt is visible in a circular inset on the right side of the slide.
- Logos:** Logos for IIT Bombay and NPTEL are at the bottom right.

Now, let us try to apply this result, the Liu Layland result to few applications and see whether these applications can feasibly run on a uniprocessor when we use a rate monotonic scheduler, and we are saying that rate monotonic schedulers are very popular in real time systems. The task set that is given to us is in terms of their execution time and period and the deadline and the period are the same, so for the task 1, 2 is the execution time and 4 is the period. So, every 4 unit some time we need to execution units.

Similarly, for task 2 every 8 units we need 1 unit of execution, can this be feasibly scheduled on a rate monotonic scheduler and uniprocessor, that is the problem that we are going to analyse. And for that, we will have to compute the utilization due to these two tasks and check the bounds. So, the utilization due to the first task is $2 / 4$, and for the second task it is $1 / 8$.

And if we simplify we will get 0.625. But the Liu Layland bound is $n(2^{1/n} - 1)$ and here n is 2 we have 2 tasks in the application, so $2(2^{1/2} - 1)$ and $2^{1/2}$ is $1.41 - 1 = 0.41 \times 2 = 0.82$. And $.625 < 0.82$, and therefore the task set according to Liu and Layland, it is actually schedulable.

Now, let us draw schedule according to the rate monotonic scheduler, how will the tasks be scheduled, we know that at any time the highest priority task will run and we assign the task priorities, we assign priority 1 to T_1 and priority 2 to T_2 . So, whenever T_1 arrives, it runs and it

if T2 is running, it will preempt T2. So, that is the basic premise of the rate monotonic scheduler. Now, let us try to draw the schedule for T1 and T2.

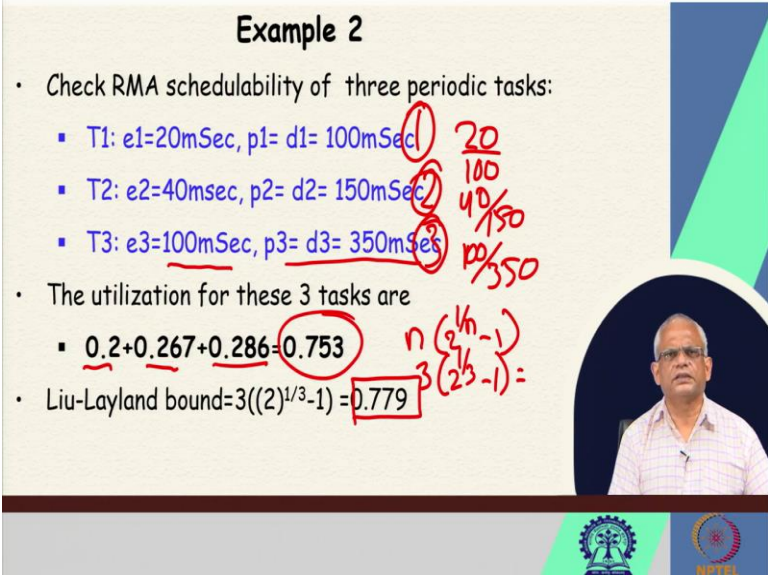
At time 0, let us say both tasks arrive and T1 will start running for 2 units of time because that has the highest priority and from 2 to 4 T2 can run, but T2 requires only 1 unit of time. So, T1 runs 0 to 2 and T2 runs 2 to 3. And then 1 unit of idle and then T1 again runs and then T2 is not there, so we can it will be idle slot here and so on.

So, we can draw the schedule and see that the task set meet their deadline as long as you can think of, as long as you can try in the time, they just keep on meeting their deadlines. So, yes, that validates the Liu Layland that if the utilization < 0.82 which is 0.625 here, then the tasks set that will meet their deadlines.

(Refer Slide Time: 14:29)

Example 2

- Check RMA schedulability of three periodic tasks:
 - T1: $e_1=20\text{mSec}, p_1= d_1= 100\text{mSec}$ $\frac{20}{100}$
 - T2: $e_2=40\text{msec}, p_2= d_2= 150\text{mSec}$ $\frac{40}{150}$
 - T3: $e_3=100\text{mSec}, p_3= d_3= 350\text{mSec}$ $\frac{100}{350}$
- The utilization for these 3 tasks are
 - $0.2+0.267+0.286=0.753$ $n \frac{(2^{1/n}-1)}{(2^{1/n}-1)}$
- Liu-Layland bound $= 3((2)^{1/3}-1) = 0.779$ $3 \frac{(2^{1/3}-1)}{(2^{1/3}-1)}$



Now, let us try another example. So, here we have 3 tasks, the first task execution is 20 millisecond and the period and deadline are the same it is 100 millisecond, the second task execution time is 40 milliseconds and the period and deadline are 150 milliseconds, the third task execution time is 100 milliseconds and the period deadline are 350 milliseconds, now are these schedulable on a uniprocessor, that is the problem that we want to try out.

And the first step here is to compute the utilization due to the 3 tasks, the first task utilization is $20 / 100$, second task $40 / 150$, third task $100 / 350$, so $20 / 100 = 0.2$, $40 / 150 = 0.265$ and $100 /$

$350 = 0.286$, and if we sum them up, we get 0.753 , and if we compute the Liu Layland bound $n(2^{1/n} - 1) = 3(2^{1/3} - 1)$, if we simplify this we will get 0.779 .

And $0.753 < 0.779$ which is the Liu Layland bound, and therefore we can say these task set can be run feasibly scheduled by a on a uniprocessor using the rate monotonic scheduler, and here the programmer would have to assign priority 1 to task 1, priority 2 to task 2 and priority 3 to task 3.

(Refer Slide Time: 16:54)

Example 3

- Check whether the following task set is schedulable using a RMS:
 - T1: $e_1 = 1, p_1 = \underline{4}, d_1 = 4$ $\frac{1}{4} = 0.25$
 - T2: $e_2 = 2, p_2 = \underline{6}, d_2 = 6$ $\frac{2}{6} = 0.33$
 - T3: $e_3 = 3, p_3 = \underline{20}, d_3 = 20$ $\frac{3}{20} = 0.15$



Now, let us look at another different example. So, here again we have been given a task set with 3 tasks, T1, T2, T3, and their periods are 4, 6 and 20, the utilization due to the first one is $1 / 4$ which is 0.25 , the second one is $2 / 6$ which is $1 / 3$ is 0.33 , the third one is $3 / 20$ which is 0.15 , sorry 0.15 point whatever 0.15 .

(Refer Slide Time: 18:01)

Solution to Example 3

$$\sum_{i=1}^n \frac{e_i}{p_i} = \sum u_i = \frac{1}{4} + \frac{1}{3} + \frac{3}{20} = \frac{44}{60} = \frac{11}{15} \leq 1 \checkmark$$
$$n(2^{\frac{1}{n}} - 1) = 3(2^{\frac{1}{3}} - 1) = 3(1.259 - 1) = 0.778$$
$$\sum u_i = \frac{11}{15} = 0.733 \leq 0.778 \checkmark$$

- Therefore the task set is schedulable.





So, now we need to add them up and that comes out to be $44 / 60$, and this simplifies to $11 / 15 < 1$ which is the necessary condition the utilization bound 1, so the utilization bound 1 is met, now let us check for the utilization bound 2. The Liu Layland bound is given by $2^{1/3} - 1 = 1.259 - 1 = 0.259 * 3 = 0.778$. And $11 / 15$ is $0.733 < 0.778$, and therefore the task set can be feasibly scheduled on a uniprocessor.

(Refer Slide Time: 19:01)

A Basic Question...

- Do EDF and RMS produce identical schedules?
- Construct an example for which EDF and RMA produce different schedules.
- Possibly simplest answer is:
 - Consider a task set unschedulable in RMA but schedulable in EDF.
- But when a task set is feasibly scheduled by both, do they produce identical schedules?



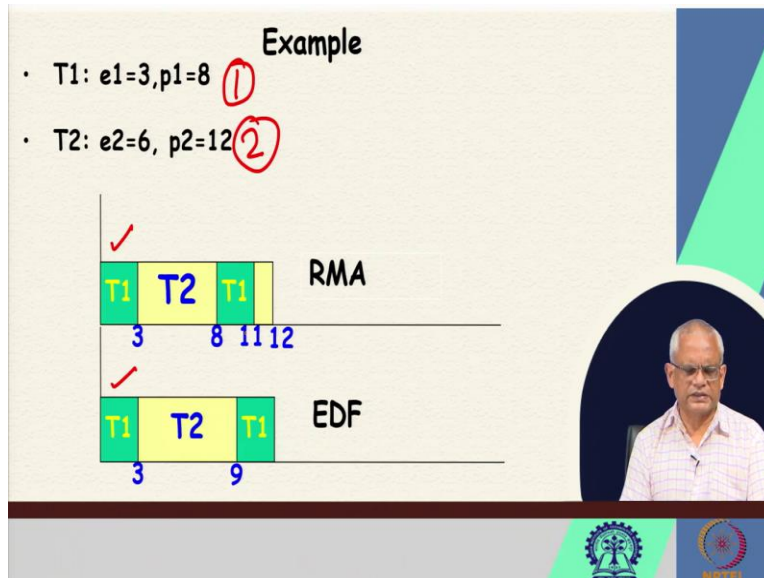
But, so far we have looked at whether given a task set can that be run feasibly on a uniprocessor using a rate monotonic scheduler, and had computed the utilization due to the tasks and sum to the utilization due to the task and check whether it is less than the utilization bound given by the Liu Layland criteria, but before we proceed further, let us try to answer a very basic question.

The basic question is that, given a task set, do EDF and RMS the rate monotonic the earliest deadline first and rate monotonic scheduler, do they produce very different schedules for the tasks or they produce identical schedules? To answer this question you can try out with various tasks set and check whether EDF and RMS produce different or identical schedules, you will find that in almost all cases they produce identical schedules.

But are there any cases where EDF and RMS may produce different schedules, one thing that we can say is that, if there are 3 tasks and the utilization is let us say 0.85 and the utilization bound given by the Liu Layland criterion is 0.778, then the task cannot be run feasibly in rate monotonic, but it is feasibly run in EDF and therefore EDF must be producing a different schedule we can argue like that, and we can actually draw that and so that they produce different schedules.

But agreed that if a task set is not schedulable in RMS but is schedulable in EDF, they would be producing different schedules okay agreed, but then is there a situation where a task set is feasibly scheduled in both rate monotonic and EDF, but they produce different schedules, can we have an example of that?

(Refer Slide Time: 21:51)



We can try that. Let us, just give one example here, e_1 task 1 execution time is 3 and the period and deadline are 8, task 2 execution time is 6 and the period and deadline are 12. Now, we can see that this is feasibly scheduled on a rate monotonic scheduler, so initially T1 runs then T2 runs up to 8 and then T1 runs, because T1 will have the priority 1 highest priority T2 is the second highest priority, the lower priority than T1 and to start with T1 will start running for 3 units of time, and then T2 needs 6 units of time.

But then at 8 T1 again arrives and runs for 3 units of time, but T2 had run only 5 units of time between 3 to 8, and therefore it runs at 11 and by 12 it completes, so the deadline for T1 and T2 are both met. Now, let us look at the EDF schedule, in EDF the earliest deadline is T1 and therefore T1 runs, and then T2 starts running and the deadline of T2 is 12 and T1 once it arrives at 8 its deadline is 16. And therefore T1, therefore T2 continues to run up to its completion up to 9 and then only T1 runs. So, the schedule produced is different for this example.

(Refer Slide Time: 23:58)

Liu and Layland Condition

- The upper bound on utilization converges to 69% ($\ln 2$):
 - As the number of tasks approaches infinity
- Liu and Layland's condition is conservative: *Sufficient*
 - If a set of tasks passes Liu and Layland test, then it is definitely RMA schedulable.
- But, even if a task set fails Liu-Layland test:
 - *Still it may be RMA schedulable (completion time theorem).*

The Liu Layland bound is very pessimistic, because the processor is only 70 %, 69 % utilized, the rest of time it idles and also is it possible that the Liu Layland condition even though it is a sufficient condition that Liu Layland condition is a task set fails the Liu Layland condition but then it is actually schedulable.

The Liu Layland condition is conservative, it is the sufficient condition, it is a sufficient condition as long as the task set meet the Liu Layland condition, we can confidently say that yes the task set will be scheduled on the rate monotonic scheduler on a uniprocessor. But is there a case that even if a task set fails the Liu Layland test and still it is schedulable? Yes, that is very likely that it fails the Liu Layland test.

If it passes Liu Layland condition, we do not have to look further, we are confident that it will run feasibly, but if it fails the Liu Layland condition then we need to check that is it really not schedulable, or is there a possibility that the task set can be feasibly scheduled. And that we will discuss with the help of the completion time theorem.

Once a task set fails the Liu Layland condition then we need to apply the completion time theorem. If it passes Liu Layland condition, nothing to worry we can just confidently develop the application and run on a uniprocessor and it will run satisfactorily. But if the task set fails the Liu

Layland condition we need to try out the completion time theorem, and check if it passes that, if it passes then it is schedulable even though it failed the Liu Layland condition.

So, we are almost at the end of this lecture, we will take, we will first discuss about the completion time theorem and we will apply this completion time theorem to cases where a task set fails the Liu Layland condition and check that whether it is actually not schedulable or is it can be feasibly scheduled on the uniprocessor. We will stop here for this class, thank you.